一台虚拟机性能比不上服务器，虚拟机是以量取胜。

devops：python、git、jenkins、ansible

yum源：https://www.centos.org/download/mirrors/

json对象：嵌套键值对

ELK:是一套解决方案 Elasticsearch：负责日志检索和存储

Logstash：负责日志的收集和分析、处理 Kibana：负责日志的可视化

分布式（分片）优点：可拓展、容量大，节点性能高

缺点：故障率高（解决方法：原数据+副本数[少于节点数量]）

elk流程：web访问写进日志--> filebeat发送

--->logstach配置文件logstash.conf-->input filter output

--->Elasticsearch存储---kibana展示

部署es集群：Elasticsearch:全文索引，倒排索引【负责日志检索和存储】

yum -y install elasticsearch.noarch java-1.8.0-openjdk

配置文件：

egrep -v "^(#)" /etc/elasticsearch/elasticsearch.yml

cluster.name: nsd1808 ----集群名称（相同）

node.name: es1 ----节点名称（不同）

network.host: 0.0.0.0 ----节点ip

discovery.zen.ping.unicast.hosts: ["es1", "es2", "es3"] 声明节点

systemctl start elasticsearch------->声明节点一定要先启动

ss -ltun

vim /etc/hosts

测试：curl 192.168.1.11:9200/\_cluster/health?pretty

curl -X "DELETE" http://ip

curl -A 修改请求user-agent

-X 设置请求方法（PUT、DELETE、POST、GET..）

-i 显示返回头信息 -I 显示头部信息

下载插件：(head插件:索引和节点级别操作，查询API展现集群的各种状态

kopf插件：管理工具 bigdesk插件：集群监控工具)

./plugin install file:///root/elasticsearch-kopf-master.zip

./plugin install ftp://192.168.1.254/plugin/elasticsearch-head-master.zip

./plugin install ftp://192.168.1.254/plugin/bigdesk-master.zip

测试：http://192.168.1.15:9200/\_plugin/head/

拓展插件：RESTful API（增删查改）调用

curl http://192.168.1.13:9200/\_cat/?h -----帮助

curl http://192.168.1.13:9200/\_cat/nodes?v -----详细信息

创建index

curl -X PUT "http://192.168.1.15:9200/index" -d '

{

"settings":{

"index":{

"number\_of\_shards":5, //分片数

"number\_of\_replicas":1 //副本数

}

}

}'

插入：JSON格式

curl -X PUT "http://192.168.1.15:9200/taindex/teacher/1" -d '{

"职业":"诗人",

"名字":"李白",

"称号":"诗仙",

"年代":"唐"

}'

修改数据：

curl -X PUT "http://192.168.1.15:9200/taindex/teacher/1" -d '{

"doc":{

"年代": "唐代"

}

}'

查：curl -X GET "http://192.168.1.15:9200/taindex/teacher/3?pretty"

删数据：curl -X DELETE "http://192.168.1.15:9200/taindex/teacher/3?pretty"

删索引：curl -X DELETE http://192.168.1.15:9200/taindex/

curl -X DELETE http://192.168.1.15:9200/\*

部署kibana：/opt/kibana/config/kibana.yml

测试：http://192.168.4.20:5601

导入:shakespeare.json.gz(自带表结构)

curl -XPOST http://192.168.1.11:9200/\_bulk --data-binary @shakespeare.json

导入:不带表结构

curl -XPOST http://192.168.1.11:9200/xi/ha/\_bulk --data-binary @accounts.json

查询数据：

curl -XGET http://192.168.1.11:9200/\_mget?pretty -d '{

"docs": [

{

"\_index": "shakespeare",

"\_type": "act",

"\_id": "0"

},

{

"\_index": "logstash-2015.05.19",

"\_type": "log",

"\_id": "AWfoJ-yiHKGwKqjdYTV3"

}

]

}'

logstash:

map映射：mapping，创建索引的时候，预先定义字段的类型及相关属性

分为动态映射（自动根据数据进行相应的映射）和静态映射自定义字段映射数据类型

input（收集）---filter（数据处理）---output（输出） codec编码解码

cat 0<logstash >&1

cd /opt/logstash/bin/

alias logstash=/opt/logstash/bin/logstash

cd /etc/logstash/

vim logstash.conf

input{

file {

path => ["/tmp/a.log", "/var/tmp/b.log"]

sincedb\_path => "/var/lib/logstash/since.db"

start\_position => "beginning"

type => "test log"

}

tcp {

mode => "server"

host => "0.0.0.0"

port => 8888

type => "tcplog"

}

udp {

port => 8888

type => "udplog"

}

syslog {

port => 514

type => "syslog"

}

}

filter{}

output{

stdout{ codec => "rubydebug" }

}

logstash -f /etc/logstash/logstash.conf

测试：

文件描述符:/proc/self/fd/pid/

function send() { exec 9<>/dev/tcp/192.168.1.20/8888;

echo "$1" >&9; exec 9<&-; }

send hshs

echo "abcd" >/dev/udp/192.168.1.20/8888

定义日志：

man logger --->/example

vim /etc/rsyslog.conf

local0.info /var/log/t.log

local0.info @@192.168.1.20:514 --->发送到另一台主机

systemctl restart rsyslog.service

日志发送测试：logger -p local0.info -t testlog "ooxx"(默认是/var/log/messages)

logstash正则文件：/opt/logstash/vendor/bundle/jruby/1.9/gems/ \

logstash-patterns-core-2.0.5/patterns/grok-patterns

yaml:一个可读性高，用来表示数据序列的格式

Ansible:

一台虚拟机性能比不上服务器，虚拟机是以量取胜。需要批量部署

Devops:python、git、jenkins、ansible

yum源：https://www.centos.org/download/mirrors/

os storage(ceph,gluster) cloud(openstack) extras(ansible)

Ansible:IT自动化和devops软件,基于python实现,批量操作系统，程序部署，运行命令

可以实现自动化部署，管理配置项，持续交付，AWS云服务管理

只需ssh和python，无客户端，模块丰富，社区活跃，上手容易，支持自定义及多层部署

Ansible七种武器：

Ansible命令行命令，执行临时性工作 重要

Ansible-doc是ansible模块的文档说明 重要

Ansible-playbook通过读取预先编写好的playbook文件实现批量管理 重要

Ansible-console提供虚拟终端上的交互式工具

Ansible-pull/push 主动/被动模式

Ansible-galaxy从github下载管理roles的工具

Ansible-vault 用于配置文件加密，也可解密

源码安装：git clone git://github.com/ansible/ansible.git

yum -y install python-setuptools python-devel

python setup.py build && python setup.py install

pip install ansible

配置两方面：主机分组，命令分组及playbook

下载ansible软件包

修改配置文件/etc/ansible/ansible.cfg

14 inventory = /etc/ansible/hosts (打开注释定义主机配置文件,有额外参数)

61 host\_key\_checking = False (打开注释,跳过ssh\_key检查)

Inventory包含静态和动态主机[通过脚本获取主机列表（Json格式）]

配置主机：/etc/ansible/hosts

[组名]

IP

主机名 ansible\_ssh\_pass="密码"

db[1:2]

...

[组名:children]

组名1

组名2

[组名:vars] 定义公共变量

ansible\_ssh\_pass="a"

ansible配置文件查找顺序：

1.ANSIBLE\_CONFIG变量定义的配置文件 2.当前目录下的./ansible.cfg文件

3.检查当前用户家目录下~/ansible.cfg文件 4.检查/etc/ansible/ansible.cfg文件

格式：ansible 主机/分组 参数 模块名称 模块的参数

ansible web,db[all] --list-hosts 查看主机

ansible web -m ping -k 交互式检测（这个ping就是ssh一次）

ansible all -m command -a "uptime" -k 批量执行命令

调用外部命令会产生子进程。 AA=$（< id\_rsa.pub）

生成密钥：ssh-keygen -t rsa -b 2048 -f id\_rsa -N ""

批量部署密钥：ansible web,db,other -m authorized\_key -a "user=root

exclusive=true manage\_dir=true key='$(< /root/.ssh/id\_rsa.pub)'" -k

查看模块：ansible-doc -l / ansible-doc 模块名 ---查看所有模块及帮助

ping 模块 shell|raw 模块 command 模块(不启动bahs模块)

bash功能:< > | & chdir=切换路径在执行命令

ansible cache -m shell -a 'chdir=/tmp touch f1'

执行命令：ansible 主名/组名 -m 模块 -a '命令'

ansible web1,db2 -m shell -a 'chage -d 0 z3'

执行脚本：ansible web -m script -a '执行路径'

ansible web -m script -a '/tt.sh'

拷贝copy模块：src源文件 dest目标文件 （适合相同文件拷贝）

覆盖前是否先备份原文件（backup：yes|no） 是否强制覆盖（force yes|no）

ansible all -m copy -a 'src=/etc/resolv.conf dest=/etc/resolv.conf'

编辑替换模块：lineinfile:参数path regexp line 编辑[配置]文件

replace :参数path regexp replace 编辑[配置]文件

ansible cache -m lineinfile -a 'path="/etc/sysconfig/network-scripts/ifcfg-eth0"

regexp="^ONBOOT" line="ONBOOT=\"no\""'

ansible cache -m replace -a 'path="/etc/sysconfig/network-scripts/ifcfg-eth0"

regexp="^(ONBOOT).\*" replace="\1=\"yes\""'

yum :参数 1.name 包1，包2.. 2.state 动作(installed removed)

service:参数 1.name 服务名 2.state 动作 3.enable yes|no 开机是否自起

查看是否开机自起：systemctl is-enabled 服务名

ansible web1 -m yum -a 'name=telnet state=installed'

ansible web1 -m service -a 'name=chronyd state=started enabled=yes'

统计信息:ansible cache -m setup [-a 'filter="ansible\_nodename"'] 输出Json格式

Json语法规则：数据在名称/值对中---键值对key:value -----中文要用""

,数据由逗号分隔---对象或数组间的值/键值对分隔

{}大括号保存对象----------一个人的不同方面(姓名， 爱好..)

[ ]中括号保存数组----------不同人的相同元素(姓名1， 姓名2..)

YAML语法规则：playbook

结构通过空格来展示，一定不要用tab键

数组使用"-"来表示

键值对使用":"来表示

使用固定的缩进风格(双数空格、对齐)表示数据层级结构关系

#表示注释

Jinja2语法：playbook模板

模板的表达式(变量)都是包含在分隔符"{{}}"内的

控制语句都是包含在分隔符"{% %}"内的

模板支持注释，都是包含在分隔符"{# #}"

变量可通过过滤器修改，( | ),圆括号传递可选参数,多个过滤器可以链式调用

ad-hoc：点对点模式

playbook构成：Target（主机列表） Variable（运行变量）

Tasks （任务列表） Handler（task执行完后调用的任务列表）

节点并发执行,顺序命令 返回颜色：绿色表成功 黄色表修改成功 红色失败

执行playbook命令：ansible-playbook [-f 5] [-e ‘传参’]ping.yml

cat ping.yml

---

- hosts: web,db..

remote\_user: root

tasks:

- name: add user #解释

user:

name: lly

- name: "安装并修改端口及配置文件，设置开机自起"

---

- hosts: web2

remote\_user: root

vars:

username: wo

tasks:

- name: add user "{{username}}"

user:

name: "{{username}}"

password: "{{'123'|password\_hash('sha512')}}" 过滤器

- shell: chage -d 0 "{{username}}"

1.ansible-playbook ping.yml 直接执行

2.ansible-playbook -e '{"username": "100"}' ping.yml 命令行传参

3.ansible-playbook ping.yml -e '@aa' Json文件传参

负载逐渐增加：awk 'BEGIN{while(1){}}'

register和when---当cpu负载大于0.7时，关闭httpd服务

---

- hosts: web

remote\_user: root

tasks:

-shell: uptime|awk '{printf("%.2f",$(NF-2))}'

register: result

-service:

name: httpd

state: stopped

when: result.stdout | float > 0.7

循环：

---

- hosts: web

remote\_user: root

tasks:

- user:

name: "{{item.name}}"

group: "{{item.group}}"

password: "{{item.pwd|password\_hash('sha512')}}"

with\_items:

-

name: "nb"

group: "bin"

pwd: "bb"

---

- hosts: web2

remote\_user: root

tasks:

- shell: adduser z3 || true 不提示错误

- shell: adduser li4

ignore\_errors: True 提示但忽略错误

- shell: echo 123456 | passwd --stdin z3

- shell: echo 123456 | passwd --stdin li4

---

- hosts: web

remote\_user: root

tasks:

- name: install the latest version of Apache

yum:

name: httpd

state: latest

- lineinfile:

path: /etc/httpd/conf/httpd.conf

regexp: '^Listen '

insertafter: '^#Listen '

line: 'Listen 8080'

tags: editconf 打上标签，以便命令行单独调用执行

notify: 调用handlers

- reload apache 调用handlers的关键字

- replace: path=/etc/httpd/conf/httpd.conf 与下一句是连接成一句的

regexp='^#ServerName' replace='ServerName localhost'

- copy:

src: index.html

dest: /var/www/html/

owner: apache

group: apache

mode: 0644

handlers: ----最后一步执行，且触发多次也只执行一次

- name: reload apache ----必须添加name

service:

name: httpd

state: restarted

enabled: yes

handlers(条件触发调用:notify)和标识tags

ansible-playbook httpd.yml --list-tags

ansible-playbook httpd.yml --tags editconf

When 与 register（变量注册）

当cpu负载大于0.7时，关闭httpd服务

---

- hosts: cache

remote\_user: root

tasks:

- shell: uptime | awk '{printf("%.2f",$(NF-2))}'

register: result

- service:

name: httpd

state: stopped

when: result.stdout|float > 0.7

With\_items 标准循环

---

- hosts: web

remote\_user: root

tasks:

- user:

name: "{{item.name}}"

group: "{{item.group}}"

password: "{{item.pwd|password\_hash('sha512')}}"

with\_items:

-

name: "nb"

group: "bin"

pwd: "bb"

With\_nested 嵌套循环

---

- hosts: cache

remote\_user: root

vars:

id: [1, 2, 3]

un: ['a', 'b', 'c']

tasks:

- shell: echo "{{item}}"

with\_nested:

- "{{id}}"

- "{{un}}"

一般所需的目录层级有

vars：变量层

tasks：任务层

handlers：触发条件

files：文件

template：模板

default：默认，优先级最低

Include调用文件 和roles调用目录

---

- hosts: web

remote\_user: root

tasks:

- include: tasks/setup.yml

- include: tasks/users.yml user=plj 可以通过{{user}}来传参

handlers:

- include: handlers/handlers.yml

在play中调用”x”的roles

---

- hosts:group

roles:

-x

显示受到影响的主机：ansible-playbook user.yml --list-hosts

显示工作的task：ansible-playbook user.yml --list-tasks

显示将要运行的tag: ansible-playbook user.yml --list-tags