Linux集群搭建

Linux基础配置

修改主机名

```
master:
hostnamectl set-hostname master
bash

slave1
hostnamectl set-hostname slave1
bash

slave2:
hostnamectl set-hostname slave2
bash
```

修改IP地址

```
在master、slave1、slave2上分别进行
vi /etc/sysconfig/network-scripts/ifcfg-ens33
将以下内容进行修改
BOOTPROTO="static"
ONBOOT="yes"
添加以下内容
IPADDR=192.168.10.138
NETMASK=255.255.255.0
GATEWAY=192.168.10.2
DNS1=198.168.10.2
DNS2=114.114.114.114
```

配置hosts域名解析

```
vi /etc/hosts
192.168.10.138 master
192.168.10.139 slave1
192.168.10.140 slave2

scp /etc/hosts slave1:/etc/
scp /etc/hosts slave2:/etc/
```

关闭防火墙

```
systemctl stop firewalld
systemctl disable firewalld
```

重启网络

```
service network restart
```

配置SSH

```
ssh-keygen -t rsa
ssh-copy-id -i master
ssh-copy-id -i slave1
ssh-copy-id -i slave2
```

Java

卸载openjdk

```
rpm -qa |grep openjdk
通过rpm -e --nodeps "查询出来的rpm包" 去卸载
```

安装Java

解压并命名

```
tar -zxvf /opt/software/jdk-8u162-linux-x64.tar.gz -C /opt/module/
mv /opt/module/jdk-8u162-linux-x64 /opt/module/java
```

环境变量

```
 vi /root/.bash_profile
 export JAVA_HOME=/opt/module/java
 export PATH=$PATH:$JAVA_HOME/bin

 source /root/.bash_profile
 java -version

 scp -r /opt/module/java slave1:/opt/module/
 scp -r /opt/module/java slave2:/opt/module/
 scp /root/.bash_profile slave1:/root/
 scp /root/.bash_profile slave2:/root/
```

Hadoop完全分布式

解压并命名

```
tar -zxvf /opt/software/hadoop-2.7.1.tar.gz -C /opt/module
mv /opt/module/hadoop-2.7.1 /opt/module/hadoop
```

环境变量

```
vi /root/.bash_profile
export HADOOP_HOME=/opt/module/hadoop
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin

(3.X版本，如果是root账号必须加以下内容)
export HDFS_NAMENODE_USER=root
export HDFS_DATANODE_USER=root
export HDFS_SECONDARYNAMENODE_USER=root
export YARN_RESOURCEMANAGER_USER=root
export YARN_NODEMANAGER_USER=root

source /root/.bash_profile
```

配置文件

hadoop-env.sh

```
vi /opt/module/hadoop/etc/hadoop/hadoop-env.sh

export JAVA_HOME=/opt/module/java
```

core-site.xml

```
vi /opt/module/hadoop/etc/hadoop/core-site.xml

<property>
  <!--namenode的URL地址(必须写)-->
  <name>fs.defaultFS</name>
  <value>hdfs://master:9000</value>
</property>
<property>
  <!--SequenceFiles中使用的读/写缓冲区的大小，单位为KB,131072KB默认为64M(该配置可选)-->
  <name>io.file.buffer.size</name>
  <value>131072</value>
</property>
<property>
  <!--hadoop临时文件路径(可选配置)-->
  <name>hadoop.tmp.dir</name>
  <value>/opt/module/hadoop/dfs/tmp</value>
</property>
```

hdfs-site.xml

```
vi /opt/module/hadoop/etc/hadoop/hdfs-site.xml

<property>
  <!--hadoop的副本数量，默认为3(必须写)-->
  <name>dfs.replication</name>
  <value>3</value>
</property>
<property>
  <!--在本地文件系统所在的NameNode的存储空间和持续化处理日志(必须写)-->
  <name>dfs.namenode.name.dir</name>
  <value>/opt/module/hadoop/dfs/name</value>
```

```
  </property>
  <property>
    <!--在本地文件系统所在的DataNode的存储空间和持续化处理日志(必须写)-->
    <name>dfs.datanode.data.dir</name>
    <value>/opt/module/hadoop/dfs/data</value>
  </property>
  <property>
    <!--设置namenode线程，处理datanode发出rpc请求数量（可选配置）-->
    <name>dfs.namenode.handler.count</name>
    <value>100</value>
  </property>
```

mapred-site.xml

```
cp /opt/module/hadoop/etc/hadoop/mapred-site.xml.template /opt/module/hadoop/etc/hadoop/mapred-site.xml（3.X版本不用）
vi /opt/module/hadoop/etc/hadoop/mapred-site.xml

<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>
```

yarn-site.xml

```
 vi /opt/module/hadoop/etc/hadoop/yarn-site.xml

 <property>
   <name>yarn.nodemanager.aux-services</name>
   <value>mapreduce_shuffle</value>
 </property>
 <property>
      <name>yarn.resourcemanager.address</name>
      <value>master:8032</value>
 </property>
 <property>
      <name>yarn.resourcemanager.scheduler.address</name>
      <value>master:8030</value>
 </property>
 <property>
      <name>yarn.resourcemanager.resource-tracker.address</name>
      <value>master:8031</value>
 </property>
```

workers

```
(3.X版本配workers)
vi /opt/module/hadoop/etc/hadoop/workers

master
slave1
slave2
```

slaves

```
(2.X版本配slaves)
vi /opt/module/hadoop/etc/hadoop/slaves

master
slave1
slave2
```

分发文件

```
scp -r /opt/module/hadoop slave1:/opt/module/
scp -r /opt/module/hadoop slave2:/opt/module/
scp /root/.bash_profile slave1:/root
scp /root/.bash_profile slave2:/root
```

格式化

```
hdfs namenode -format
```

启动集群

```
start-all.sh
```

Web端

```
2.X版本：master:50070
3.x版本：master:9870
```

MySQL数据库

安装MySQL

解压安装包

```
mkdir mysql_lib
tar -xvf mysql-5.7.28-1.el7.x86_64.rpm-bundle.tar -C mysql_lib/
```

卸载mariadb

```
sudo rpm -qa | grep mariadb | xargs sudo rpm -e --nodeps
```

安装Mysql依赖

```
cd mysql_lib
sudo rpm -ivh mysql-community-common-5.7.28-1.el7.x86_64.rpm
sudo rpm -ivh mysql-community-libs-5.7.28-1.el7.x86_64.rpm
sudo rpm -ivh mysql-community-libs-compat-5.7.28-1.el7.x86_64.rpm
```

安装mysql-client

```
sudo rpm -ivh mysql-community-client-5.7.28-1.el7.x86_64.rpm
```

安装mysql-server

```
 sudo rpm -ivh mysql-community-server-5.7.28-1.el7.x86_64.rpm

 若出现需要依赖
 net-tools 被 mysql-community-server-8.0.18-1.el7.x86_64 需要
 /usr/bin/perl 被 mysql-community-server-8.0.18-1.el7.x86_64 需要
 perl(Getopt::Long) 被 mysql-community-server-8.0.18-1.el7.x86_64 需要
 perl(strict) 被 mysql-community-server-8.0.18-1.el7.x86_64 需要
 则用yum源安装如下依赖包

 yum install -y perl-Module-Install.noarch
 yum install net-tools
```

启动mysql

```
systemctl start mysqld
```

查看mysql密码

```
cat /var/log/mysqld.log | grep password
```

配置MySQL

```
 mysql -uroot -p'password'
 set password=password("Qs23=zs32");
 set global validate_password_policy=0;
 set global validate_password_length=4;
 set password=password("123456");
 use mysql;
 select user, host from user;
 update user set host="%" where user="root";
 flush privileges;
 quit;
```

Hive数仓组件

解压并命名

```
tar -zxvf /opt/software/apache-hive-2.0.0-bin.tar.gz -C /opt/module
mv /opt/module/apache-hive-2.0.0-bin /opt/module/hive
```

环境变量

```
vi /root/.bash_profile
export HIVE_HOME=/opt/module/hive
export PATH=$PATH:$HIVE_HOME/bin

source /root/.bash_profile
```

驱动包

```
cp mysql-connector-java-5.1.27-bin.jar /opt/module/hive/lib/
```

配置文件

```
新建一个hive-site.xml
vi /opt/module/hive/conf/hive-site.xml

<configuration>
<property>
  <!--连接数据库URL(必选参数)-->
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:mysql://master:3306/hive?createDatabaseIfNotExist=true&amp;useSSL=false</value>
</property>
<property>
  <!--连接数据驱动(必选参数)-->
  <name>javax.jdo.option.ConnectionDriverName</name>
  <value>com.mysql.jdbc.Driver</value>
</property>
<property>
  <!--数据库连接用户名(必选参数)-->
  <name>javax.jdo.option.ConnectionUserName</name>
  <value>root</value>
</property>
<property>
  <!--数据库连接密码(必选参数)-->
  <name>javax.jdo.option.ConnectionPassword</name>
  <value>123456</value>
</property>
<property>
  <!--验证元数据的一致性，默认为false(可选参数)-->
  <name>hive.metastore.schema.verification</name>
  <value>false</value>
</property>
<property>
  <!--指定HDFS内hive数据临时文件存放目录,启动hive>，HDFS自动创建(可选参数)-->
  <name>hive.exec.scratchdir</name>
  <value>/hive/warehouse/tmp</value>
</property>
<property>
  <!--指定HDFS内hive数据的存放目录，HDFS自动创建(可选参数)-->
  <name>hive.metastore.warehouse.dir</name>
  <value>/hive/warehouse/home</value>
</property>
<property>
  <!--客户端显示当前数据库名称信息(可选参数)-->
  <name>hive.cli.print.header</name>
```

```
    <value>true</value>
  </property>
  <property>
    <!--客户端显示当前数据库名称(可选参数)-->
    <name>hive.cli.print.current.db</name>
    <value>true</value>
  </property>
  <property>
    <!--支持正则匹配(可选参数)-->
    <name>hive.support.quoted.identifiers</name>
    <value>none</value>
  </property>
  </configuration>
```

初始化

```
schematool -dbType mysql -initSchema
```

日志

```
在hive的conf下新建log4j.properties

log4j.rootLogger=WARN, CA
log4j.appender.CA=org.apache.log4j.ConsoleAppender
log4j.appender.CA.layout=org.apache.log4j.PatternLayout
log4j.appender.CA.layout.ConversionPattern=%-4r [%t] %-5p %c %x - %m%n
```

启动Hive

```
nohup hive --service metastore &
```

服务部署

```
 cd $HADOOP_HOME/etc/hadoop
 vi core-site.xml

 <property>
  <name>hadoop.proxyuser.root.hosts</name>
  <value>*</value>
 </property>
 <property>
  <name>hadoop.proxyuser.root.groups</name>
  <value>*</value>
 </property>
```

```
vi hive-site.xml

<!-- 指定hiveserver2连接的host -->
<property>
    <name>hive.server2.thrift.bind.host</name>
    <value>master</value>
</property>

<!-- 指定hiveserver2连接的端口号 -->
<property>
    <name>hive.server2.thrift.port</name>
    <value>10000</value>
</property>
```

```
hive --service hiveserver2 &
```

Zookeeper集群部署

解压并命名

```
tar -zxvf /opt/software/zookeeper-3.4.8.tar.gz -C /opt/module/
mv /opt/module/zookeeper-3.4.8 /opt/module/zookeeper
```

环境变量

```
vi /root/.bash_profile
export ZOOKEEPER_HOME=/opt/module/zookeeper
export PATH=$PATH:$ZOOKEEPER_HOME/bin

source /root/.bash_profile
```

配置文件

zoo.cfg

```
cp /opt/module/zookeeper/conf/zoo_sample.cfg /opt/module/zookeeper/conf/zoo.cfg
vi /opt/module/zookeeper/conf/zoo.cfg

修改datadir
dataDir=/opt/module/zookeeper/data
增加以下三列
server.1=master:2888:3888
server.2=slave1:2888:3888
server.3=slave2:2888:3888
```

myid

```
mkdir /opt/module/zookeeper/data
echo "1" > /opt/module/zookeeper/data/myid
```

分发文件

```
scp -r /opt/module/zookeeper slave1:/opt/module/
scp -r /opt/module/zookeeper slave2:/opt/module/
scp /root/.bash_profile slave1:/root/
scp /root/.bash_profile slave2:/root/

slave1上：
echo 2 > /opt/module/zookeeper/data/myid
source /root/.bash_profile

slave2上：
echo 3 > /opt/module/zookeeper/data/myid
source /root/.bash_profile
```

启动集群

```
分别在master、slave1、slave2上开集群
zkServer.sh start
```

HBase完全分布式

解压并命名

```
tar -zxvf /opt/software/hbase-1.2.1-bin.tar.gz -C /opt/module/
mv /opt/module/hbase-1.2.1-bin /opt/module/hbase
```

环境变量

```
vi /root/.bash_profile
export HBASE_HOME=/opt/module/hbase
export PATH=$PATH:$HBASE_HOME/bin

source /root/.bash_profile
```

配置文件

hbase-site.xml

```
 vi /opt/module/hbase/conf/hbase-site.xml

<property>
  <!--是否分布式部署（必选）-->
  <name>hbase.cluster.distributed</name>
  <value>true</value>
</property>
<property>
  <!--hbase存放数据目录（必选）-->
  <name>hbase.rootdir</name>
  <value>hdfs://master:9000/hbase</value>
</property>
```

```
  <property>
    <!--zookooper配置、日志等的存储位置（必选）-->
    <name>hbase.zookeeper.property.dataDir</name>
    <value>/opt/module/zookeeper/ZKdata</value>
  </property>
  <property>
    <!--配置zk端口（必选）-->
    <name>hbase.zookeeper.property.clientPort</name>
    <value>2181</value>
  </property>
  <property>
    <!--zookeeper地址（必选）-->
    <name>hbase.zookeeper.quorum</name>
    <value>master,slave1,slave2</value>
  </property>
  <property>
    <!--设置hbase端口（必选）-->
    <name>hbase.master.info.port</name>
    <value>16010</value>
  </property>
```

hbase-env.sh

```
vi /opt/module/hbase/conf/hbase-env.sh

export JAVA_HOME=/opt/module/java
export HBASE_MANAGES_ZK=false
```

regionservers

```
vi /opt/module/hbase/conf/regionservers

master
slave1
slave2
```

backup-master

```
可选
vi /opt/module/hbase/conf/backup-masters
slave1
```

分发

```
scp /root/.bash_profile slave1:/root/
scp /root/.bash_profile slave2:/root/

scp -r /opt/module/hbase slave1:/opt/module/
scp -r /opt/module/hbase slave2:/opt/module/
```

启动Hbase

```
start-hbase.sh
```

Phoenix部署

解压并命名

```
tar -zxvf phoenix-hbase-2.2-5.1.3-bin.tar.gz -C /opt/module
mv /opt/module/phoenix-hbase-2.2-5.1.3-bin /opt/module/phoenix
```

拷贝并分发server包

```
cp /opt/module/phoenix/phoenix-server-hbase-2.2-5.1.3.jar /opt/module/hbase/lib

scp /opt/module/hbase/lib/ root@slave1:/opt/module/hbase/lib/
scp /opt/module/hbase/lib/ root@slave2:/opt/module/hbase/lib/
```

配置环境变量

```
vim /root/.bash_profile
export PHOENIX_HOME=/opt/module/phoenix
export PHOENIX_CLASSPATH=$PHOENIX_HOME
export PATH=$PATH:$PHOENIX_HOME/bin

source /root/.bash_profile
```

连接Phoenix

```
/opt/module/phoenix/bin/sqlline.py master,slave1,slave2:2181
```

Spark集群

Spark完全分布式

解压并命名

```
tar -zxvf /opt/software/spark-2.0.0-bin-hadoop2.7.tgz -C /opt/module/
mv /opt/module/spark-2.0.0-bin-hadoop2.7 /opt/module/spark
```

配置文件

spark-env.sh

```
cp  /opt/module/spark/conf/spark-env.sh.template  /opt/module/spark/conf/spark-env.sh
vi /opt/module/spark/conf/spark-env.sh

# java位置
export JAVA_HOME=/opt/module/java
# master节点IP或域名
export SPARK_MASTER_IP=master
# worker内存大小
export SPARK_WORKER_MEMORY=1G
# Worker的cpu核数
SPARK_WORKER_CORES=1
# hadoop配置文件路径
export HADOOP_CONF_DIR=/opt/module/hadoop/etc/hadoop
```

slaves

```
cp /opt/module/spark/conf/slaves.template /opt/module/spark/conf/slaves
vi /opt/module/spark/conf/slaves

master
slave1
slave2
```

分发文件

```
scp -r /opt/module/spark slave1:/opt/module/
scp -r /opt/module/spark slave2:/opt/module/
```

启动集群

```
/opt/module/spark/sbin/start-all.sh
```

Spark On Yarn

解压并命名

```
tar -zxvf spark-3.0.0-bin-hadoop3.2.tgz -C /opt/module
mv spark-3.0.0-bin-hadoop3.2 spark-yarn
```

修改配置文件

yarn-site.xml

```
vi /opt/module/hadoop/etc/hadoop/yarn-site.xml

<!--是否启动一个线程检查每个任务正使用的物理内存量，如果任务超出分配值，则直接将其杀掉，默认是true -->
<property>
        <name>yarn.nodemanager.pmem-check-enabled</name>
        <value>false</value>
</property>

<!--是否启动一个线程检查每个任务正使用的虚拟内存量，如果任务超出分配值，则直接将其杀掉，默认是true -->
<property>
        <name>yarn.nodemanager.vmem-check-enabled</name>
        <value>false</value>
</property>
```

分发yarn-site.xml！！！！！

spark-env.sh

```
cd /opt/module/spark-yarn/conf/
mv spark-env.sh.template spark-env.sh
vi spark-env.sh

export JAVA_HOME=/opt/module/java
YARN_CONF_DIR=/opt/module/hadoop/etc/hadoop
```

提交应用

```
bin/spark-submit --class org.apache.spark.examples.SparkPi --master yarn --deploy-mode cluster./examples/jars/spark-examples_2.12-3.0.0.jar  10
```

Flink On Yarn

解压并命名

```
tar -zxvf /opt/software/flink-1.13.0-bin-scala_2.12.tgz /opt/module
mv /opt/module/flink-1.13.0-bin-scala_2.12 /opt/module/flink
```

环境变量

```
vi /root/.bash_profile

export HADOOP_CONF_DIR=${HADOOP_HOME}/etc/hadoop
export HADOOP_CLASSPATH=`hadoop classpath`

source /root/.bash_profile
```

配置文件

```
vi /opt/module/flink/conf/flink-conf.yaml

jobmanager.memory.process.size: 1600m
taskmanager.memory.process.size: 1728m
taskmanager.numberOfTaskSlots: 8
parallelism.default: 1
```

Flume日志传输

解压并命名

```
tar -zxvf /opt/software/apache-flume-1.9.0-bin.tar.gz /opt/module
mv /opt/module/apache-flume-1.9.0-bin /opt/module/flume
```

配置文件

```
cp flume-env.sh.template flume-env.sh
vi /opt/module/flume/conf/flume-env.sh

export JAVA_HOME=/opt/module/java
```

分发

```
scp -r /opt/module/flume slave1:/opt/module/
scp -r /opt/module/flume slave2:/opt/module/
```

Kafka消息队列

解压并命名

```
tar -zxvf /opt/software/kafka_2.12-3.0.0.tgz /opt/module
mv /opt/module/kafka_2.12-3.0.0 /opt/module/kafka
```

配置文件

```
vi /opt/module/kafka/config/server.properties

# 需要修改的第一个参数
broker.id=1

# 默认删除topic功能是注释状态 可取消注释
delete.topic.enable=true

# kafka数据存放目录 默认存在tmp下 可以自定义到其他位置
log.dirs=/data/kafka-logs/

# zookeeper的连接地址
zookeeper.connect=master:2181,slave1:2181,slave2:2181
```

分发

```
scp -r /opt/module/kafka slave1:/opt/module/
scp -r /opt/module/kafka slave2:/opt/module/
```

修改配置文件

```
slave1上将broker.id改成2
slave2上将broker.id改成3
```

启动服务

```
/opt/module/kafka/bin/kafka-server-start.sh -daemon /opt/module/kafka/config/server.properties
```

ClickHouse单机部署

安装RPM

```
mv /opt/software/clickhouse /opt/module/
cd /opt/module/clickhouse
rpm -ivh *.rpm
```

配置文件

```
vi /etc/clickhouse-server/config.xml

修改listen_host
```

启动客户端

```
systemctl start clickhouse-server
systemctl status clickhouse-server

clickhouse-client --password 123456
```

Redis单机部署模式

解压并命名

```
tar -zxvf /opt/software/redis-7.0.5.tar.gz /opt/module
mv /opt/module/redis-7.0.5 /opt/module/redis
```

编译并安装

```
yum install gcc
cd /opt/module/redis/
make && make install
```

配置文件

```
vi /opt/module/redis/redis.conf

(修改以下参数)
daemonize yes
appendonly yes
bind master
(增加以下参数)
requirepass 123123
```

启动服务

```
redis-server /opt/module/redis/redis.conf
```

连接客户端

```
redis-cli -h master -p 6379
AUTH 123123

set key1 v1
get key1
```

MaxWell采集工具

部署基础

安装kafka和MySQL。

解压并命名

```
 tar -zxvf /opt/software/maxwell-1.29.2.tar.gz -C /opt/module/
```

MySQL环境准备

1. 修改mysql的配置文件，开启MySQL Binlog设置

```
vi /etc/my.cnf
在[mysqld]模块下添加一下内容
server_id=1
log-bin=master
binlog_format=row
#binlog-do-db=test_maxwell

并重启Mysql服务
systemctl restart mysqld
登录mysql并查看是否修改完成
mysql\> show variables like '%binlog%';
查看下列属性  binlog_format | ROW
```

2. 进入/var/lib/mysql目录，查看MySQL生成的binlog文件

```
cd /var/lib/mysql
master.000001
master.index
```

注：MySQL生成的binlog文件初始大小一定是154字节，然后前缀是log-bin参数配置的，后缀是默认从.000001，然后依次递增。除了binlog文件文件以外，MySQL还会额外生产一个.index索引文件用来记录当前使用的binlog文件。

初始化Maxwell元数据库

```
1.在MySQL中建立一个maxwell库用于存储Maxwell的元数据

mysql\> CREATE DATABASE maxwell;

2.设置mysql用户密码安全级别

mysql\> set global validate_password_length=4;

mysql\> set global validate_password_policy=0;

3.分配一个账号可以操作该数据库

mysql\> GRANT ALL ON maxwell.* TO 'maxwell'@'%' IDENTIFIED BY '123456';

4.分配这个账号可以监控其他数据库的权限

mysql\> GRANT SELECT ,REPLICATION SLAVE , REPLICATION CLIENT ON *.* TO maxwell@'%';

5.刷新mysql表权限

mysql\> flush privileges;
```

Maxwell进程启动

```
bin/maxwell -user='maxwell' --password='123456' --host='master' -producer=stdout
```

Prometheus监控组件

Prometheus

```
tar -zxvf prometheus-2.29.1.linux-amd64.tar.gz -C /opt/module/
mv prometheus-2.29.1.linux-amd64/ prometheus

vim prometheus.yml
将localhost改为master
添加以下内容
- job_name: 'pushgateway'
 static_configs:
 - targets: ['master:9091']
 labels:
 instance: pushgateway


 - job_name: 'node exporter'
 static_configs:
 - targets: ['master:9100', 'master:9100', 'master:9100']
```

Alertmanager

```
tar -zxvf alertmanager-0.23.0.linux-amd64.tar.gz -C /opt/module/
mv ../module/alertmanager-0.23.0.linux-amd64/ ../module/alertmanager
```

pushgateway

```
tar -zxvf pushgateway-1.4.1.linux-amd64.tar.gz -C /opt/module/
mv ../module/pushgateway-1.4.1.linux-amd64/ ../module/pushgateway
```

node_exporter

```
tar -zxvf node_exporter-1.2.2.linux-amd64.tar.gz -C /opt/module/
mv ../module/node_exporter-1.2.2.linux-amd64/ ../module/node_exporter
scp -r node_exporter/ root@slave1:/opt/module/
scp -r node_exporter/ root@slave2:/opt/module/
```

启动

```
nohup ./prometheus --config.file=prometheus.yml > ./prometheus.log 2>&1 &
nohup ./pushgateway --web.listen-address :9091 > ./pushgateway.log 2>&1 &
nohup ./alertmanager --config.file=alertmanager.yml > ./alertmanager.log 2>&1 &
```

Yml文件

```
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
     - static_configs:
          - targets:
             # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"

     # metrics_path defaults to '/metrics'
     # scheme defaults to 'http'.

     static_configs:
        - targets: ['master:9090']

  - job_name: 'pushgateway'
     static_configs:
        - targets: ['master:9091']
          labels:
             instance: pushgateway


  - job_name: 'node exporter'
     static_configs:
          - targets: ['master:9100', 'slave1:9100', 'slave2:9100']
```

Web端

```
192.168.10.138:9090
```

grafana

直接解压即可