重装系统：

dracut： ls |grep sdb

查看端口：blkid

# 一：webOPI可视化

部署DBWR

1. maven版本3.8.4

指定maven的仓库和下载源

<localRepository>/home/training/Downloads/apache-maven-3.8.4/maven</localRepository>

<mirror>

<id>alimaven</id>

<name>aliyun maven</name>

<url>http://maven.aliyun.com/nexus/content/groups/public/</url>

<mirrorOf>central</mirrorOf>

</mirror>

1. Jdk版本15.0.2
2. Tomcat版本8.5.78

---特别注意，需要PVWS和DBWR两个包一起用

# 二：elog服务器搭建

网址地址: <https://elog.psi.ch/elog/>

Make

Make install

1. Elog 服务器指定端口号启动： elogd -p 8088
2. Elog 服务器后台运行： elogd -p 8088 -c /usr/local/elog/elog.cfg（elog.cfg的路径） -D

（3）通过make install 会自动生成elogd.service（/usr/lib/systemd/system）

# Elog 手机端代码

网址：<https://github.com/wanglin86769/elog-mobile>

Nodejs安装

1. yum clean all &&yum makecache fast
2. yum install -y gcc-c++ make
3. Curl -sL <https://rpm.nodesource.com/setup_10.x> | sudo -E bash -
4. sudo yum install nodejs
5. npm install -g n
6. sudo n 14.12.0 （安装指定nodejs的版本）

Nodejs版本为v14.12.0

Npm版本为6.14.8

# Elog 手机端运行

进入到backend 后

npm install

npm run dev

进入到frontend

npm install

npm run serve

注意：

（1）需要进入到frontend/src/Home.vue修改源代码

（2）注意在backend/config/logbook.js配置中需要http而不是https

（3）注意backend/config/logbook.js中的配置最后端口号后没有/

# 三：机时计划

源码为accplan.zip

解压后放在tomcat下的webapps下，修改源代码

# 四：Grafana配置

# 安装教程

<https://grafana.com/docs/grafana/latest/installation/>

（最新版本安装）

<https://grafana.com/grafana/download>（指定版本）

**wget <https://dl.grafana.com/enterprise/release/grafana-enterprise-7.5.15-1.x86_64.rpm>**

**sudo yum install grafana-enterprise-7.5.15-1.x86\_64.rpm**

1. Installs binary to

**/usr/sbin/grafana-server**

(2)Copies init.d script to

**/etc/init.d/grafana-server**

(3)Installs default file (environment vars) to

**/etc/sysconfig/grafana-server**

(4)Copies configuration file to

**/etc/grafana/grafana.ini**

(5)Installs systemd service (if systemd is available) name

**grafana-server.service**

(6)The default configuration uses a log file at

**/var/log/grafana/grafana.log**

1. The default configuration specifies an sqlite3 database at

**/var/lib/grafana/grafana.db**

# Grafana 免密登录

修改/**/etc/grafana/grafana.ini**

**[auth.anonymous]**

**# enable anonymous access**

**enabled = true         #默认false**

**# specify role for unauthenticated users**

**org\_role = Admin    #默认Viewer**

# 五：Grafana配置archiver数据源

下载地址：

<https://github.com/sasaki77/archiverappliance-datasource>

（版本为1.3.2）

配置网页：

<https://sasaki77.github.io/archiverappliance-datasource/configuration.html>

1. 到/var/lib/grafana/下创建plugins文件夹

mkdir plugins

（2）将加压后的包放到/var/lib/grafana/plugins下

（3）配置 **/etc/grafana/grafana.ini**

**【plugins】**

**allow\_loading\_unsigned\_plugins=sasaki77-archiverappliance-datasource**

在grafana里的配置数据源为：

http://localhost:177665/retrieval

# 六：报警

# Grafana邮箱报警

**【smtp】**

修改/**/etc/grafana/grafana.ini**

**enabled = true**

**host = localhost:25**

**from\_address = admin@grafana.localhost**

**from\_name = Grafana**

# Grafana 微信报警

Prometheus、alertmanager、node\_exporter下载地址：

服务位置：/usr/lib/systemd/system/

<https://prometheus.io/download/>

**Prometheus服务配置（system管理，端口9090）：**

[Unit]

Description=prometheus

After=network.target

[Service]

Type=simple

ExecStart=文件位置 --config.file=文件所在位置

Restart=on-failure

[Install]

WantedBy=multi-user.target

**Alertmanager服务配置（system管理，端口9093）：**

[Unit]

Description=alertmanager

After=network.target

[Service]

Type=simple

ExecStart=文件位置 --config.file=文件所在位置

Restart=on-failure

[Install]

WantedBy=multi-user.target

**Node\_exporter服务配置（system管理，端口9100）：**

[Unit]

Description=node\_exporter

After=network.target

[Service]

Type=simple

ExecStart=文件位置

Restart=on-failure

[Install]

WantedBy=multi-user.target

**Mysql\_exporter服务配置（system管理，端口9104）**

安装mysql ，启动mysql（systemctl start mysqld）

需要在Mysql\_exporter文件中创建my.cnf文件

在文件中添加（mysql的用户密码）：

[client

]

user：root

password：admin

[Unit]

Description=mysql\_exporter

After=network.target

[Service]

Type=simple

ExecStart=文件位置 --config.my-cnf=文件所在位置

Restart=on-failure

[Install]

WantedBy=multi-user.target

**Prometheus微信报警（system管理，端口9090）：**

**一：首先配置prometheus.yml中的**

（1）alerting:

（2）rule\_files:

- "rules.yml文件位置"（需要自己创建一个文件）

1. scrape\_configs:（添加需要监测的主机）

rules.yml内容格式

groups：

- name: Grafana

rules:

- alert: server\_status

expr: up{job="Grafana"} == 0

for: 15s

labels:

severity: warning

annotations:

summary: "机器{{ $labels.instance }} 挂了"

description: "报告.请立即查看!"

- name: prometheus

rules:

- alert: server\_status

expr: up{job="prometheus"} == 0

for: 15s

labels:

severity: warning

annotations:

summary: "机器{{ $labels.instance }} 挂了"

description: "报告.请立即查看!"

- name: node\_exporter

rules:

- alert: server\_status

expr: up{job="node\_exporter"} == 0

for: 15s

labels:

severity: warning

annotations:

summary: "机器{{ $labels.instance }} 挂了"

description: "报告.请立即查看!"

- name: mysqld\_exporter

rules:

- alert: server\_status

expr: up{job="mysqld\_exporter"} == 0

for: 15s

labels:

severity: warning

annotations:

summary: "机器{{ $labels.instance }} 挂了"

description: "报告.请立即查看!"

**二：配置alertmanager.yml**

在alertmanager.yml中添加属性

alertmanager.yml内容格式

global:

resolve\_timeout: 5s

templates:

- '/home/HEPS/wechat/alertmanager/alertmanager-0.24.0.linux-amd64/wechat.tmpl'

route:

group\_by: ['alertname']

group\_wait: 30s

# group\_interval: 10s

repeat\_interval: 1m

# receiver: 'web.hook'

receiver: 'wechat'

#receivers:

# - name: 'web.hook'

# webhook\_configs:

# - url: 'http://127.0.0.1:5001/'

receivers:

- name: 'wechat'

wechat\_configs:

- send\_resolved: true

agent\_id: '1000003'

corp\_id: 'wwa5507ee0884aaac4'

to\_user: '@all'

api\_secret: 'K\_06Go60VHblXhOZ5eCFbHaZLGdNtu29\_1103KKVizc'

inhibit\_rules:

- source\_match:

severity: 'critical'

target\_match:

severity: 'warning'

equal: ['alertname', 'dev', 'instance']

templates:

- ‘xxx.tmpl文件的位置’(xxx.tmpl需要自己创建)

xxx.tmpl内容格式：

{{ define "wechat.default.message" }}

{{- if gt (len .Alerts.Firing) 0 -}}

{{- range $index, $alert := .Alerts -}}

{{- if eq $index 0 }}

=== 监控报警 ===

告警状态：{{ .Status }}

告警级别：{{ .Labels.severity }}

告警类型：{{ $alert.Labels.alertname }}

故障主机: {{ $alert.Labels.instance }} {{ $alert.Labels.pod }}

告警主题: {{ $alert.Annotations.summary }}

告警详情: {{ $alert.Annotations.message }}{{ $alert.Annotations.description}};

触发阀值：{{ .Annotations.value }}

故障时间: {{ ($alert.StartsAt.Add 28800e9).Format "2006-01-02 15:04:05" }}

=== end ===

{{- end }}

{{- end }}

{{- end }}

{{- if gt (len .Alerts.Resolved) 0 -}}

{{- range $index, $alert := .Alerts -}}

{{- if eq $index 0 }}

=== 异常恢复 ===

告警类型：{{ .Labels.alertname }}

告警状态：{{ .Status }}

告警主题: {{ $alert.Annotations.summary }}

告警详情: {{ $alert.Annotations.message }}{{ $alert.Annotations.description}};

故障时间: {{ ($alert.StartsAt.Add 28800e9).Format "2006-01-02 15:04:05" }}

恢复时间: {{ ($alert.EndsAt.Add 28800e9).Format "2006-01-02 15:04:05" }}

{{- if gt (len $alert.Labels.instance) 0 }}

实例信息: {{ $alert.Labels.instance }}

{{- end }}

=== end ===

{{- end }}

{{- end }}

{{- end }}

{{- end }}

group\_wait: 10s # 初次发送告警延时

（默认为5m）group\_interval: 1m# 组内重复检查的时间

repeat\_interval: 5m # 告警重发时间，若修复，则不再提醒。

<https://www.dianbanjiu.com/post/alertmanager-%E4%B8%AD%E4%B8%89%E4%B8%AA%E6%97%B6%E9%97%B4%E5%8F%82%E6%95%B0%E4%B8%8A%E7%9A%84%E4%B8%80%E4%BA%9B%E5%9D%91/>

Alertmanager 中有三个关于告警时间的参数：

group\_wait：alertmanager 在接收到一条新的告警（第一次出现的告警）时，将这条告警发送给 receiver 之前需要等待的时间

group\_interval：对于一条已经出现过的告警，alertmanager 检查会每隔 group\_interval 时间检查一次告警

repeat\_interval： 对于一条已经出现过的告警，每隔 repeat\_interval 会重新发送给 receiver

这三个参数的基础含义比较简单，但是在最近的使用中，我发现当我按照如下格式设置这三个参数时：

1group\_wait: 15m

2group\_interval: 5m

3repeat\_interval: 15m

我将会每隔 20m 收到一次告警，如果我将 repeat\_interval 设置为 20m，那我将会每 25m 收到一次告警…

这种现象好像跟这三个参数在 alertmanager 文档中的说明不太一样啊。在搜寻无果之后，我在 alertmanager 的 github 仓库上提了一个 issue，官方给出了一个比较合理的解释。

点此查看此 issue

我根据 Issue 的回答简单总结了一下这三个参数的含义：

Alertmanager 在收到一条新的告警之后，会等待 group\_wait 时间，对这条新的告警做一些分组、更新、静默的操作。当第一条告警经过 group\_wait 时间之后，Alertmanager 会每隔 group\_interval 时间检查一次这条告警，判断是否需要对这条告警进行一些操作，当 Alertmanager 经过 n 次 group\_interval 的检查后，n\*group\_interval 恰好大于 repeat\_interval 的时候，Alertmanager 才会将这条告警再次发送给对应的 receiver。

结合我上面的例子，第一条告警在发送给 alertmanager 15m 后会第一次发送给 receiver，接着 Alertmanager 会每隔 5m 检查一次这条告警，在第 4 次检查的时候，到上次告警发出的时间刚好经过了 5\*4=20>15，所以第二次告警会在第一条告警发出后的 20m 后再次发出，此后每条重复告警都会每隔 20m 发送一次。

这也就意味着在上面的例子中，当你的 15m<repeat\_interval<20m 时，任意两条重复告警的间隔都是 20m。

此外还有一个问题：在上面的例子中，如果 alertmanager 是每隔 5m 检查一次告警，那为什么不是恰好在 15m 的时候发出这条告警？关于这个问题，在 Issue 中尚未给出比较明确的答复。

根据目前的情况，这个问题有两个暂时解决方案：

调整 repeat\_interval 的值：同样结合上面的例子，如果你想每隔 20m 收到一次重复告警，那就把 repeat\_interval 设置为 [15m,20m) 之间的任意一个时间

调整 group\_interval 的值：既然 alertmanager 会每隔 group\_interval 检查一次已有的告警，那就尽可能将它的时间调的低一些，比如将其设置为 1m，这样你最多在你设置的 repeat\_interval+1m 左右就可以收到重复告警

# Grafana webhook报警

Webhook旧版 ：<https://gitee.com/yoyofx/grafana-alert-web-hook>

Webhook新版： <https://gitee.com/yoyofx/yoyogo/tree/master/examples/grafanaalertwebhook>

Go下载地址：

<https://golang.google.cn/dl/>

Go 环境变量配置：

vim .bashrc

export GO\_HOME=GO安装包位置

export PATH=$GO\_HOME/bin:$PATH

go env 看是否安装成功

go env -w GO111MODULE=on

go env -w GOPROXY=https://goproxy.cn,direct

go build -o xxx（创建可执行文件）

go build .

go run .



tag前一个对应上图中的alert，后一个对应上图中的jishi。

# Screenshot from 2022-04-22 11-05-18

在Grafana里配置数据源为：http://localhost:8089/alert

# 七：Archiver appliance安装

**Archiver appliance安装说明地址**

<http://slacmshankar.github.io/epicsarchiver_docs/index.html>

**安装包archappl\_vx.x.x\_XXXXXXX.tar.gz地址**

**<https://github.com/slacmshankar/epicsarchiverap/releases>**

**版本为JDK12/Tomcat 9 release（Archappl包 版本为2019T15-45-42）**

1. **安装jdk 15.0.2**
2. **下载tomcat包 8.5.78**
3. **启动方式为： ./quickstart.sh apache-tomcat-8.5.78.tar.gz**