**Prometheus简明教程**

**说明**

* 本示例中的系统环境为Ubuntu Server 20.04.5 X86\_64
* config-examples目录下保存有各类配置示例

**Server端组件**

以下组件可以部署在不同的节点之上，彼此间能通过网络互通即可。

**部署Prometheus Server**

下载程序包，以2.40.2版为例：

curl -LO https://github.com/prometheus/prometheus/releases/download/v2.40.2/prometheus-2.40.2.linux-amd64.tar.gz

展开程序包：

tar xf prometheus-2.40.2.linux-amd64.tar.gz -C /usr/local/

ln -sv /usr/local/prometheus-2.40.2.linux-amd64 /usr/local/prometheus

创建用户，并设定目录权限：

useradd -r prometheus

mkdir /usr/local/prometheus/data

chown -R prometheus.prometheus /usr/local/prometheus/data

创建Systemd Unitfile，保存于/usr/lib/systemd/system/prometheus.service文件中:

[Unit]

Description=Monitoring system and time series database

Documentation=https://prometheus.io/docs/introduction/overview/

[Service]

Restart=always

User=prometheus

EnvironmentFile=-/etc/default/prometheus

ExecStart=/usr/local/prometheus/prometheus \

--config.file=/usr/local/prometheus/prometheus.yml \

--storage.tsdb.path=/usr/local/prometheus/data \

--web.console.libraries=/usr/share/prometheus/console\_libraries \

--web.enable-lifecycle \

$ARGS

ExecReload=/bin/kill -HUP $MAINPID

TimeoutStopSec=20s

SendSIGKILL=no

LimitNOFILE=8192

[Install]

WantedBy=multi-user.target

如有必要，可创建环境配置文件/etc/default/prometheus，通过变量ARGS为prometheus指定启动参数

启动服务：

systemctl daemon-reload

systemctl start prometheus.service

systemctl enable prometheus.service

验证监听的端口，并测试访问其暴露的指标

ss -tnlp | grep '9090'

curl localhost:9090/metrics

修改配置后的重载命令：

curl -XPOST http://localhost:9090/-/reload

随后即可访问Prometheus的Web UI，其使用的URL如下，其中的<HOST\_IP>要替换为节点的实际地址: http://<HOST\_IP>:9090/

**部署Consul**

组件功能：用于为Prometheus提供基于Consul进行服务发现的测试环境。

**部署**

下载Consul，以1.14.1版本为例：

curl -LO https://releases.hashicorp.com/consul/1.14.1/consul\_1.14.1\_linux\_amd64.zip

展开程序包：

mkdir -p /usr/local/consul/config/data

unzip consul\_1.14.1\_linux\_amd64.zip -d /usr/local/consul

创建用户，若consul用户已经存在，可略过该步骤：

useradd -r consul

chown consul.consul /usr/local/consul/{data,config}

创建Systemd Unitfile，保存于/usr/lib/systemd/system/consul.service文件中:

[Unit]

Description="HashiCorp Consul - A service mesh solution"

Documentation=https://www.consul.io/

Requires=network-online.target

After=network-online.target

[Service]

EnvironmentFile=-/etc/consul.d/consul.env

User=consul

Group=consul

ExecStart=/usr/bin/consul agent -dev -bootstrap \

-config-dir /usr/local/consul/config \

-data-dir /usr/local/consul/data \

-ui \

-log-level INFO \

-bind 127.0.0.1 \

-client 0.0.0.0

ExecReload=/bin/kill --signal HUP $MAINPID

KillMode=process

KillSignal=SIGTERM

Restart=on-failure

LimitNOFILE=65536

[Install]

WantedBy=multi-user.target

启动服务：

systemctl daemon-reload

systemctl start consul.service

systemctl enable consul.service

验证监听的端口

ss -tnlp | grep '8500'

随后即可访问Consul的Web UI，其使用的URL如下，其中的<HOST\_IP>要替换为节点的实际地址: http://<HOST\_IP>:8500/

**直接请求API进行服务注册**

相关的[文档](https://developer.hashicorp.com/consul/api-docs/agent/service#register-service)

列出已经注册的服务：

curl -XGET http://localhost:8500/v1/agent/services

获取某个特定服务的配置信息：

curl -XGET http://localhost:8500/v1/agent/service/<SERVICE\_ID>

注册一个服务到Consul上，请求报文的body必须遵循json语法规范，且要符合Consul Service的API要求：

curl -XPUT --data @/path/to/payload\_file.json http://localhost:8500/v1/agent/service/register

例如，下面定义了一个要注册的tomcat服务示例，它保存于tomcat.json文件中

{

"id": "tomcat",

"name": "tomcat",

"address": "tomcat",

"port": 8080,

"tags": ["tomcat"],

"checks": [{

"http": "http://tomcat:8080/metrics",

"interval": "5s"

}]

}

我们可以使用类似如下命令完成服务注册。

curl -XPUT --data @tomcat.json http://localhost:8500/v1/agent/service/register

注销某个服务：

curl -XPUT http://localhost:8500/v1/agent/service/deregister/<SERVICE\_ID>

**使用register命令注册服务**

consul services register命令也可用于进行服务注册，只是其使用的配置格式与直接请求HTTP API有所不同。

consul services register /path/to/pyload\_file.json

注册单个服务时，使用service进行定义，注册多个服务时，使用services以列表格式进行定义。下面的示例定义了单个要注册的服务。

{

"service": {

"id": "tomcat",

"name": "tomcat",

"address": "tomcat",

"port": 8080,

"tags": ["tomcat"],

"checks": [{

"http": "http://tomcat:8080/metrics",

"interval": "5s"

}]

}

}

下面的示例，以多个的服务的格式给出了定义。

{

"services": [{

"id": "tomcat",

"name": "tomcat",

"address": "tomcat",

"port": 8080,

"tags": ["tomcat"],

"checks": [{

"http": "http://tomcat:8080/metrics",

"interval": "5s"

}]

}

]

}

注销服务，也可以使用consul services deregister命令进行。

consul services deregister -id <SERVICE\_ID>

**部署Grafana**

组件功能：用于为Prometheus提供完善的可视化界面。

Ubuntu/Debian系统上的部署步骤

apt-get install -y adduser libfontconfig1

VERSION=9.2.5

curl -LO https://dl.grafana.com/oss/release/grafana\_${VERSION}\_amd64.deb

dpkg -i grafana\_${VERSION}\_amd64.deb

启动服务：

systemctl daemon-reload

systemctl start grafana-server.service

systemctl enable grafana-server.service

验证监听的端口，并测试访问其暴露的指标

ss -tnlp | grep '3000'

curl localhost:3000/metrics

随后即可访问Grafana Server的Web UI，其使用的URL如下，其中的<HOST\_IP>要替换为节点的实际地址: http://<HOST\_IP>:3000/

**部署AlertManager**

组件功能：用于为Proemtheus提供发送告警的通知的系统组件；

下载程序包，以0.24.0版为例：

curl -LO https://github.com/prometheus/alertmanager/releases/download/v0.24.0/alertmanager-0.24.0.linux-amd64.tar.gz

展开程序包：

tar xf alertmanager-0.24.0.linux-amd64.tar.gz -C /usr/local/

ln -sv /usr/local/alertmanager-0.24.0.linux-amd64 /usr/local/alertmanager

创建用户，若prometheus用户已经存在，可略过该步骤：

useradd -r prometheus

mkdir /usr/local/alertmanager/data

chown -R prometheus.prometheus /usr/local/alertmanager/data

创建Systemd Unitfile，保存于/usr/lib/systemd/system/alertmanager.service文件中:

[Unit]

Description=alertmanager

Documentation=https://prometheus.io/docs/introduction/overview/

After=network.target

[Service]

Type=simple

User=prometheus

ExecStart=/usr/local/alertmanager/alertmanager \

--config.file="/usr/local/alertmanager/alertmanager.yml" \

--storage.path="/usr/local/alertmanager/data/" \

--data.retention=120h \

--log.level=info

ExecReload=/bin/kill -HUP $MAINPID

TimeoutStopSec=20s

Restart=always

[Install]

WantedBy=multi-user.target

启动服务：

systemctl daemon-reload

systemctl start alertmanager.service

systemctl enable alertmanager.service

验证监听的端口，并测试访问其暴露的指标

ss -tnlp | grep '9093'

curl localhost:9093/metrics

修改配置后的重载命令：

curl -XPOST http://localhost:9093/-/reload

随后即可访问AlertManager的Web UI，其使用的URL如下，其中的<HOST\_IP>要替换为节点的实际地址: http://<HOST\_IP>:9093/

**Exporters**

**部署node-exporter**

提示：每个主机节点上均应该部署node-exporter；

下载程序包，以1.4.0版本为例：

curl -LO https://github.com/prometheus/node\_exporter/releases/download/v1.4.0/node\_exporter-1.4.0.linux-amd64.tar.gz

展开程序包：

tar xf node\_exporter-1.4.0.linux-amd64.tar.gz -C /usr/local/

ln -sv /usr/local/node\_exporter-1.4.0.linux-amd64 /usr/local/node\_exporter

创建用户，若prometheus用户已经存在，可略过该步骤：

useradd -r prometheus

创建Systemd Unitfile，保存于/usr/lib/systemd/system/node\_exporter.service文件中:

[Unit]

Description=node\_exporter

Documentation=https://prometheus.io/docs/introduction/overview/

After=network.target

[Service]

Type=simple

User=prometheus

ExecStart=/usr/local/node\_exporter/node\_exporter \

--collector.ntp \

--collector.mountstats \

--collector.systemd \

--collector.ethtool \

--collector.tcpstat

ExecReload=/bin/kill -HUP $MAINPID

TimeoutStopSec=20s

Restart=always

[Install]

WantedBy=multi-user.target

启动服务：

systemctl daemon-reload

systemctl start node\_exporter.service

systemctl enable node\_exporter.service

验证监听的端口，并测试访问其暴露的指标

ss -tnlp | grep '9100'

curl localhost:9100/metrics

**部署Consul Exporter**

提示：仅需要为每个Consul实例部署consul-exporter，它负责将Consul的状态信息转为Prometheus兼容的指标格式并予以暴露。

下载程序包，以0.8.0版本为例：

curl -LO https://github.com/prometheus/consul\_exporter/releases/download/v0.8.0/consul\_exporter-0.8.0.linux-amd64.tar.gz

展开程序包：

tar xf consul\_exporter-0.8.0.linux-amd64.tar.gz -C /usr/local/

ln -sv /usr/local/consul\_exporter-0.8.0.linux-amd64 /usr/local/consul\_exporter

创建用户，若consul用户已经存在，可略过该步骤：

useradd -r consul

创建Systemd Unitfile，保存于/usr/lib/systemd/system/consul\_exporter.service文件中:

[Unit]

Description=consul\_exporter

Documentation=https://prometheus.io/docs/introduction/overview/

After=network.target

[Service]

Type=simple

User=consul

EnvironmentFile=-/etc/default/consul\_exporter

# 具体使用时，若consul\_exporter与consul server不在同一主机时，consul server要指向实际的地址；

ExecStart=/usr/local/consul\_exporter/consul\_exporter \

--consul.server="http://localhost:8500" \

--web.listen-address=":9107" \

--web.telemetry-path="/metrics" \

--log.level=info \

$ARGS

ExecReload=/bin/kill -HUP $MAINPID

TimeoutStopSec=20s

Restart=always

[Install]

WantedBy=multi-user.target

启动服务：

systemctl daemon-reload

systemctl start consul\_exporter.service

systemctl enable consul\_exporter.service

验证监听的端口，并测试访问其暴露的指标

ss -tnlp | grep '9107'

curl localhost:9107/metrics

**部署MySQL Exporter**

提示：仅需要为每个MySQL Server实例部署mysql-exporter，它负责将MySQL Server的状态信息转为Prometheus兼容的指标格式并予以暴露。

下载程序包，以0.14.0版本为例：

curl -LO https://github.com/prometheus/mysqld\_exporter/releases/download/v0.14.0/mysqld\_exporter-0.14.0.linux-amd64.tar.gz

展开程序包：

tar xf mysqld\_exporter-0.14.0.linux-amd64.tar.gz -C /usr/local/

ln -sv /usr/local/mysqld\_exporter-0.14.0.linux-amd64 /usr/local/mysqld\_exporter

创建用户，或mysql用户已经存在，可略过该步骤：

useradd -r mysql

创建Systemd Unitfile，保存于/usr/lib/systemd/system/mysqld\_exporter.service文件中:

[Unit]

Description=consul\_exporter

Documentation=https://prometheus.io/docs/introduction/overview/

After=network.target

[Service]

Type=simple

User=mysql

EnvironmentFile=-/etc/default/mysqld\_exporter

# 具体使用时，若mysql\_exporter与mysql server不在同一主机时，mysql server要指向实际的地址；

# mysql\_exporter连接mysql server使用的用户名和密码均为exporter，该用户要获得正确的授权；

Environment='DATA\_SOURCE\_NAME=exporter:exporter@(localhost:3306)'

ExecStart=/usr/local/mysqld\_exporter/mysqld\_exporter \

--web.listen-address=":9104" \

--web.telemetry-path="/metrics" \

--config.my-cnf=/etc/mysql/my.cnf \ # 将mysql的my.cnf路径配置到此处

--collect.info\_schema.innodb\_tablespaces \

--collect.info\_schema.innodb\_metrics \

--collect.global\_status \

--collect.global\_variables \

--collect.slave\_status \

--collect.engine\_innodb\_status \

$ARGS

ExecReload=/bin/kill -HUP $MAINPID

TimeoutStopSec=20s

Restart=always

[Install]

WantedBy=multi-user.target

在mysqld server上添加用户，并授权其能够加载mysql的信息并转换为指标输出。需要注意的是用户账号授权时使用的主机范围。

mysql> CREATE USER 'exporter'@'localhost' IDENTIFIED BY 'exporter';

mysql> GRANT PROCESS, REPLICATION CLIENT ON \*.\* TO 'exporter'@'localhost';

mysql> GRANT SELECT ON performance\_schema.\* TO 'exporter'@'localhost';

mysql> FLUSH PRIVILEGES;

vim /etc/mysql/my.cnf

[client]

user=exporter

password=exporter

重启mysql服务

systemctl restart mysqld.service

或者

systemctl restart mysql.service

启动服务：

systemctl daemon-reload

systemctl start mysqld\_exporter.service

systemctl enable mysqld\_exporter.service

验证监听的端口，并测试访问其暴露的指标

ss -tnlp | grep '9104'

curl localhost:9104/metrics

**部署Blackbox Exporter**

提示：仅需要部署的Blackbox Exporter实例数据，取决于黑盒监控的任务量及节点的可用资源。

**部署**

下载程序包，以0.22.0版本为例：

curl -LO https://github.com/prometheus/blackbox\_exporter/releases/download/v0.22.0/blackbox\_exporter-0.22.0.linux-amd64.tar.gz

展开程序包：

tar xf blackbox\_exporter-0.22.0.linux-amd64.tar.gz -C /usr/local/

ln -sv /usr/local/blackbox\_exporter-0.22.0.linux-amd64 /usr/local/blackbox\_exporter

创建用户，或prometheus用户已经存在，可略过该步骤：

useradd -r prometheus

创建Systemd Unitfile，保存于/usr/lib/systemd/system/blackbox\_exporter.service文件中:

[Unit]

Description=blackbox\_exporter

Documentation=https://prometheus.io/docs/introduction/overview/

After=network.target

[Service]

Type=simple

User=prometheus

EnvironmentFile=-/etc/default/blackbox\_exporter

ExecStart=/usr/local/blackbox\_exporter/blackbox\_exporter \

--web.listen-address=":9115" \

--config.file="/usr/local/blackbox\_exporter/blackbox.yml"

ExecReload=/bin/kill -HUP $MAINPID

TimeoutStopSec=20s

Restart=always

[Install]

WantedBy=multi-user.target

启动服务：

systemctl daemon-reload

systemctl start blackbox\_exporter.service

systemctl status blackbox\_exporter.service

systemctl enable blackbox\_exporter.service

验证监听的端口，并测试访问其暴露的指标

ss -tnlp | grep '9115'

curl localhost:9115/metrics

随后即可访问Blackbox Exporter的Web UI，其使用的URL如下，其中的<HOST\_IP>要替换为节点的实际地址: http://<HOST\_IP>:9115/

**配置Prometheus使用黑盒监控**

编辑prometheus的主配置文件prometheus.yml，添加类似如下内容，即可用户对目标站点的探测。

# Blackbox Exporter

- job\_name: 'blackbox'

metrics\_path: /probe

params:

module: [http\_2xx] # Look for a HTTP 200 response.

static\_configs:

- targets:

- "www.bfhledu.com"

- "www.baidu.com"

#refresh\_interval: 2m

relabel\_configs:

- source\_labels: [\_\_address\_\_]

target\_label: \_\_param\_target

- source\_labels: [\_\_param\_target]

target\_label: instance

- target\_label: \_\_address\_\_

replacement: "ubuntu22-server01:9115" # 指向实际的Blackbox exporter.

- target\_label: region

replacement: "local"

配置完成后，让Prometheus重载配置。

curl -XPOST http://<PROMETHEUS\_SERVER\_IP>:9090/-/reload

随后即可访问Blackbox Exporter的Web UI验证探测结果，其使用的URL如下，其中的<HOST\_IP>要替换为节点的实际地址: http://<HOST\_IP>:9115/

**部署Prometheus Webhook Dingtalk**

功能说明：Generating [DingTalk](https://www.dingtalk.com/) notification from [Prometheus](https://prometheus.io/) [AlertManager](https://github.com/prometheus/alertmanager) WebHooks.

项目地址：<https://github.com/timonwong/prometheus-webhook-dingtalk>

**部署**

下载程序包，以2.1.0版本为例：

curl -LO https://github.com/timonwong/prometheus-webhook-dingtalk/releases/download/v2.1.0/prometheus-webhook-dingtalk-2.1.0.linux-amd64.tar.gz

展开程序包：

tar xf prometheus-webhook-dingtalk-2.1.0.linux-amd64.tar.gz -C /usr/local/

ln -sv /usr/local/prometheus-webhook-dingtalk-2.1.0.linux-amd64 /usr/local/prometheus-webhook-dingtalk

创建用户，若prometheus用户已经存在，可略过该步骤：

useradd -r prometheus

创建配置文件/usr/local/prometheus-webhook-dingtalk/config.yml，其内容类似如下所示。

## Request timeout

# timeout: 5s

## Customizable templates path

# templates:

# - contrib/templates/legacy/template.tmpl

## You can also override default template using `default\_message`

## The following example to use the 'legacy' template from v0.3.0

#default\_message:

# title: '{{ template "legacy.title" . }}'

# text: '{{ template "legacy.content" . }}'

## Targets, previously was known as "profiles"

targets:

webhook1:

# 要修改为实际的钉钉群上机器人的Webhook地址

url: https://oapi.dingtalk.com/robot/send?access\_token=371988d4248ee5fda293215eafe0e3c...

# secret for signature，要修改为实际的webhook上的加签信息

secret: SEC6e73901c63df8262e81bdc284f7c03874238407bcfa7db62247317...

创建Systemd Unitfile，保存于/usr/lib/systemd/system/prometheus-webhook-dingtalk.service文件中:

[Unit]

Description=prometheus-webhook-dingtalk

Documentation=https://github.com/timonwong/prometheus-webhook-dingtalk/tree/main/docs

After=network.target

[Service]

Type=simple

User=prometheus

EnvironmentFile=-/etc/default/prometheus-webhook-dingtalk

ExecStart=/usr/local/prometheus-webhook-dingtalk/prometheus-webhook-dingtalk \

--web.listen-address=":8060" \

--config.file=/usr/local/prometheus-webhook-dingtalk/config.yml \

--web.enable-ui \

--web.enable-lifecycle \

--log.level=info \

$ARGS

ExecReload=/bin/kill -HUP $MAINPID

TimeoutStopSec=20s

Restart=always

[Install]

WantedBy=multi-user.target

启动服务：

systemctl daemon-reload

systemctl start prometheus-webhook-dingtalk.service

systemctl enable prometheus-webhook-dingtalk.service

验证监听的端口：

ss -tnlp | grep '8060'

随后即可访问Prometheus-Webhook-Dingtalk内置的Web UI接口，其使用的URL如下，其中的<HOST\_IP>要替换为节点的实际地址: http://<HOST\_IP>:8060/ui/

**配置AlertManager通过钉钉进行告警**

编辑alertmanager的配置文 件alertmanager.yml，添加如下receiver，并需要路由中调用它。

- name: 'team-devops-dingtalk'

webhook\_configs:

# 注意配置调用的地址和webhook的名称与前面配置中启用的targets保持一致

- url: http://prometheus-webhook-dingtalk.magedu.com:8060/dingtalk/webhook1/send

send\_resolved: true

