

目录

| | |
|------------------------------|-----------|
| Introduction | 1.1 |
| 作者 | 1.2 |
| 版本变更历史 | 1.3 |
| 第一章 天降奇兵 | 1.4 |
| Prometheus简介 | 1.4.1 |
| Prometheus是什么 | 1.4.1.1 |
| 前世今生 | 1.4.1.2 |
| 适用场景 | 1.4.1.3 |
| 百里挑一 | 1.4.1.4 |
| Prometheus Vs Graphite | 1.4.1.4.1 |
| Prometheus Vs InfluxDB | 1.4.1.4.2 |
| Prometheus Vs OpenTSDB | 1.4.1.4.3 |
| Prometheus Vs Nagios | 1.4.1.4.4 |
| Prometheus的核心组件 | 1.4.2 |
| Prometheus Server | 1.4.2.1 |
| Exporters | 1.4.2.2 |
| AlertManager | 1.4.2.3 |
| PushGateway | 1.4.2.4 |
| Pull Vs Push | 1.4.2.4.1 |
| 在Linux环境下安装Prometheus | 1.4.3 |
| 安装Prometheus Server | 1.4.3.1 |
| 安装NodeExporter采集主机信息 | 1.4.3.2 |
| 启用Basic Auth认证 | 1.4.3.3 |
| 在Docker环境下安装Prometheus | 1.4.4 |
| 使用Docker容器启动Prometheus | 1.4.4.1 |
| 使用Docker Compose启动Prometheus | 1.4.4.2 |
| 小结 | 1.4.5 |
| 第二章 理解Prometheus模型 | 1.5 |
| 什么是Metrics和Labels | 1.5.1 |
| 数据存储模型 | 1.5.1.1 |
| 新的存储层 | 1.5.1.2 |
| 最佳实践 | 1.5.1.3 |
| Prometheus Query Language | 1.5.2 |
| 基础 | 1.5.2.1 |
| 过滤 | 1.5.2.2 |
| 聚合 | 1.5.2.3 |
| 使用 | 1.5.2.4 |

| | |
|---------------------------------|-----------|
| Metrics类型以及使用场景 | 1.5.3 |
| Counter | 1.5.3.1 |
| Gauges | 1.5.3.2 |
| Histograms | 1.5.3.3 |
| Summaries | 1.5.3.4 |
| 小结 | 1.5.4 |
| 第三章 Prometheus告警处理 | 1.6 |
| AlertManager简介 | 1.6.1 |
| 部署AlertManager | 1.6.2 |
| 使用二进制包部署AlertManager | 1.6.2.1 |
| 使用容器部署AlertManager | 1.6.2.2 |
| 自定义Prometheus告警规则 | 1.6.3 |
| 基于Label的动态告警处理 | 1.6.4 |
| 通知对象Receivers | 1.6.4.1 |
| 告警路由规则Route | 1.6.4.2 |
| 抑制机制 inhibit | 1.6.5 |
| 告警模板 | 1.6.6 |
| 第三方集成 | 1.6.7 |
| 与邮件系统集成 | 1.6.7.1 |
| 自定义邮件模板 | 1.6.7.1.1 |
| 与Slack集成 | 1.6.7.2 |
| 与Webhook集成 | 1.6.7.3 |
| 示例：基于Webhook创建自定义扩展 | 1.6.7.3.1 |
| 小结 | 1.6.8 |
| 第四章 可视化一切 | 1.7 |
| Grafana简介 | 1.7.1 |
| 安装Grafana | 1.7.2 |
| 使用二进制包安装Grafana | 1.7.2.1 |
| 使用容器安装Grafana | 1.7.2.2 |
| 使用Prometheus数据源 | 1.7.3 |
| 创建监控Dashboard | 1.7.4 |
| 自定义Panel | 1.7.4.1 |
| 共享你的仪表盘 | 1.7.4.2 |
| 基于Grafana的告警配置 | 1.7.5 |
| 小结 | 1.7.6 |
| 第五章 扩展Prometheus | 1.8 |
| 常用Exporter | 1.8.1 |
| 使用NodeExporter采集主机数据 | 1.8.1.1 |
| 使用MysqLExporter采集Mysql Server数据 | 1.8.1.2 |
| 使用RabbitMQExporter采集RabbitMQ数据 | 1.8.1.3 |

| | |
|-----------------------------------|---------------|
| 使用Cadvisor采集容器数据 | 1.8.1.4 |
| 使用Java创建自定义Metrics | 1.8.2 |
| 使用Golang创建自定义Metrics | 1.8.3 |
| 扩展Spring Boot应用支持应用指标采集 | 1.8.4 |
| 小结 | 1.8.5 |
| 第六章 Prometheus服务发现 | 1.9 |
| 基于文件的服务发现 | 1.9.1 |
| 创建文件 | 1.9.1.1 |
| 配置Prometheus使用基于文件的动态发现 | 1.9.1.2 |
| 定义刷新时间 | 1.9.1.3 |
| 基于Consul的服务发现 | 1.9.2 |
| Consul介绍 | 1.9.2.1 |
| Consul的安装和使用 | 1.9.2.2 |
| 注册Exporter到Consul | 1.9.2.3 |
| 配置Prometheus使用Consul动态发现 | 1.9.2.4 |
| 小结 | 1.9.3 |
| 第七章 运行和管理Prometheus | 1.10 |
| 数据管理 | 1.10.1 |
| 本地存储 | 1.10.1.1 |
| 创建快照 | 1.10.1.2 |
| 远程数据存储 | 1.10.2 |
| 远程读 | 1.10.2.1 |
| 远程写 | 1.10.2.2 |
| 分片 | 1.10.3 |
| 联邦集群 | 1.10.4 |
| 使用Prometheus Operator管理Prometheus | 1.10.5 |
| 使用Promgen管理Prometheus | 1.10.6 |
| 从1.0迁移到2.0 | 1.10.7 |
| 总结 | 1.10.8 |
| 第八章 Kubernetes监控实战 | 1.11 |
| Kubernetes简介 | 1.11.1 |
| 搭建Kubernetes本地测试环境 | 1.11.2 |
| Prometheus Vs Heapster | 1.11.3 |
| 在Kubernetes下部署Prometheus | 1.11.4 |
| 采集Kubelet状态 | 1.11.5 |
| 采集集群状态 | 1.11.6 |
| 采集应用资源用量 | 1.11.7 |
| 采集集群节点状态指标 | 1.11.8 |
| 使用Grafana创建可视化仪表盘 | 1.11.9 |
| 基于Prometheus实现应用的弹性伸缩 | 1.11.10 |

总结

1.11.11

参考资料

1.12

Prometheus实践之路

Prometheus实践之路

全书组织

我们假定你已经对Linux系统以及Docker技术有一定的基本认识，也可能使用过像Java, Golang这样的编程语言，所以我们不会像对任何一个没有任何基础的初学者那样事无巨细的讲述所有事。

第1章，是Prometheus基础的综述，通过一个简单的（使用Prometheus采集主机的监控数据）例子来了解Prometheus是什么？能做什么？以及Prometheus的基础架构。希望读者能从这一章中能对Prometheus有一个基本的理解和认识。

第2章中我们先讲述Prometheus的数据模型以及，以及时间序列存储方式。并且利用Prometheus的数据查询语言(Prometheus Query Language)对监控数据进行查询，聚合以及计算等。

第3章，我们重点放在监控告警部分，作为监控系统的重要能力之一，我们需要及时了解系统环境的变化，因此这一章，我们会介绍如何在Prometheus中定义告警规则，并且结合Prometheus中的另外一个重要组件AlertManager来对告警进行处理。

第4章，“You can't fix what you can't see”我们讨论可视化，如何基于Grafana这一可视化工具自定义我们的可视化仪表盘，介绍Grafana作为一个通用的可视化工具是如何与Prometheus进行配合。

可以看出，从第1章到第4章的部分是基础性的，读者通过前5章对于大部分的研发或者运维人员能够很快的掌握，并且能够使用Prometheus来完成一些基本的日常任务。余下的章节我们会关注到Prometheus的高级用法部分。

第5章，我们会介绍一些常用的Prometheus Exporter的使用场景以及用法。最本章的最后部分我们会带领读者通过Java和Golang实现我们的Exporter,以及如何在现有应用系统上添加Prometheus支持，从而实现应用层面的监控对接。

第6章，我们会了解如何通过Prometheus的服务发现能力，实现动态监控。特别是在云平台或者容器平台中，资源的创建和销毁成本变得如此的低的情况下，通过服务发现如何自动化的去动态发现监控目标,能够充分简化Prometheus的运维和管理难度。

第7章，Prometheus在单个节点的情况下能够轻松完成对N级别资源的监控，但是监控的目标资源以及数据量变得更大的时候，我们如何实现对Prometheus的扩展。这一章节我们重点在Prometheus的数据管理和高可用方面。

第8章，这一章节中我们的另外一位重要成员Kubernetes将会登场，这里我们会带领读者对Kubernetes有一个基本的认识，并且通过Prometheus构建我们的容器云监控系统。并且介绍如何通过Prometheus与Kubernetes结合实现应用程序的弹性伸缩。

目录结构

```

book
|- CHANGELOGS.md 版本更新历史
|- SUMMARY.md 目录
|- chapter[n] 章节
  |- static/ 静态文件资源
  |- README.md 章节头
  |- SUMMARY.md 章节尾

```

本地预览

```
npm install
```

Start Local Preview

```
npm run start
```

作者介绍

郑云龙，全栈工程师，CNCF基金会Certified Kubernetes Administrator。在敏捷和DevOps领域有丰富的实践经验，过去作为敏捷和DevOps技术教练向多家大型企业提供咨询和培训。当前在一家容器创业公司负责CaaS产品研发和设计。

Prometheus实践之路

“You can't fix what you can't see”。Prometheus被誉为下一代监控系统的首选，是继Kubernetes之后的第一个CNCF基金会顶级项目。本书是一本介绍Prometheus以及周边相关技术的实践书籍。本书主要分为三个部分。第一部分，主要通过一个实际的案例介绍了Prometheus的是什么？能做什么？以及Prometheus的基础架构，让读者对Prometheus有一个基本的认识以及概念。第二部分，本书则重点放在Prometheus的高级实践，包括社区提供的已经实现的Exporter的使用场景以及方法，以及教会读者如果通过Prometheus提供的Client Library根据自身需求创建自定义的Exporter。基于如何使用Prometheus的服务发现能力，提升平台的动态能力。在第二部分的最后我们会介绍如何对Prometheus运维管理，根据需求的不同实现Prometheus的扩容。以及如何通过一些其他的开源工具如Prometheus Operator实现对Prometheus的管理。最后，在第三部分，我们将会在Kubernetes下构建基于Prometheus构建我们的容器云监控平台，并且基于监控数据实现对应用的Auto Scaling。

在通读这些内容后，相信读者能够对Prometheus有一个全面的认识。

版本更新历史

- 2018/1/18 初始化目录

第一章：天降奇兵

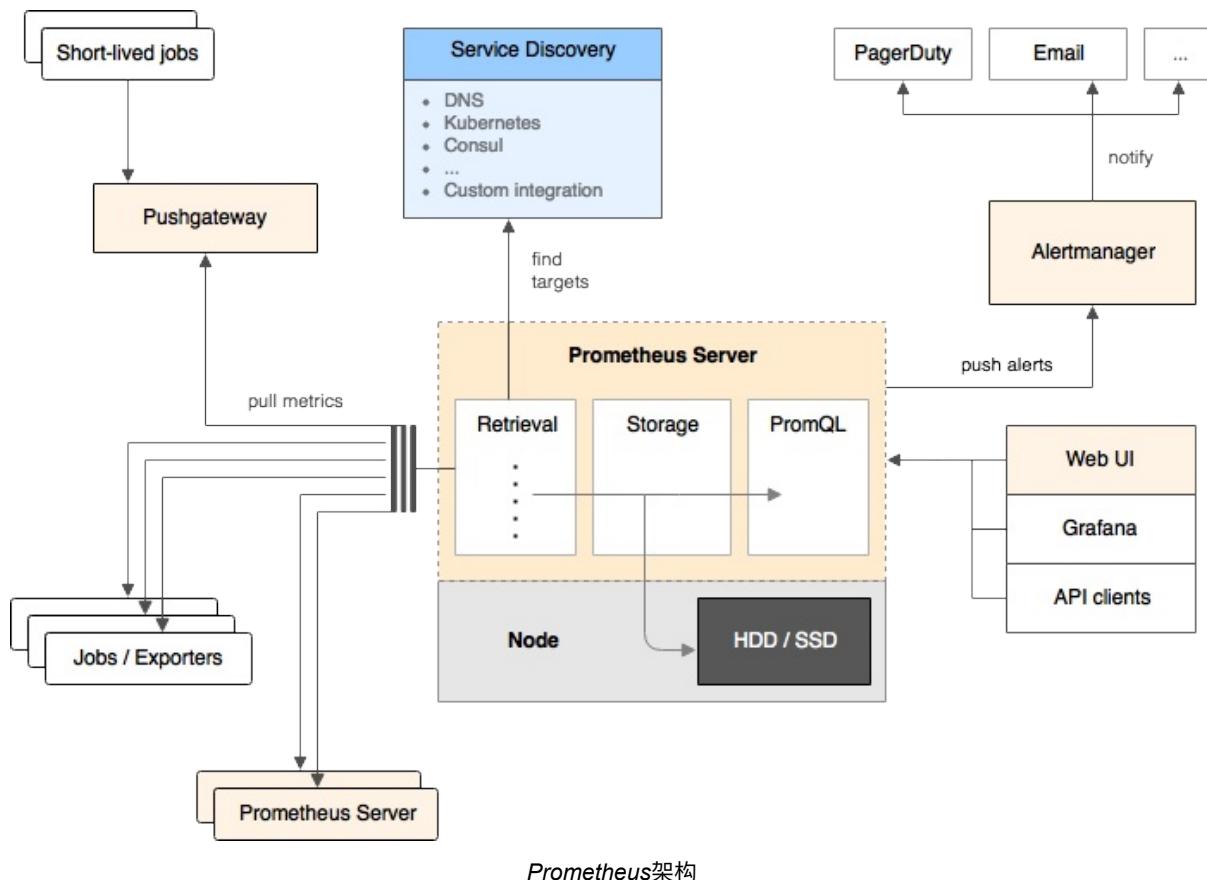
第一章作为本书的第一个部分，我将带领读者快速了解Prometheus。

Prometheus是什么？

Prometheus是SoundCloud开源的监控与告警平台，由于其从推出就提供了完整的基于容器的部署方式，开发者可以快速的基于容器搭建自己的监控平台。因此在Docker社区迅速聚集了大量的人气。

自2012起，有越来越多的组织加入了Prometheus的社区当中，并且有众多的开发者加入并为Prometheus贡献代码。2016年Prometheus正式加入了CNCF基金会(Cloud Native Computing Foundation)，成为继Kubernetes之后的第二个顶级项目。

Prometheus主要包含以下组件：



- Prometheus Server: 负责采集以及存储时间序列数据；
- Exporter: 用于向Prometheus Server暴露目标监控指标的EndPoint， 社区提供了大量的Exporter可以使Prometheus实现对服务器，容器，中间件等监控数据采集。
- AlertManager: 用于处理由Prometheus产生的告警，实现与第三方如，邮件，Slack,Webhook的集成。
- Push Gateway: Prometheus代理层，支持客户端向Push Gateway主动Push数据。Prometheus与Push Gateway之间依然通过Pull的形式收集数据。

一个时序数据库

首先Prometheus是一个基于时间序列(time-series)的数据库系统。Prometheus与其他主流的监控系统都采用了基于时间序列的方式对数据进行存储。其他典型的时序数据库诸如(OpenTSD, InfluxDB等等)。相比于其他的时序数据库，Prometheus中存储时间序列标示包含一个度量名称(Metric)以及多个的键值对(Label)进行标示 (例如 `api_http_requests_total{method="POST", handler="/messages"}`)。因此可以基于多种维度实现对监控数据的聚合。

Prometheus用一种高效的自定义格式存储时间序列，并且将数据存储在内存和本地磁盘上。因此如果希望对Prometheus进行扩展，需要通过功能分片或者联邦集群实现。关于如果实现Prometheus扩展的部分我们会在第四章进行讨论。

一个数据查询引擎

Prometheus提供了灵活高效的数据查询系统，通过Prometheus自定义的数据查询语言，可以对数据进行查询和聚合，实现各种报表和数据分析。

关于使用Prometheus进行数据聚合分析的部分我们会在第二章节进行讨论。

一个监控告警平台

Prometheus支持用户创建基于Prometheus的自定义查询语言，定义告警。并且结合AlertManager组件实现件监控告警，可以与第三方工具和平台进行集成。

一个监控框架

社区同时提供了大量的第三方Exporter。可以快速实现对诸如服务器，容器，中间件的监控指标采集。Prometheus同时提供超过10中以上的客户端SDK，基于这些客户端SDK，我们可以轻松实现我们自己的Exporter或者直接在我们的应用程序上集成Prometheus。

使用二进制包安装Prometheus

本小节我们尝试在Ubuntu/trusty版本下基于二进制软件包安装Prometheus并且通过NodeExporter采集主机监控数据。

创建本地用户

```
sudo useradd --no-create-home prometheus  
sudo useradd --no-create-home node_exporter
```

```
sudo mkdir /etc/prometheus  
sudo mkdir /var/lib/prometheus
```

```
sudo chown prometheus:prometheus /etc/prometheus  
sudo chown prometheus:prometheus /var/lib/prometheus
```

获取安装包

```
cd ~  
curl -LO https://github.com/prometheus/prometheus/releases/download/v2.0.0/prometheus-2.0.0.linux-amd64.tar.gz
```

```
tar xvf prometheus-2.0.0.linux-amd64.tar.gz
```

```
sudo cp prometheus-2.0.0.linux-amd64/prometheus /usr/local/bin/  
sudo cp prometheus-2.0.0.linux-amd64/promtool /usr/local/bin/
```

```
sudo chown prometheus:prometheus /usr/local/bin/prometheus  
sudo chown prometheus:prometheus /usr/local/bin/promtool
```

```
sudo cp -r prometheus-2.0.0.linux-amd64/consoles /etc/prometheus  
sudo cp -r prometheus-2.0.0.linux-amd64/console_libraries /etc/prometheus
```

```
sudo chown -R prometheus:prometheus /etc/prometheus/consoles  
sudo chown -R prometheus:prometheus /etc/prometheus/console_libraries
```

```
rm -rf prometheus-2.0.0.linux-amd64.tar.gz prometheus-2.0.0.linux-amd64
```

创建Prometheus配合文件

```
sudo vim /etc/prometheus/prometheus.yml
```

```
global:  
  scrape_interval: 15s  
  
scrape_configs:  
  - job_name: 'prometheus'  
    scrape_interval: 5s
```

```
static_configs:  
  - targets: ['localhost:9090']
```

```
sudo chown prometheus:prometheus /etc/prometheus/prometheus.yml
```

运行Prometheus

```
sudo -u prometheus /usr/local/bin/prometheus \  
  --config.file /etc/prometheus/prometheus.yml \  
  --storage.tsdb.path /var/lib/prometheus/ \  
  --web.console.templates=/etc/prometheus/consoles \  
  --web.console.libraries=/etc/prometheus/console_libraries
```

```
level=info ts=2018-01-08T09:28:02.177521724Z caller=main.go:215 msg="Starting Prometheus" version="(version=2.0.0, branch=HEAD, revision=0a74f98628a0463dddc90528220c94de5032d1a0)"  
level=info ts=2018-01-08T09:28:02.178437351Z caller=main.go:216 build_context="(go=go1.9.2, user=root@615b82cb36b6, date=20171108-07:11:59)"  
level=info ts=2018-01-08T09:28:02.179001141Z caller=main.go:217 host_details="(Linux 3.13.0-100-generic #147-Ubuntu SMP Tue Oct 18 16:48:51 UTC 2016 x86_64 vagrant-ubuntu-trusty-64 (none))"  
level=info ts=2018-01-08T09:28:02.182742386Z caller=web.go:380 component=web msg="Start listening for connections" address="0.0.0.0:9090"  
level=info ts=2018-01-08T09:28:02.188609949Z caller=main.go:314 msg="Starting TSDB"  
level=info ts=2018-01-08T09:28:02.193920127Z caller=targetmanager.go:71 component="target manager" msg="Starting target manager..."  
level=info ts=2018-01-08T09:28:02.196274334Z caller=main.go:326 msg="TSDB started"  
level=info ts=2018-01-08T09:28:02.197030914Z caller=main.go:394 msg="Loading configuration file" filename=/etc/prometheus/prometheus.yml  
level=info ts=2018-01-08T09:28:02.202861729Z caller=main.go:371 msg="Server is ready to receive requests."
```

```
sudo vim /etc/systemd/system/prometheus.service
```

```
[Unit]  
Description=Prometheus  
Wants=network-online.target  
After=network-online.target  
  
[Service]  
User=prometheus  
Group=prometheus  
Type=simple  
ExecStart=/usr/local/bin/prometheus \  
  --config.file /etc/prometheus/prometheus.yml \  
  --storage.tsdb.path /var/lib/prometheus/ \  
  --web.console.templates=/etc/prometheus/consoles \  
  --web.console.libraries=/etc/prometheus/console_libraries  
  
[Install]  
WantedBy=multi-user.target
```

```
sudo systemctl daemon-reload  
sudo systemctl status prometheus  
sudo systemctl enable prometheus
```

安装NodeExporter

```
cd ~
```

```
curl -LO https://github.com/prometheus/node_exporter/releases/download/v0.15.1/node_exporter-0.15.1.linux-amd64.tar.gz
```

```
tar xvf node_exporter-0.15.1.linux-amd64.tar.gz  
sudo cp node_exporter-0.15.1.linux-amd64/node_exporter /usr/local/bin  
sudo chown node_exporter:node_exporter /usr/local/bin/node_exporter  
rm -rf node_exporter-0.15.1.linux-amd64.tar.gz node_exporter-0.15.1.linux-amd64
```

```
sudo vim /etc/systemd/system/node_exporter.service
```

```
[Unit]  
Description=Node Exporter  
Wants=network-online.target  
After=network-online.target  
  
[Service]  
User=node_exporter  
Group=node_exporter  
Type=simple  
ExecStart=/usr/local/bin/node_exporter  
  
[Install]  
WantedBy=multi-user.target
```

配置Prometheus采集Node Exporter数据

```
sudo vim /etc/prometheus/prometheus.yml
```

```
- job_name: 'node_exporter'  
  scrape_interval: 5s  
  static_configs:  
    - targets: ['localhost:9100']
```

/etc/prometheus/prometheus.yml

```
global:  
  scrape_interval: 15s  
  
scrape_configs:  
  - job_name: 'prometheus'  
    scrape_interval: 5s  
    static_configs:  
      - targets: ['localhost:9090']  
  - job_name: 'node_exporter'  
    scrape_interval: 5s  
    static_configs:  
      - targets: ['localhost:9100']
```

启用Basic Auth认证

Docker简介

本章小结

参考资料:

- <https://www.robustperception.io/scaling-and-federating-prometheus/>

参考资料

- <https://www.robustperception.io/how-much-ram-does-my-prometheus-need-for-ingestion/>
- https://www.youtube.com/watch?list=PLoz-W_CUquUICq-Q0hy53ToIAhaED9vm&v=likpVWB5Lvo

参考资料

- <https://github.com/kubernetes/kube-state-metrics>

参考资料

Install & Configuration

- <https://www.digitalocean.com/community/tutorials/how-to-install-prometheus-on-ubuntu-16-04>

Storage

- <https://coreos.com/blog/prometheus-2.0-storage-layer-optimization>

Kubernetes

- <https://docs.bitnami.com/kubernetes/how-to/configure-autoscaling-custom-metrics/>

Others

- <https://news.ycombinator.com/item?id=12455045>
- <https://github.com/digitalocean/vulcan>
- <https://github.com/coreos/prometheus-operator/blob/master/Documentation/high-availability.md>
- <https://github.com/katosys/kato/issues/43>
- <https://www.robustperception.io/tag/tuning/>
- <https://www.robustperception.io/how-much-ram-does-my-prometheus-need-for-ingestion/>