# Go培训第14天

tony

# Outline

1. 日志收集系统架构

2. 内存信息采集

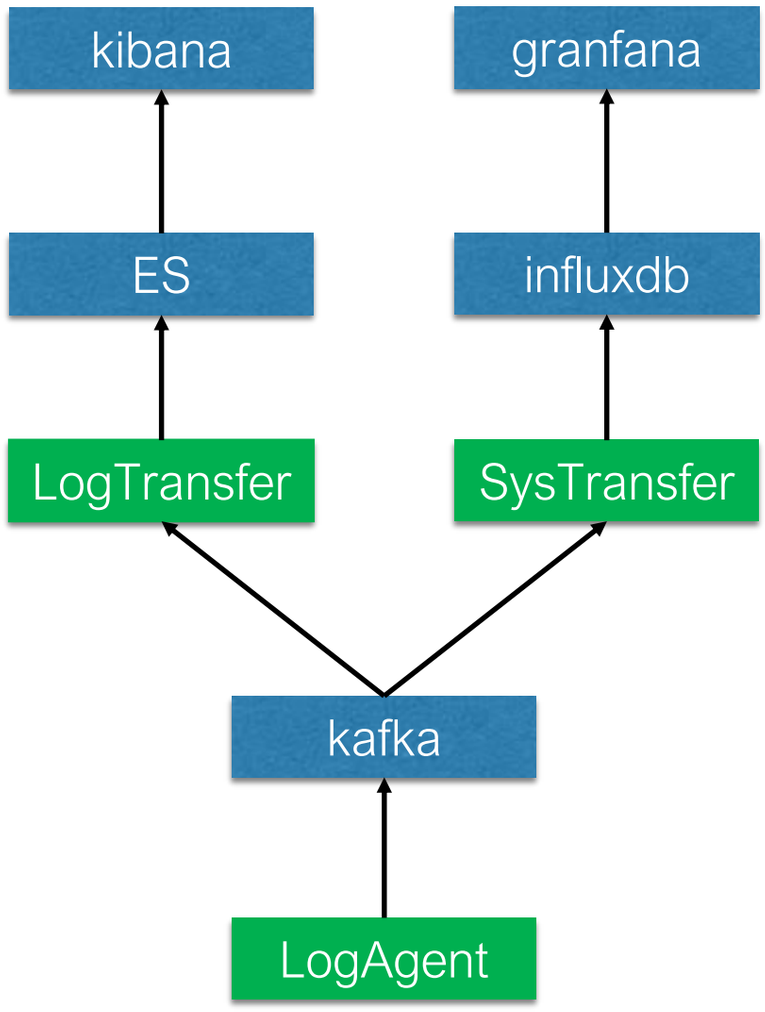
3. 磁盘信息采集

4. 网络信息采集

5. 集成logagent

# 日志收集系统架构

1. 日志收集系统架构



# ElasticSearch介绍与使用

## 2. EasticSearch安装

1) 下载ES，下载地：github.com/elastic/elasticsearch

2) 修改config/elasticsearch.ymal配置：
network.host: 本地ip
node.name:node_1

3) 启动es， ./bin/elasticsearch.bat

# ElasticSearch介绍与使用

3. EasticSearch基本概念

1) 分布式的搜索引擎，基于Lucene。

2) EasticSearch每个实例称之为节点，一组节点构成一个集群。

3) 本质上是一个分布式数据库，所有数据保存在磁盘上。一般一主多幅本的结构。

# ElasticSearch介绍与使用

4. 索引概念

1) 索引相当于mysql中的数据库。ES会索引所有字段，然后写入一个倒排索引中。

2) curl -X GET 'http://localhost:9200/_cat/indices?v

3) Index当中的单条记录称为文档document。Document使用json表示。

5. 新建和删除索引

1) 新建索引：curl -X PUT 'localhost:9200/weather'

2) 删除索引：curl -X DELETE 'localhost:9200/weather'

6. 新增记录

1) 向指定的/Index/Type发送put请求，例如:

2) Curl -H "Content-Type:application/json" -X PUT 'localhost:9200/accounts/person/1'

```
$ curl -X PUT 'localhost:9200/accounts/person/1' -d '
{
  "user": "张三",
  "title": "工程师",
  "desc": "数据库管理"
}'
```

## 7. 新增记录

1)新增记录的时候，也可以不指定 Id，这时要改成 POST 请求。
，例如:

2) curl -X POST 'localhost:9200/accounts/person' -d

```
$ curl -X POST 'localhost:9200/accounts/person' -d '
{
  "user": "李四",
  "title": "工程师",
  "desc": "系统管理"
}'
```

## 8. 查找记录

1) 向/Index/Type/Id发出 GET 请求，就可以查看这条记录

```
$ curl 'localhost:9200/accounts/person/1?pretty=true'
```

## 9. 删除记录

1) 向/Index/Type/Id发出 DELETE 请求，就删除这条记录

```
$ curl -X DELETE 'localhost:9200/accounts/person/1'
```

# ElasticSearch介绍与使用

## 10. 更新记录

1) 向/Index/Type/Id发出 PUT 请求，就更新这条记录

```
$ curl -X PUT 'localhost:9200/accounts/person/1' -d '
{
    "user" : "张三",
    "title" : "工程师",
    "desc" : "数据库管理，软件开发"
}'
```

# ElasticSearch介绍与使用

## 11.数据查询

1)使用 GET 方法，直接请求/Index/Type/_search，就会返回所有记录

```
$ curl 'localhost:9200/accounts/person/_search'
```

# ElasticSearch介绍与使用

## 12. 全文检索

1) Elastic 的查询非常特别，使用自己的查询语法，要求 GET 请求带有数据体.

```
$ curl 'localhost:9200/accounts/person/_search'  -d '
{
  "query" : { "match" : { "desc" : "软件" }}
}'
```

2) Elastic 默认一次返回10条结果，可以通过size字段改变这个设置。

```
$ curl 'localhost:9200/accounts/person/_search'  -d '
{
  "query" : { "match" : { "desc" : "管理" }},
  "size": 1
}'
```

## 12. 全文检索

1) 还可以通过from字段，指定位移。

```
$ curl 'localhost:9200/accounts/person/_search'  -d '
{
  "query" : { "match" : { "desc" : "管理" }},
  "from": 1,
  "size": 1
}'
```

# 13. Kibana介绍与使用

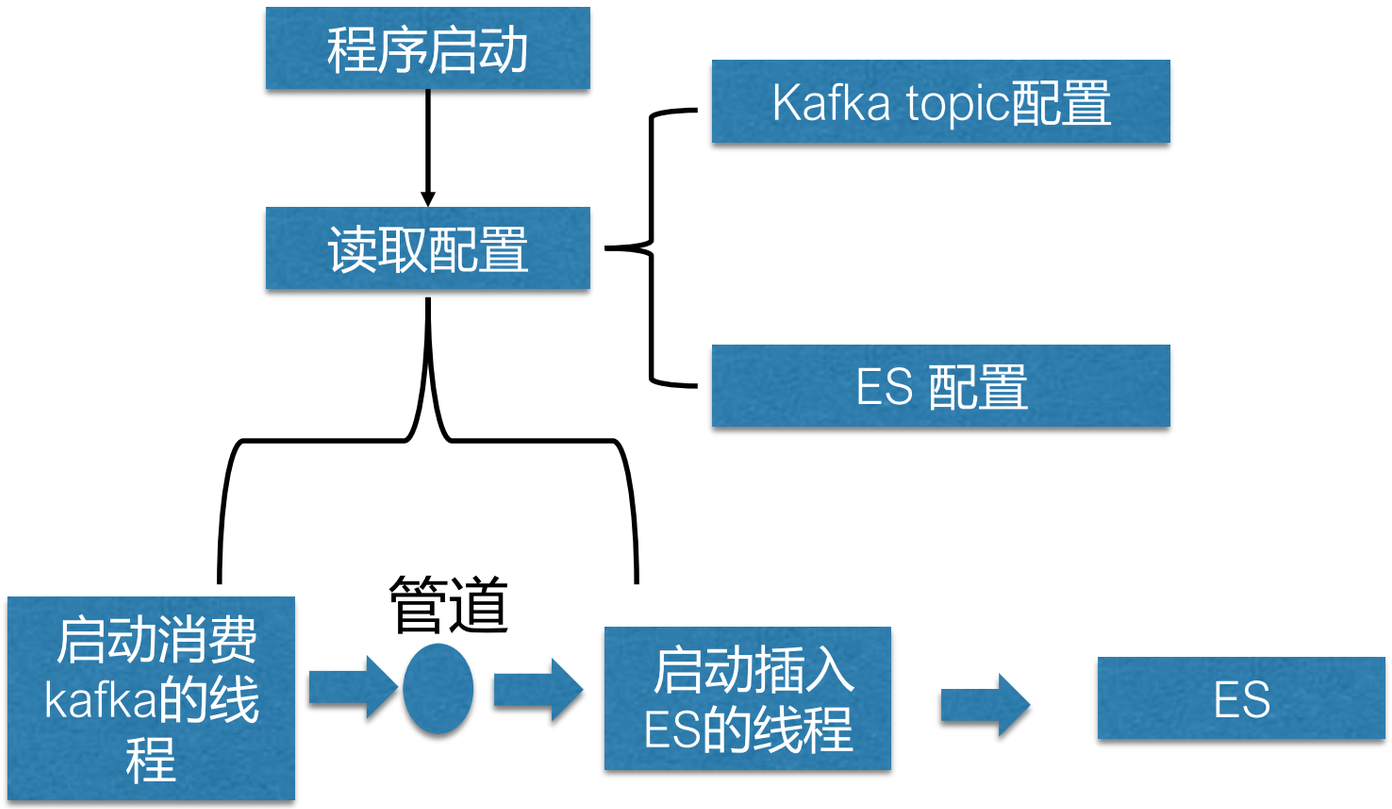
1) 配置es的地址：elasticsearch.url: "http://192.168.1.25:9200

2) bin\kibana.bat

3) 访问kibana，http://localhost:5601

```
package main

import (
	"fmt"
	elastic "gopkg.in/olivere/elastic.v2"
)

type Tweet struct {
	User    string
	Message string
}

func main() {
	client, err := elastic.NewClient(elastic.SetSniff(false), elastic.SetURL("http://192.168.31.177:9200/"))
	if err != nil {
		fmt.Println("connect es error", err)
		return
	}

	fmt.Println("conn es succ")

	tweet := Tweet{User: "olivere", Message: "Take Five"}
	_, err = client.Index().
		Index("twitter").
		Type("tweet").
		Id("1").
		BodyJson(tweet).
		Do()
	if err != nil {
		// Handle error
		panic(err)
		return
	}

	fmt.Println("insert succ")
}
```

# 14. LogTransfer 程序设计



# grafana介绍与使用

## 15. grafana介绍与使用

1) 解压grafana，进入grafana.5.3.2目录

2) cp conf/sample.ini conf/custom.ini

3) 启动grafana， bin\grafana-server.exe

4) 访问grafana，http://localhost:3000

# 16. influxdb介绍与使用

1) 开源的分布式时序、时间和指标数据库，使用Go语言编写，无需外部依赖

2) 非常适合存储各种各样的统计指标，采用数据。

3) 使用Go语言开发，自带管理界面。

# 16. influxdb介绍与使用

| InfluxDB中的名词 | 传统数据库中的概念 |
| --- | --- |
| database | 数据库 |
| measurement | 数据库中的表 |
| points | 表里面的一行数据 |

# influxdb介绍与使用

## 17. Influxdb中的point

Point由时间戳（time）、数据（field）、标签（tags）组成。

Point相当于传统数据库里的一行数据，如下表所示：

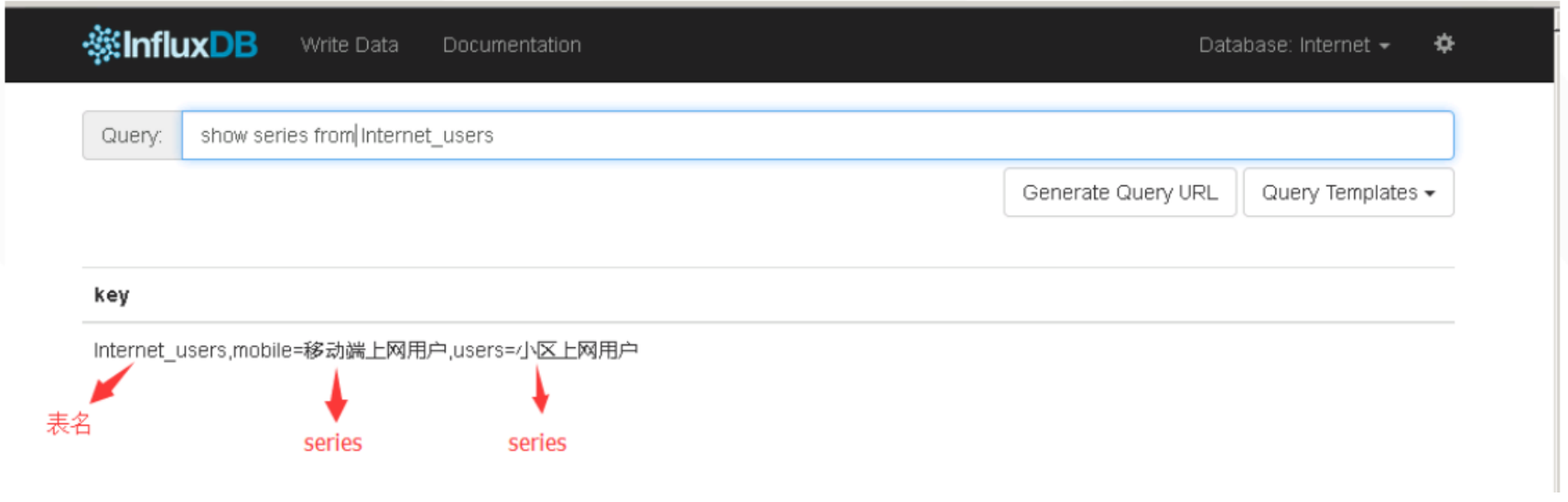
| **Point属性** | **传统数据库中的概念** |
| --- | --- |
| time | 每个数据记录时间，是数据库中的主索引(会自动生成) |
| fields | 各种记录值（没有索引的属性）也就是记录的值：温度，　湿度 |
| tags | 各种有索引的属性：地区，海拔 |

# influxdb介绍与使用

## 18. Influxdb中的series

所有在数据库中的数据，都需要通过图表来展示，而这个series表示这个表里面的数据，可以在图表上画成几条线：通过tags排列组合算出来。

如下所示：



# influxdb介绍与使用

## 19. Influxdb 操作

```go
func WritesPoints(cli client.Client) {
        bp, err := client.NewBatchPoints(client.BatchPointsConfig{
                Database:  MyDB,
                Precision: "s",
        })
        if err != nil {
                log.Fatal(err)
        }

        tags := map[string]string{"cpu": "ih-cpu"}
        fields := map[string]interface{}{
                "idle":   20.1,
                "system": 43.3,
                "user":   86.6,
        }

        pt, err := client.NewPoint(
                "cpu_usage",
                tags,
                fields,
                time.Now(),
        )
        if err != nil {
                log.Fatal(err)
        }
        bp.AddPoint(pt)

        if err := cli.Write(bp); err != nil {
                log.Fatal(err)
        }
}
```

# 课后作业

1. 把今天的日志收集客户端，自己实现一遍