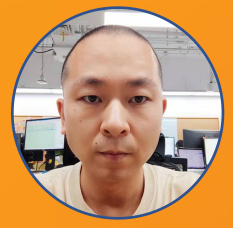
利用夜莺扩展能力打造全方位监控系统

×



喻波

滴滴
专家工程师

GopherChina 2021

目 录

第一部分

运维监控需求来源

如果贵司的业务强依赖IT技术，IT故障会直接影响营业收入，稳定性体系一定要重视起来，而监控，就是稳定性体系中至关重要的一环

运维监控需求来源

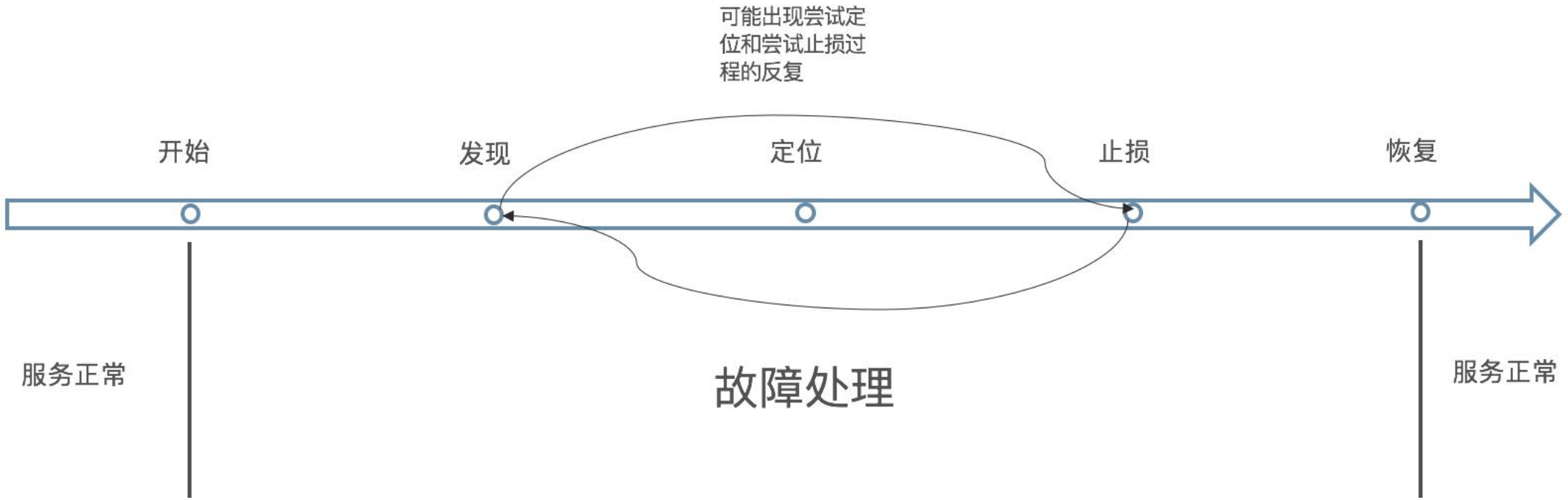
左图是2013年的一个新闻，讲Google宕机的影响。2020年也出现过aws大规模宕机的情况，影响不止是55万美元，直接影响大半个互联网！

2018年有美国调研机构指出，如果服务器宕机1分钟，银行会损失27万美元，制造业会损失42万美元

美团故障？滴滴故障？腾讯故障？

运维监控需求来源

如何减少服务停摆导致的经济损失？尽快发现故障并止损！故障处理过程中，监控是『发现』和『定位』两个环节的关键工具。故障处理过程的首要原则是『止损』，因此，过程中的『发现』和『定位』都是面向尽快『止损』来实现。



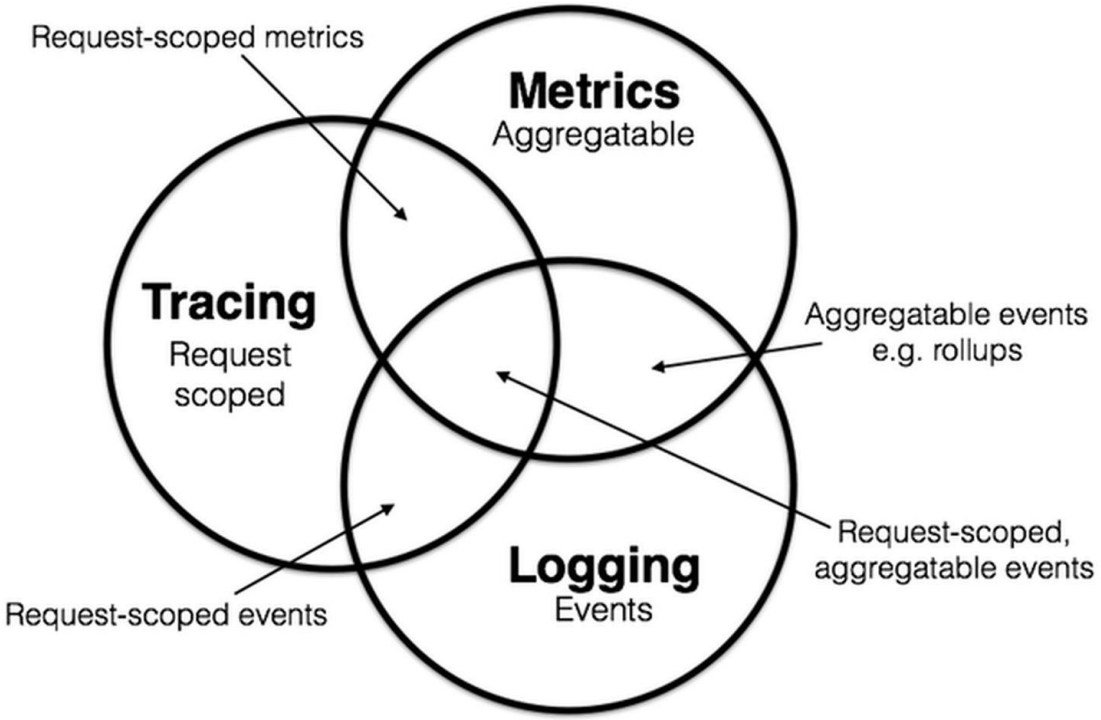
第二部分

监控痛点：全面完备、跨云

端上、链路、资源、组件、应用多维度跨云监控，不管哪个环节出问题都能及时感知

产品要求

- 端上
  - 卡顿
  - 崩溃
- 链路
  - 连通性
  - 链路质量
- 服务端
  - 硬件资源
  - 组件服务
  - 业务应用



第三部分

夜莺介绍：国产开源监控系统

国产开源监控产品相对比较匮乏，夜莺希望重新定义国产开源监控，支持云原生监控，经受了滴滴大规模生产检验

Nightingale

夜莺是**新一代国产智能监控平台**，既可以解决传统物理机虚拟机的场景，也可以解决容器的场景。衍生自Open-Falcon和滴滴Odin监控，经受了包括小米、美团、滴滴在内的数百家企业的生产环境验证，**简单可依赖，好用到爆！**

3500+ star

600+ issue

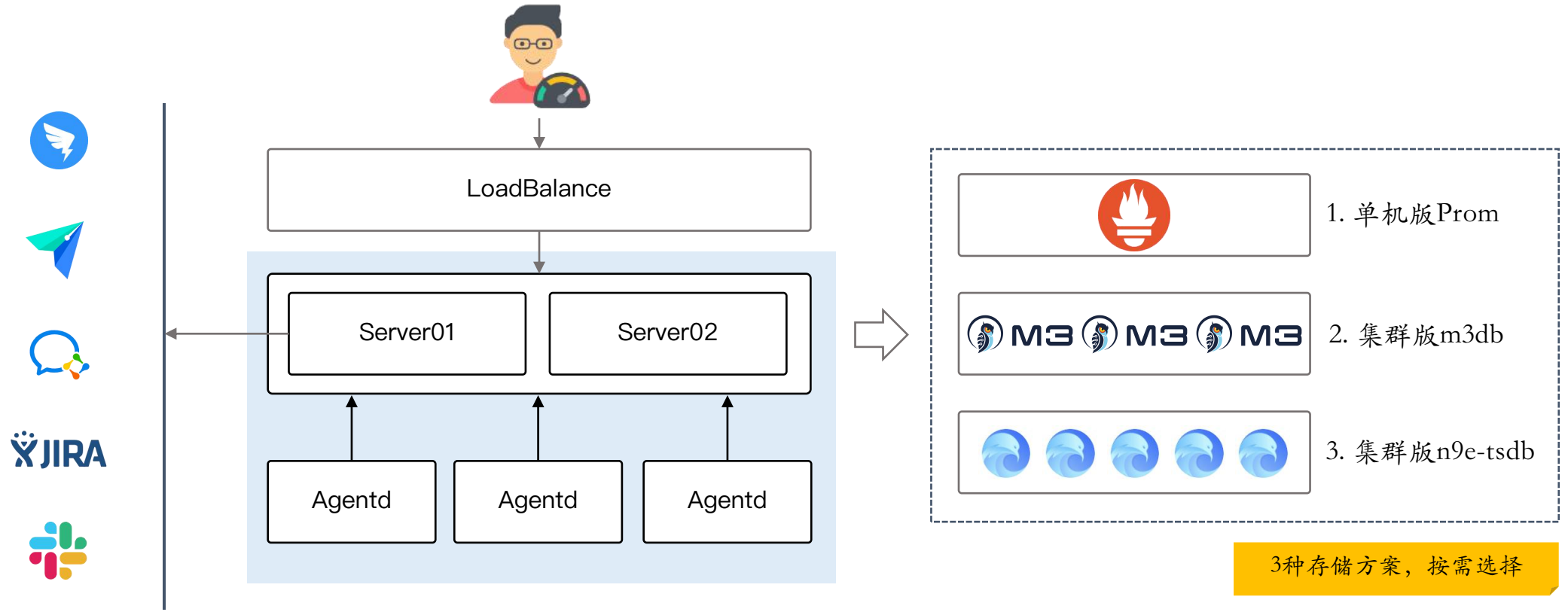
500+ fork

项目：https://github.com/didi/nightingale 官网：https://n9e.didiyun.com/

Nightingale 众多企业已上生产，共同打磨夜莺

上图展示部分社区用户，加入夜莺社群，请联系微信：UlricQin

Nightingale 众多企业已上生产，共同打磨夜莺



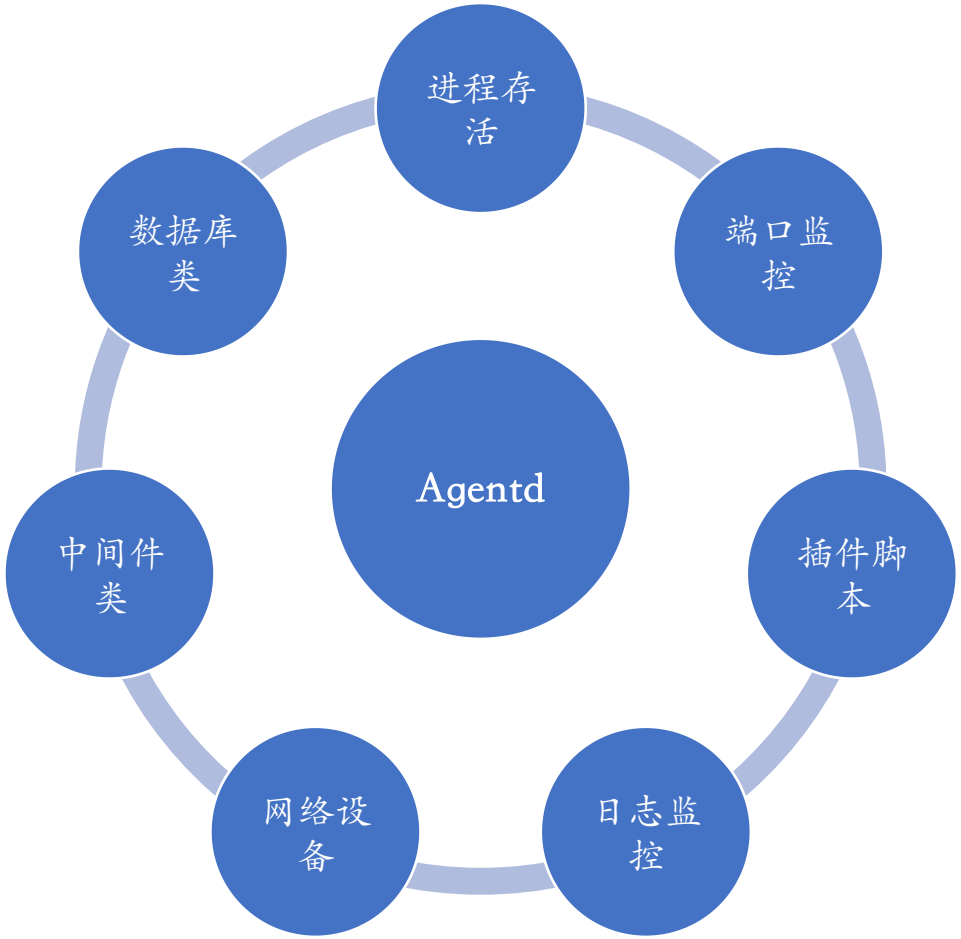
第四部分

夜莺设计实现 Agentd 数据采集

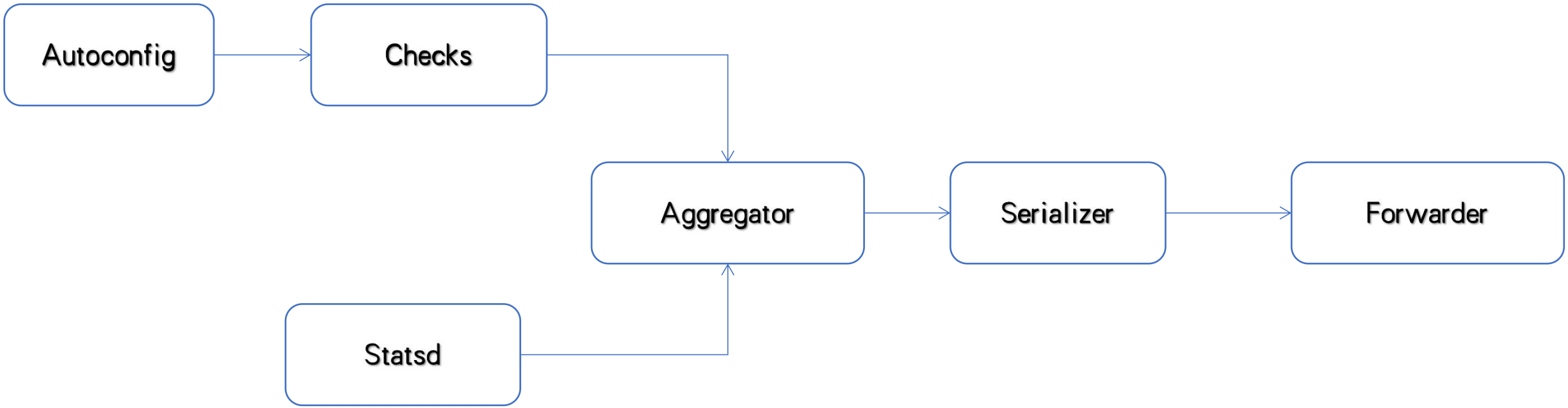
监控系统的核心功能，是数据采集、存储、分析、展示，完备性看采集能力，是否能够兼容并包，纳入更多生态的能力，至关重要

夜莺数据采集

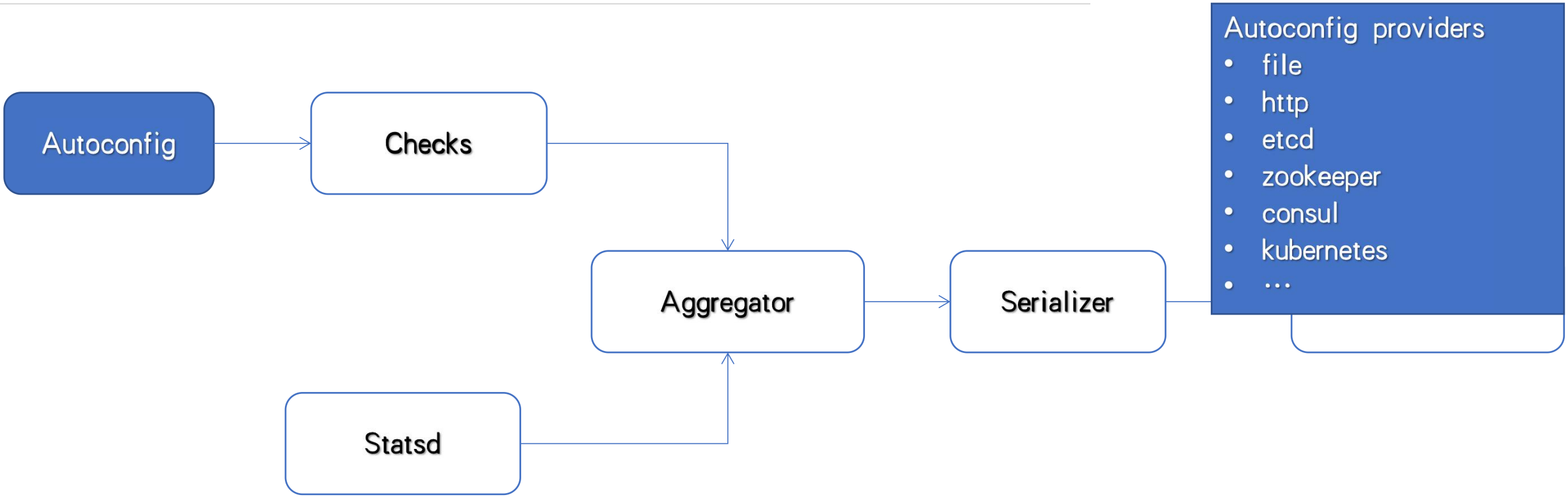
- 支持在web上配置采集策略，不同的采集可以指定不同的探针机器、目标机器，便于管理和知识传承
- 独创在端上流式读取日志，根据正则提取指标的机制，轻量易用，无业务侵入性
- 内置集成了多种数据库中间件的采集以及网络设备的采集，复用telegraf和datadog-agent的能力
- 支持statsd的udp协议，用于业务应用的apm监控分析



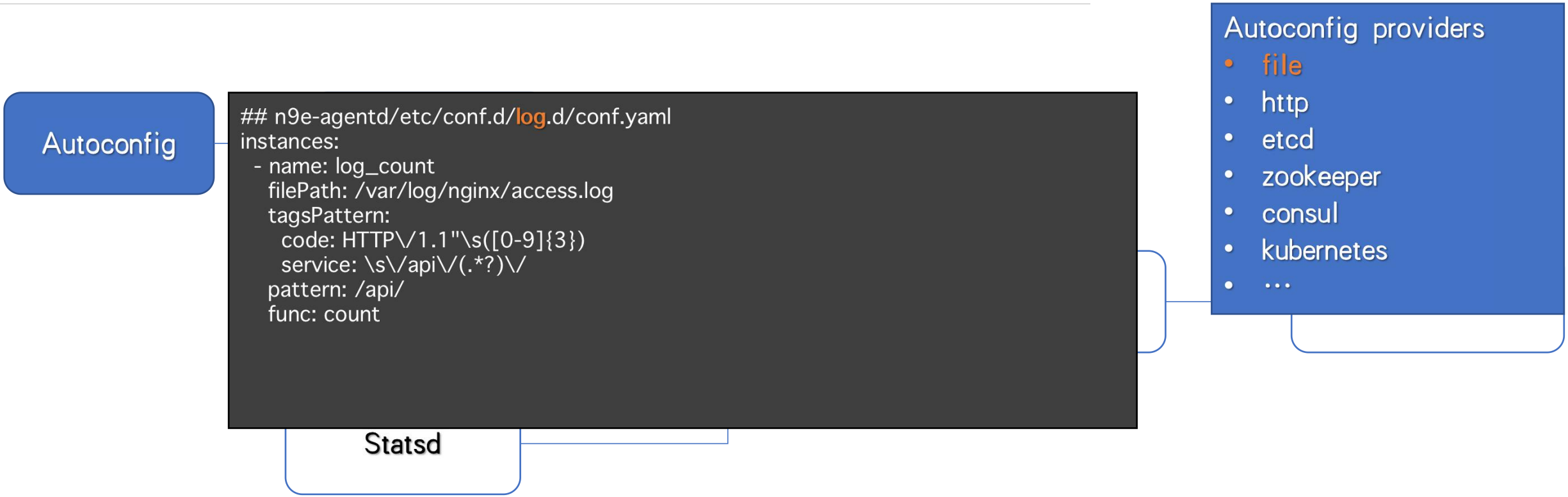
夜莺数据采集


Autoconfig
Checks
Aggregator
Serializer
Forwarder
Statsd

夜莺数据采集


Autoconfig
Checks
Aggregator
Serializer
Statsd
Autoconfig providers
• file
• http
• etcd
• zookeeper
• consul
• kubernetes
• …

夜莺数据采集


Autoconfig
n9e-agentd/etc/conf.d/log.d/conf.yaml
instances:
- name: log_count
filePath: /var/log/nginx/access.log
tagsPattern:
code: HTTP\/1.1"\s([0-9]{3})
service: \s\/api\/(.*?)\/
pattern: /api/
func: count
Statsd
Autoconfig providers
file
http
etcd
zookeeper
consul
kubernetes
…

夜莺数据采集

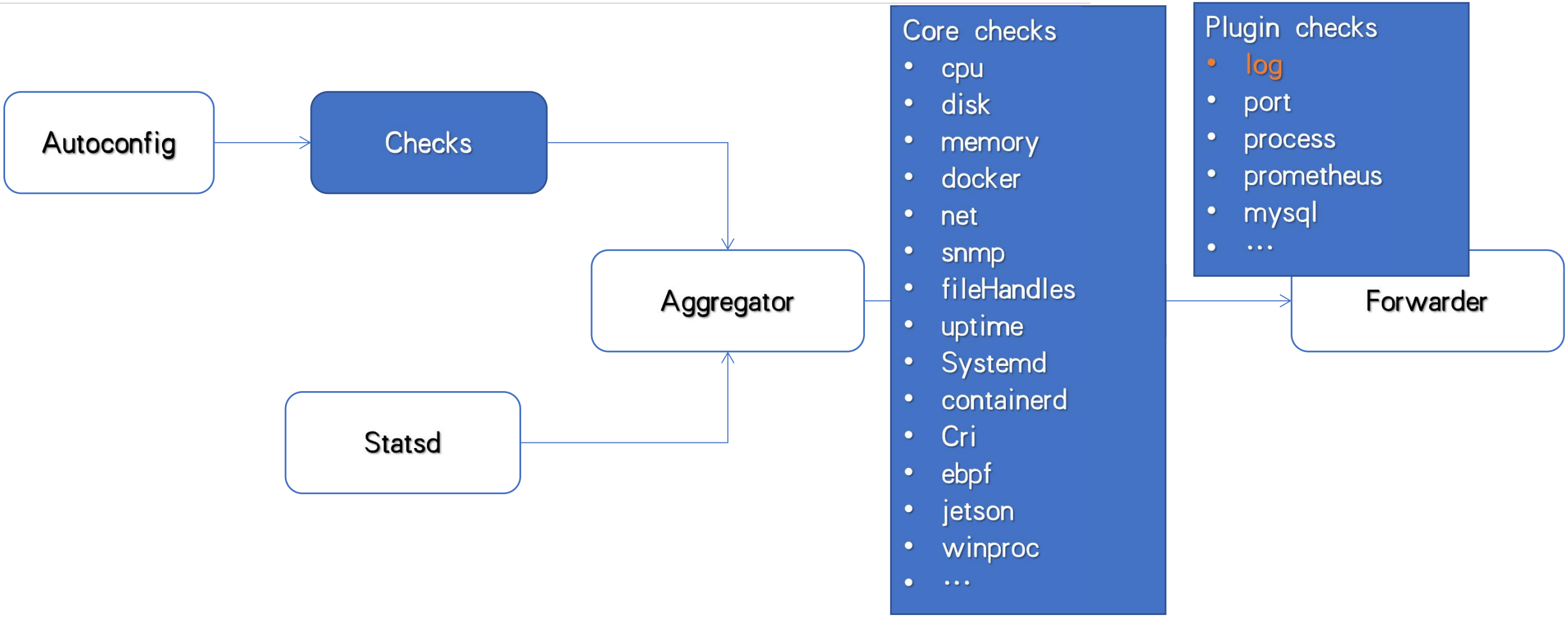
Autoconfig

```
$ curl http://172.20.70.14:8080/api/collect-rules
[
  {
    "ID": 1001,
    "name": "log-name",
    "type": "log",
    "data":
"{\"adIdentifiers\":null,\"clusterCheck\":false,\"ignoreAutodiscoveryTags\":false,\"initC
onfig\":null,\"instances\":[{\"filePath\":\"/var/log/nginx/access.log\",\"func\":\"count
\",\"name\":\"log.count\",\"pattern\":\"/api/\",\"tagsPattern\":{\"code\":\"HTTP\\\\
/1.1\\\"\\\\s([0-
9]{3})\",\"service\":\"\\\\s\\\\/api\\\\/(.*?)\\\\/\"}}],\"jmxMetrics\":null,\"logs\":n
ull}"
  }
]
```

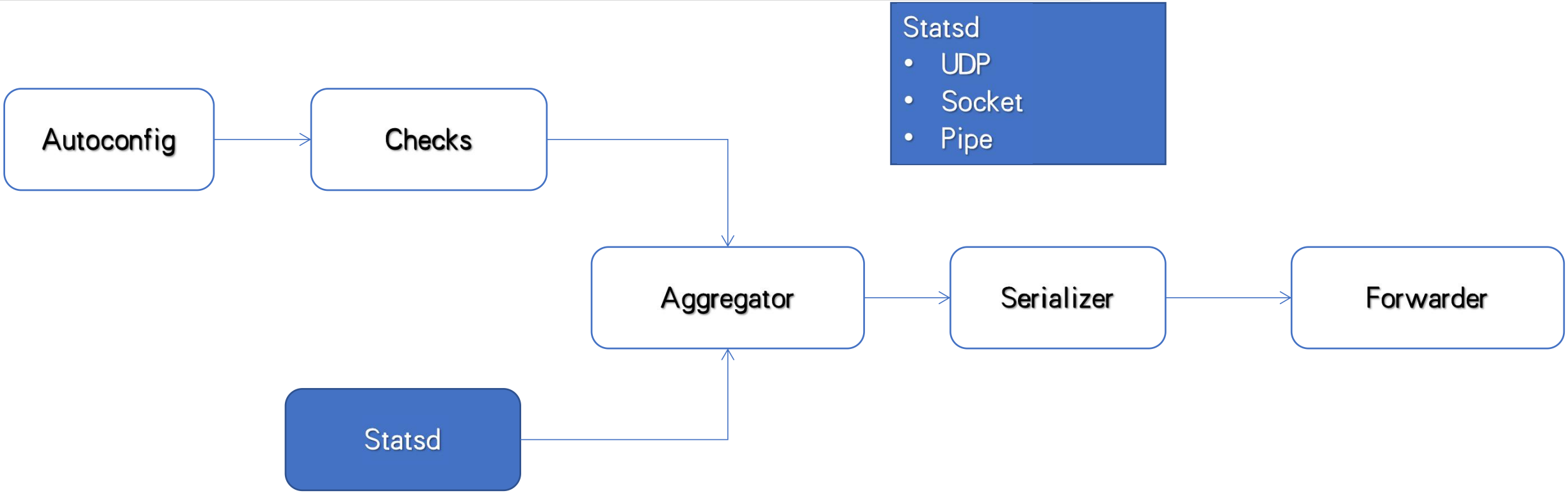
Autoconfig providers

- file
- http
- etcd
- zookeeper
- consul
- kubernetes
- …

夜莺数据采集


Autoconfig
Checks
Aggregator
Statsd
Core checks
cpu
disk
memory
docker
net
snmp
fileHandles
uptime
Systemd
containerd
Cri
ebpf
jetson
winproc
…
Plugin checks
log
port
process
prometheus
mysql
…
Forwarder

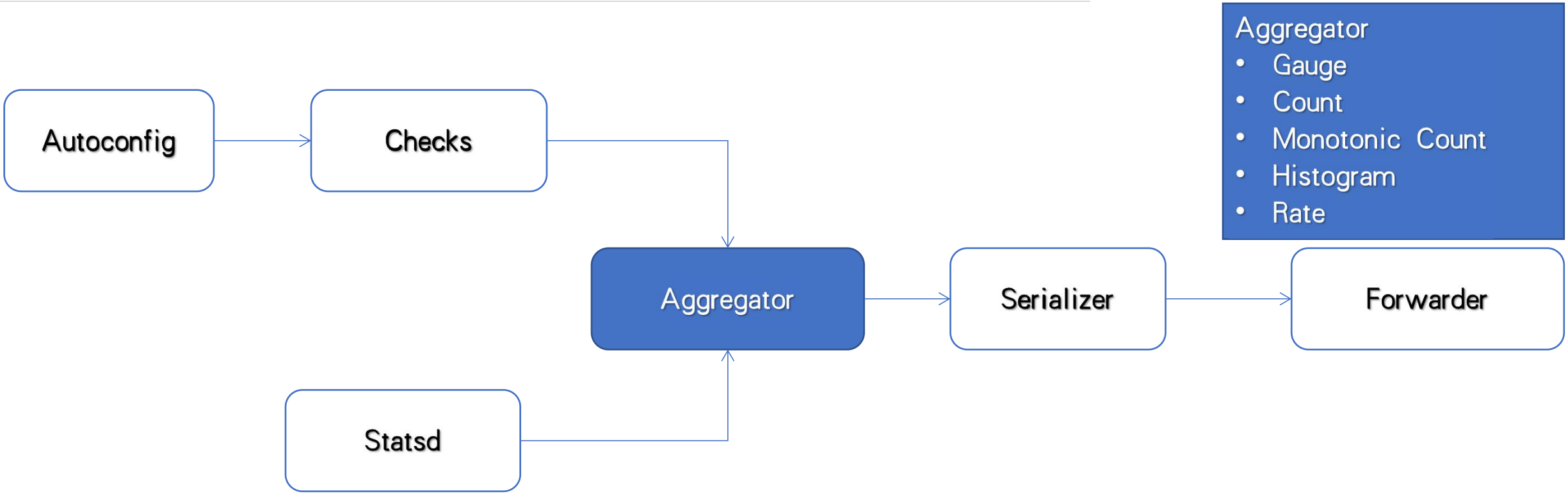
夜莺数据采集


Statsd
• UDP
• Socket
• Pipe
Autoconfig
Checks
Aggregator
Serializer
Forwarder
Statsd

夜莺数据采集

```
# 这个报文里包含了一条 metricName: my_metric, value: 21, type=Gauge, tag: tag1=1, tag2=2 的记录
my_metric:21|g|#tag1:1,tag2:2

#发送到stasd的接口即可,使用sdk来实现也很容易

import (
	"log "

	"github.com/DataDog/datadog-go/statsd"
)

func main() {
	client, err := statsd.New("127.0.0.1:8125" ,
		statsd.WithTags([]string{"env:prod", "service:myservice"}),
	)
	if err != nil {
		log.Fatal(err)
	}

	client.Gauge("my_metrics", 21, []string{"tag1:1", "tag2:2"}, 1)
	client.Close()
}
```

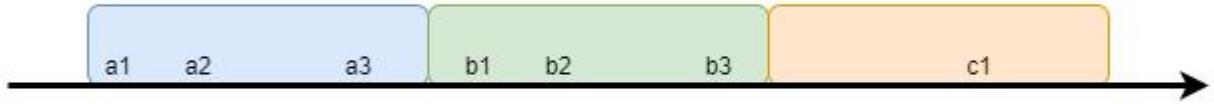
夜莺数据采集


Aggregator
• Gauge
• Count
• Monotonic Count
• Histogram
• Rate
Autoconfig
Checks
Aggregator
Serializer
Forwarder
Statsd

夜莺数据采集

Autoconfig

Aggragator



Aggregator
- Gauge
- Count
- Monotonic Count
- Histogram
- Rate

Forwarder

gauge

取每个周期的最后一个值

- a3
- b3
- c1

count

周期内求和

- a1 + a2 + a3
- b1 + b2 + b3
- c1

monotonic_count

周期内的变化值(delta)

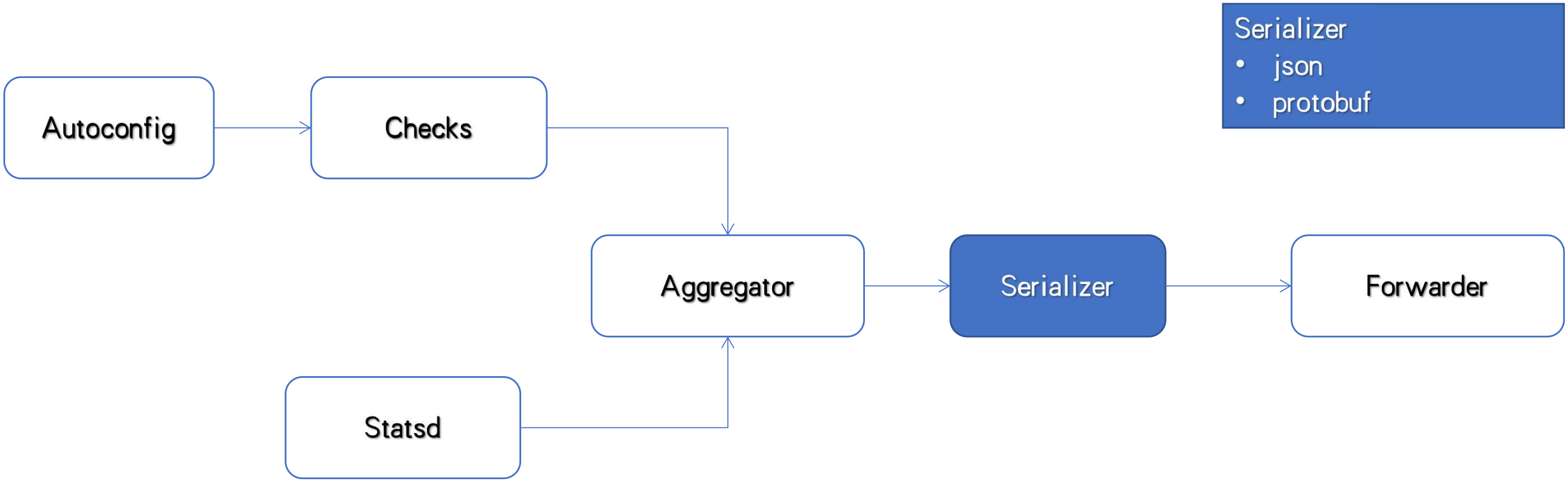
- a3 - a1
- b3 - a3
- c1 - b3

rate

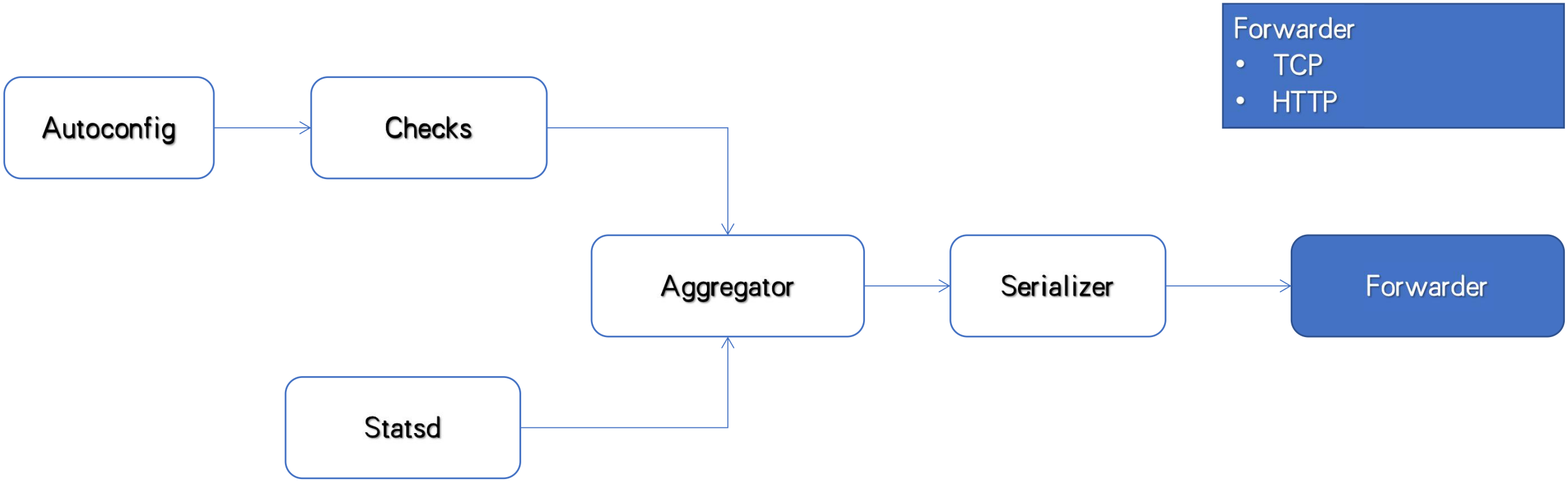
周期内最近2次采样的变化率

- (a3 - a2) / (a3.ts - a2.ts)
- (b3 - b2) / (b3.ts - b2.ts)
- (c1 - b3) / (c1.ts - b3.ts)

夜莺数据采集


Serializer
• json
• protobuf
Autoconfig
Checks
Aggregator
Serializer
Forwarder
Statsd

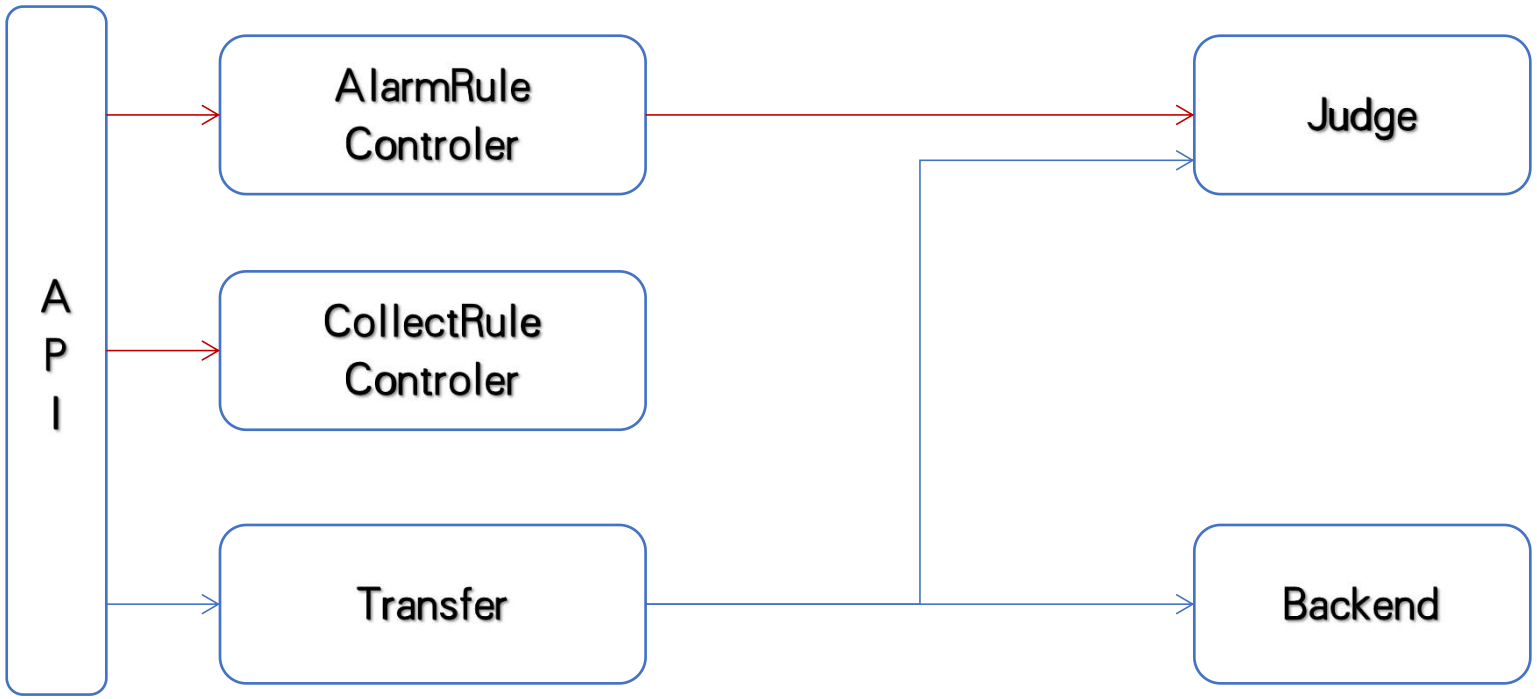
夜莺数据采集


Forwarder
• TCP
• HTTP
Autoconfig
Checks
Aggregator
Serializer
Forwarder
Statsd

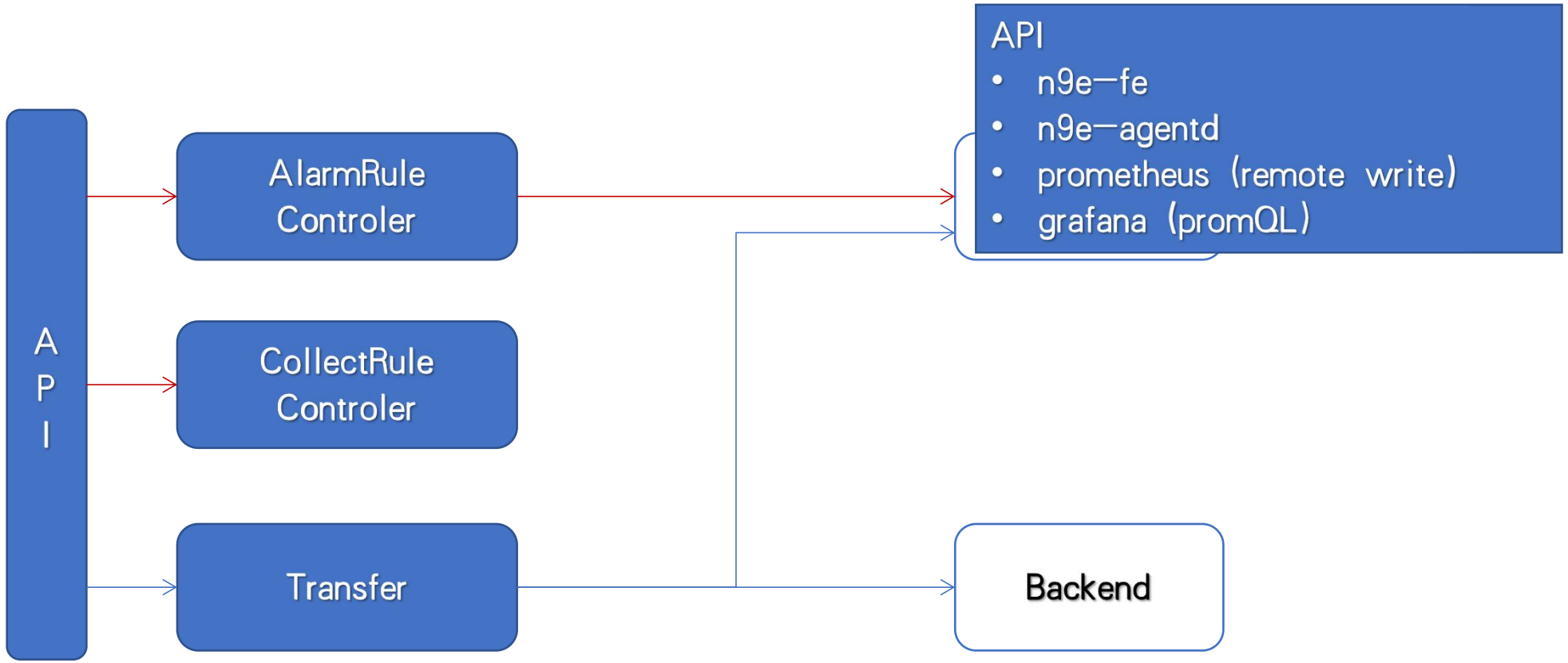
第五部分

夜莺设计实现 Server 数据处理

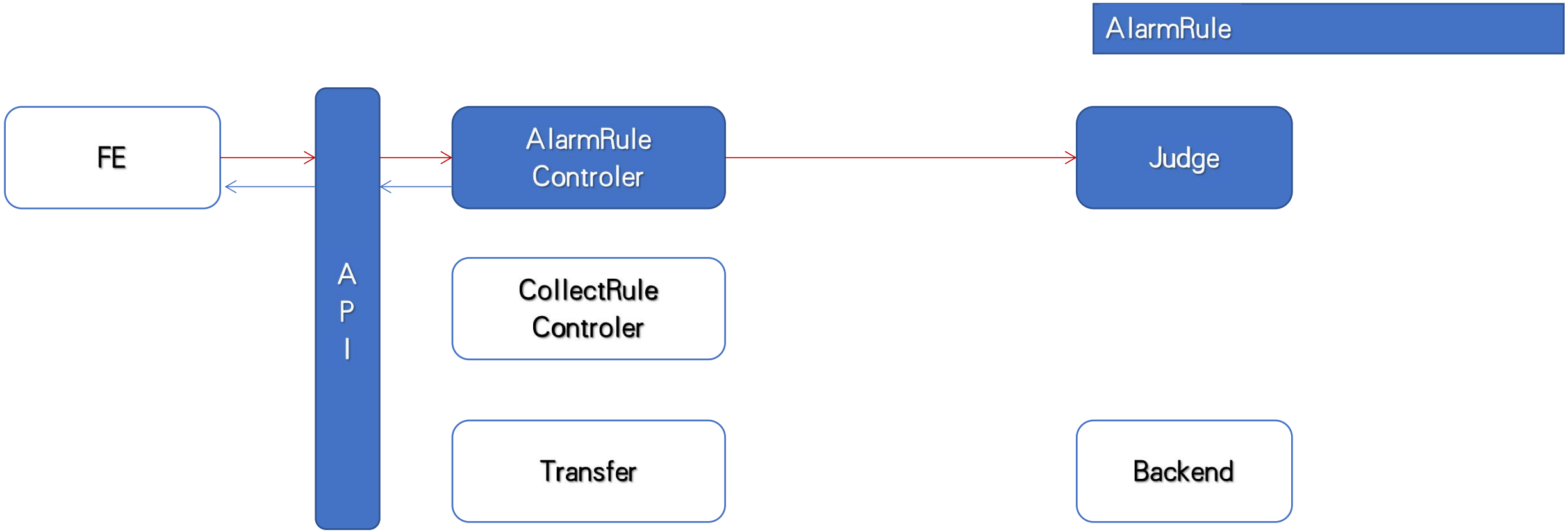
夜莺Server数据处理



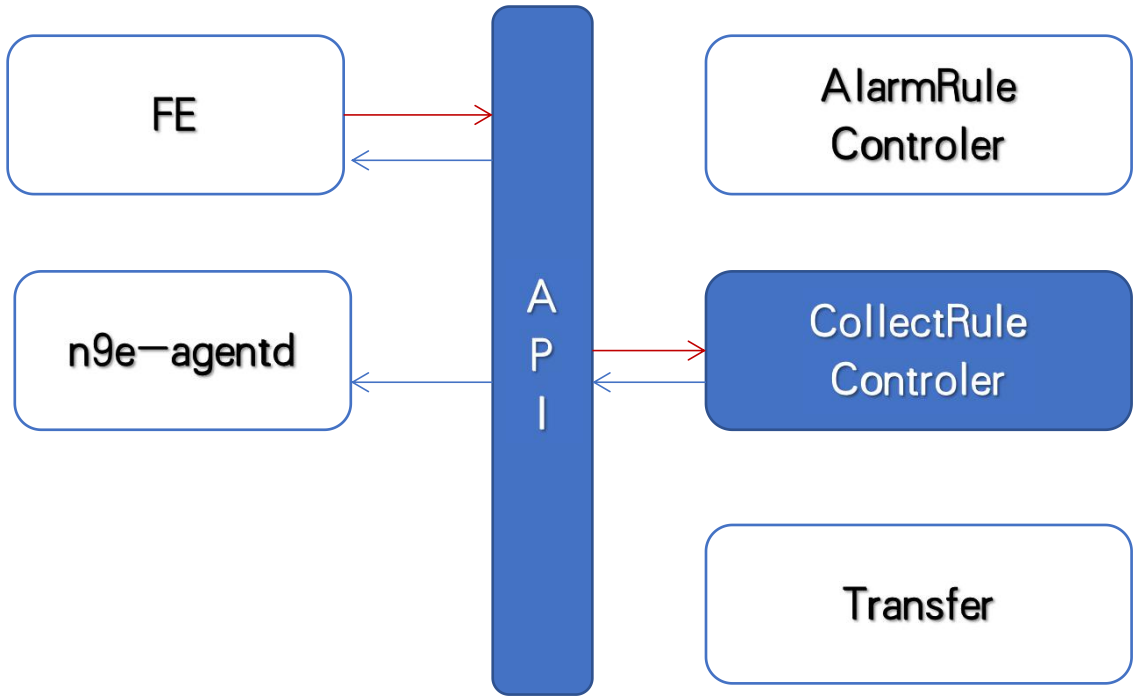
夜莺Server数据处理



夜莺Server数据处理



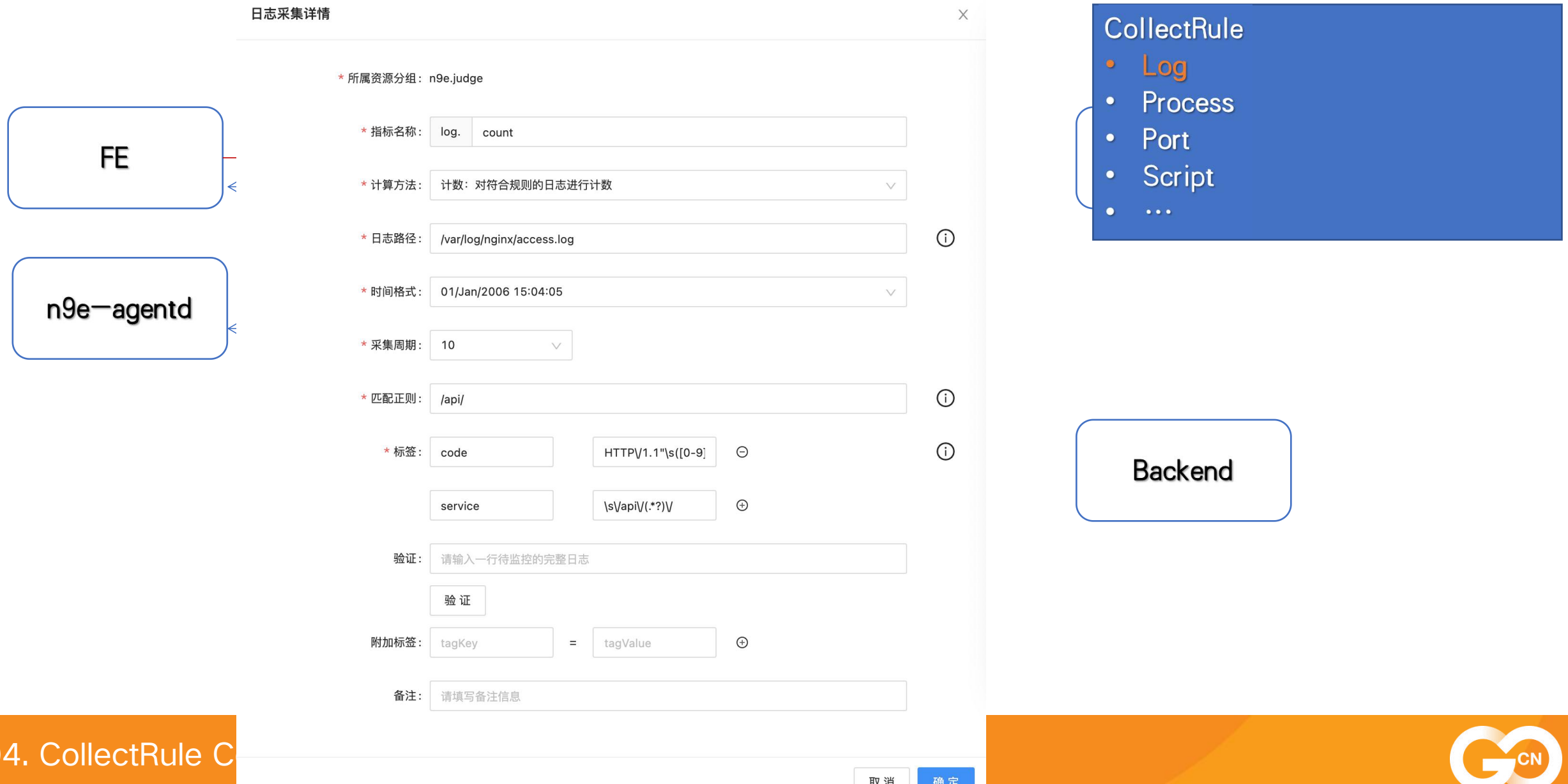
夜莺Server数据处理



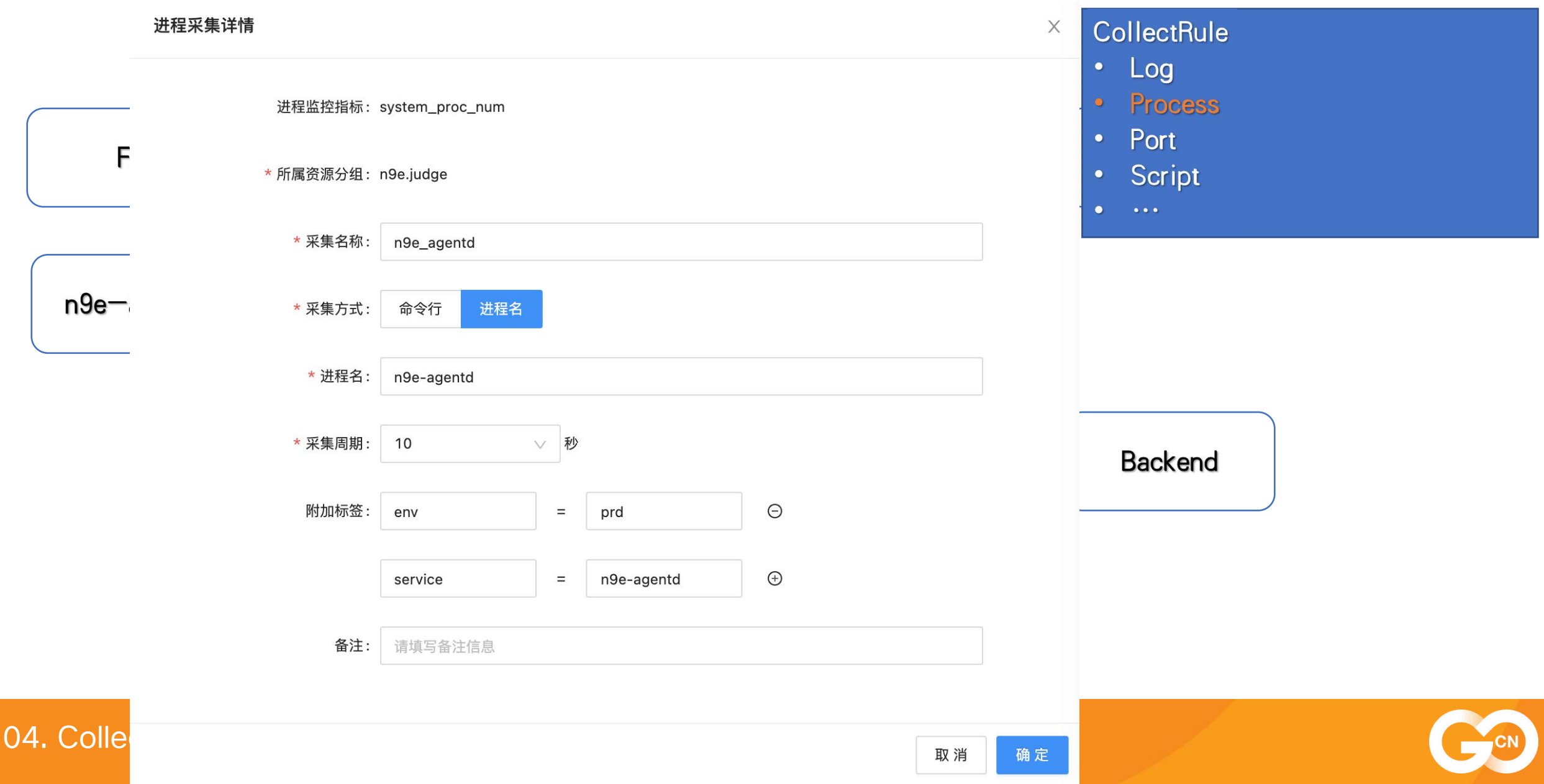
CollectRule
- Log
- Process
- Port
- Script
- …

Backend

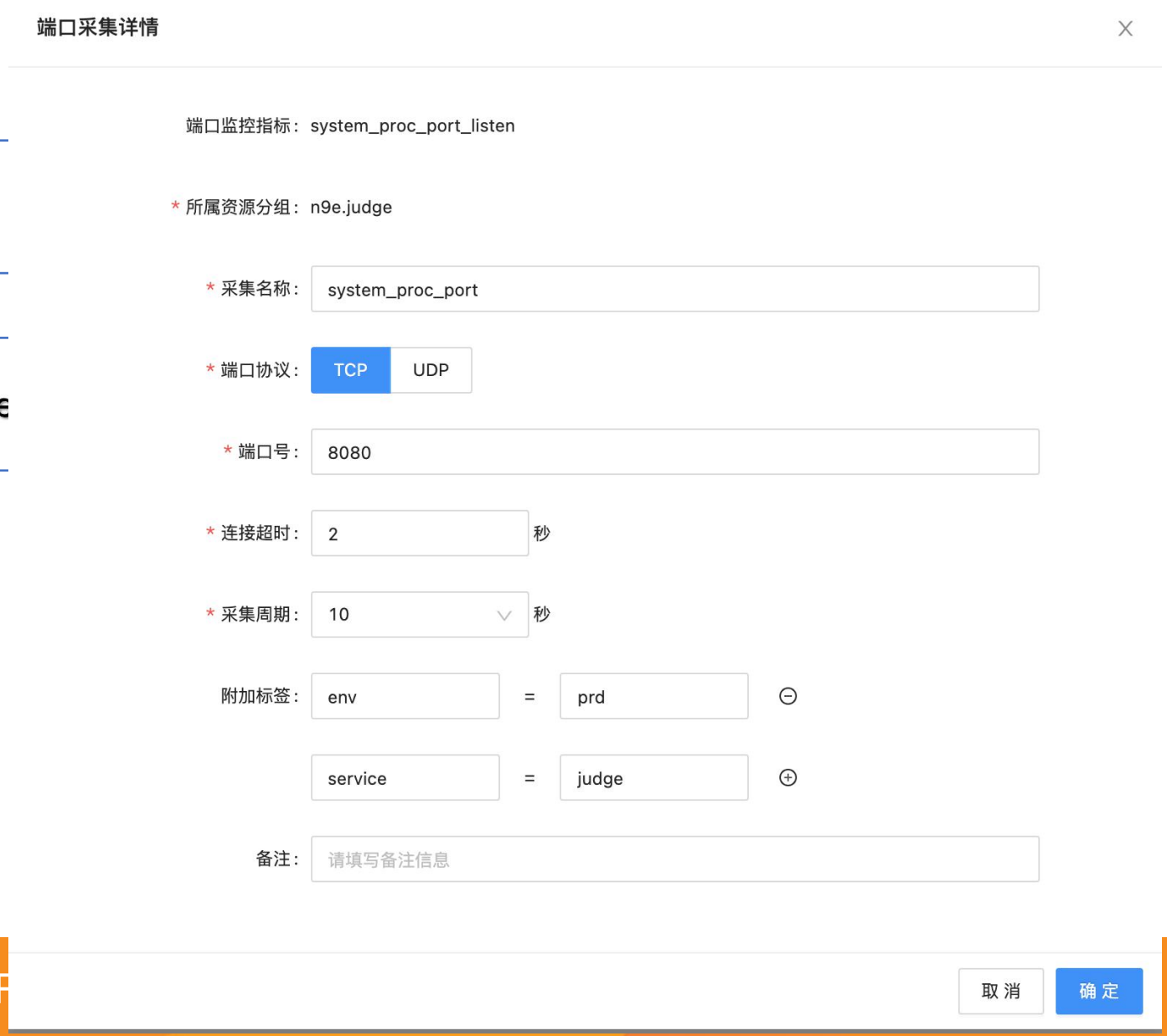
夜莺Server数据处理



夜莺Server数据处理



夜莺Server数据处理



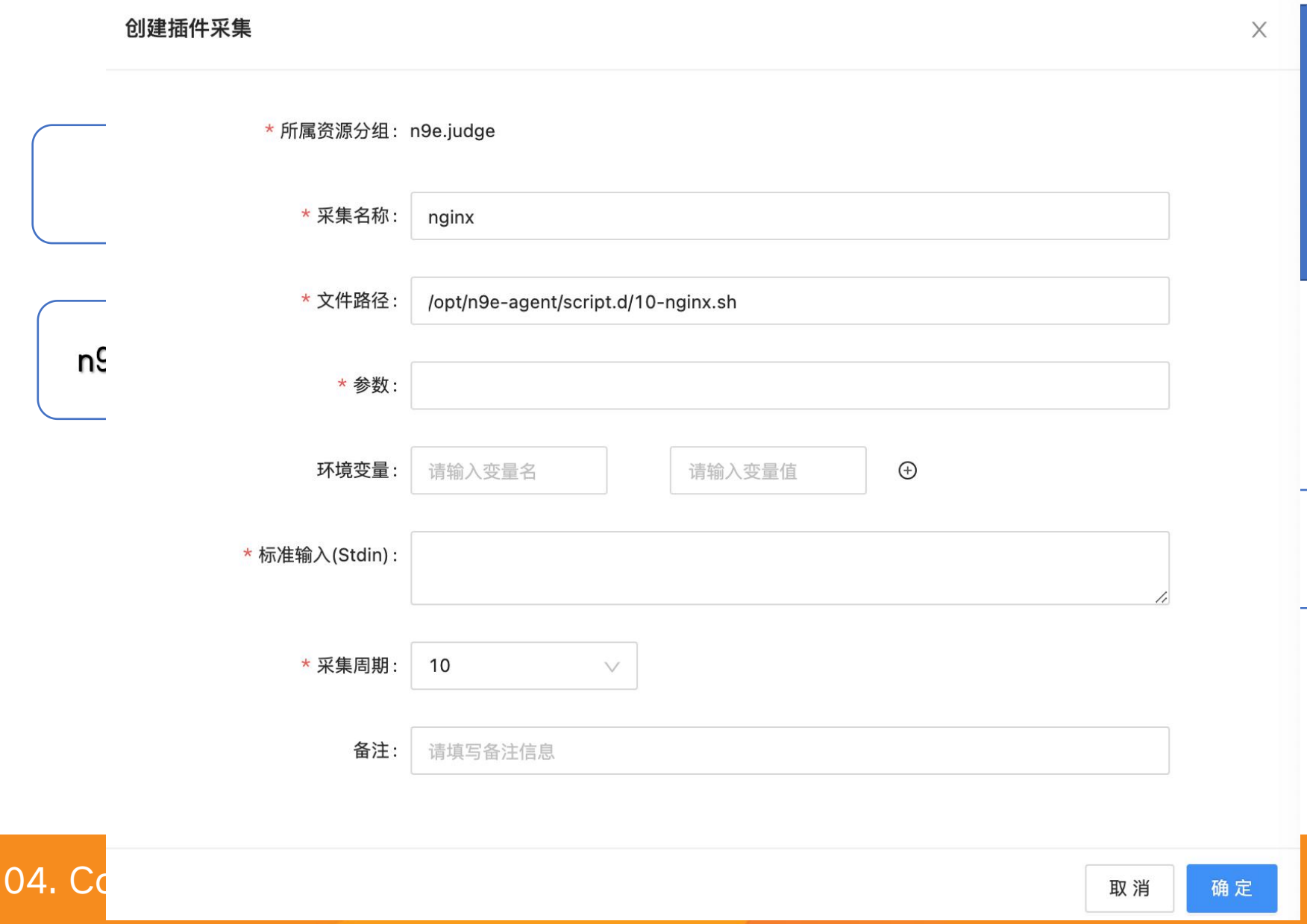
FE

n9e-age

CollectRule
- Log
- Process
- Port
- Script
- ...

Backend

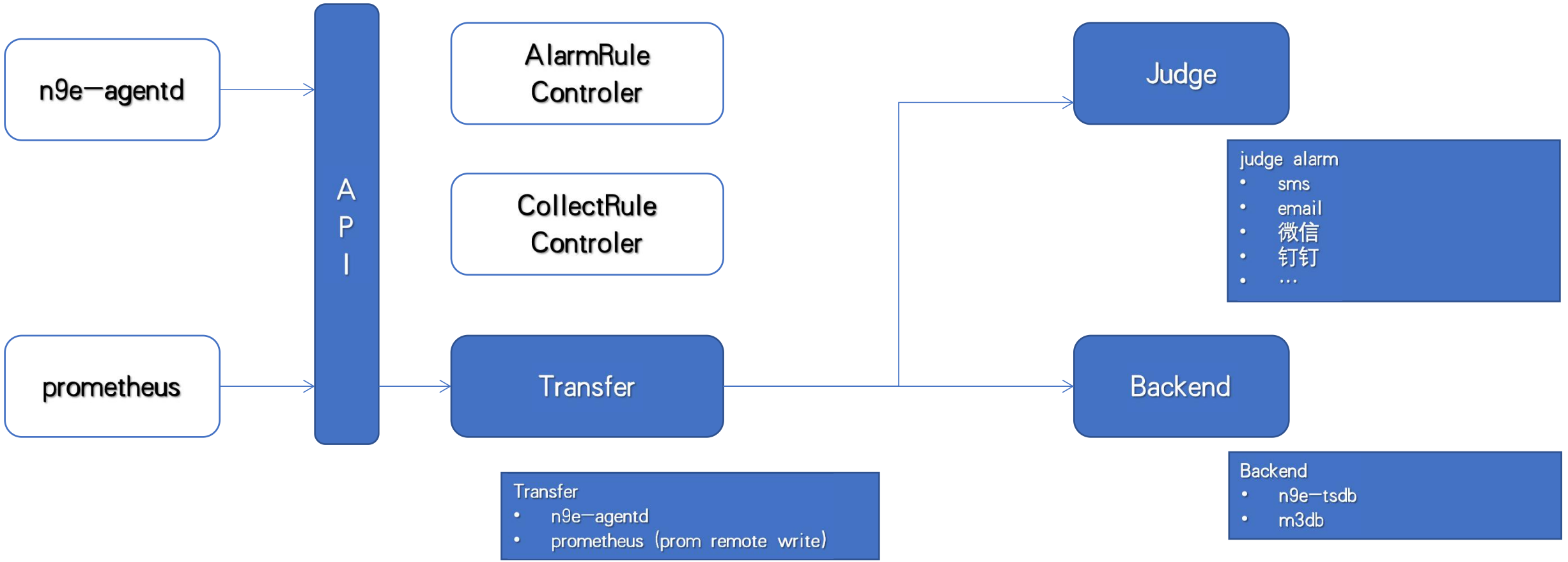
夜莺Server数据处理



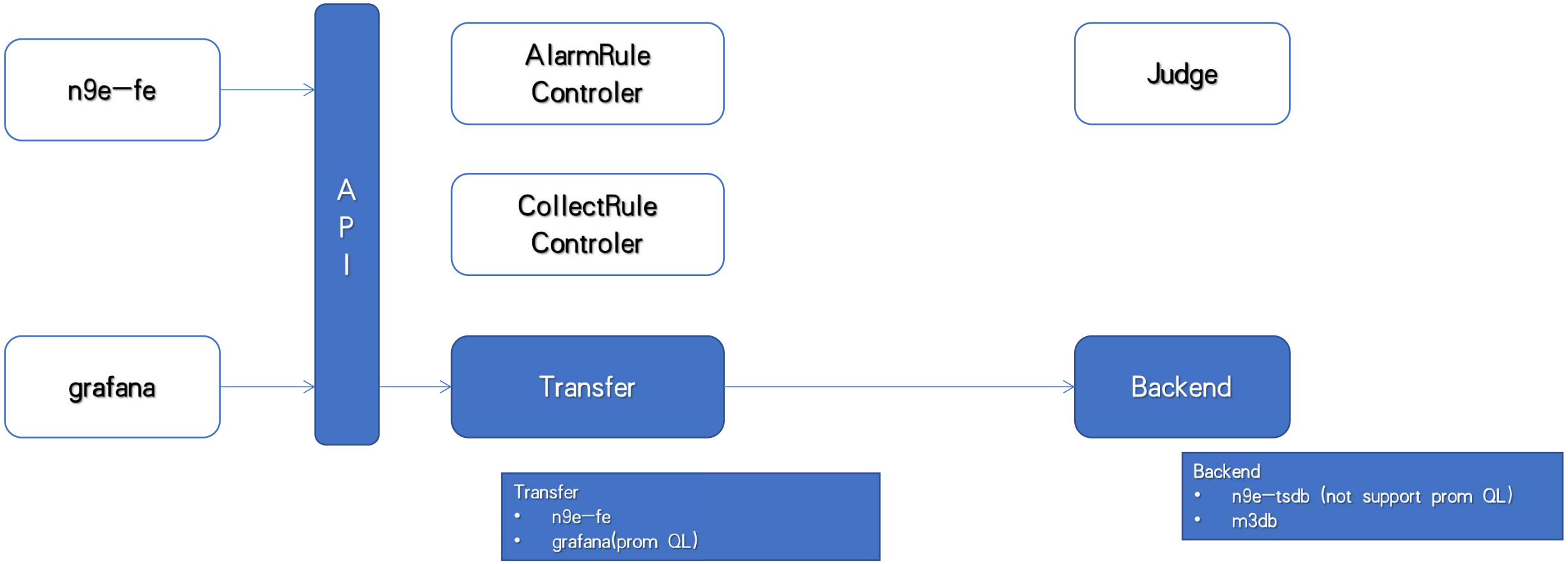
CollectRule
- Log
- Process
- Port
- Script
- …

Backend

夜莺Server数据处理



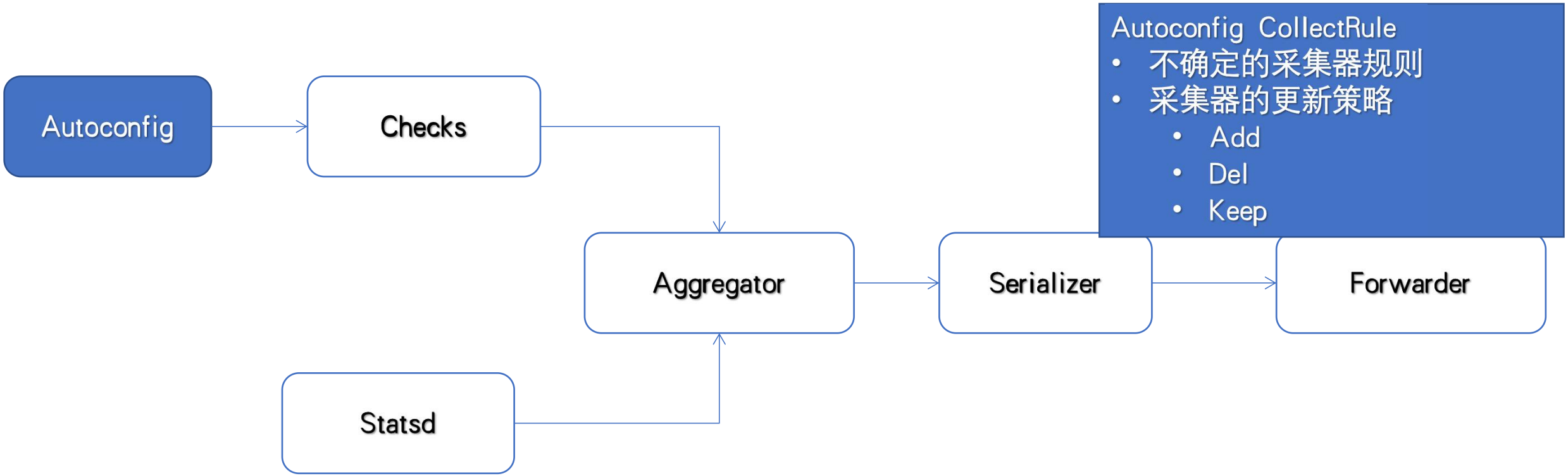
夜莺Server数据处理



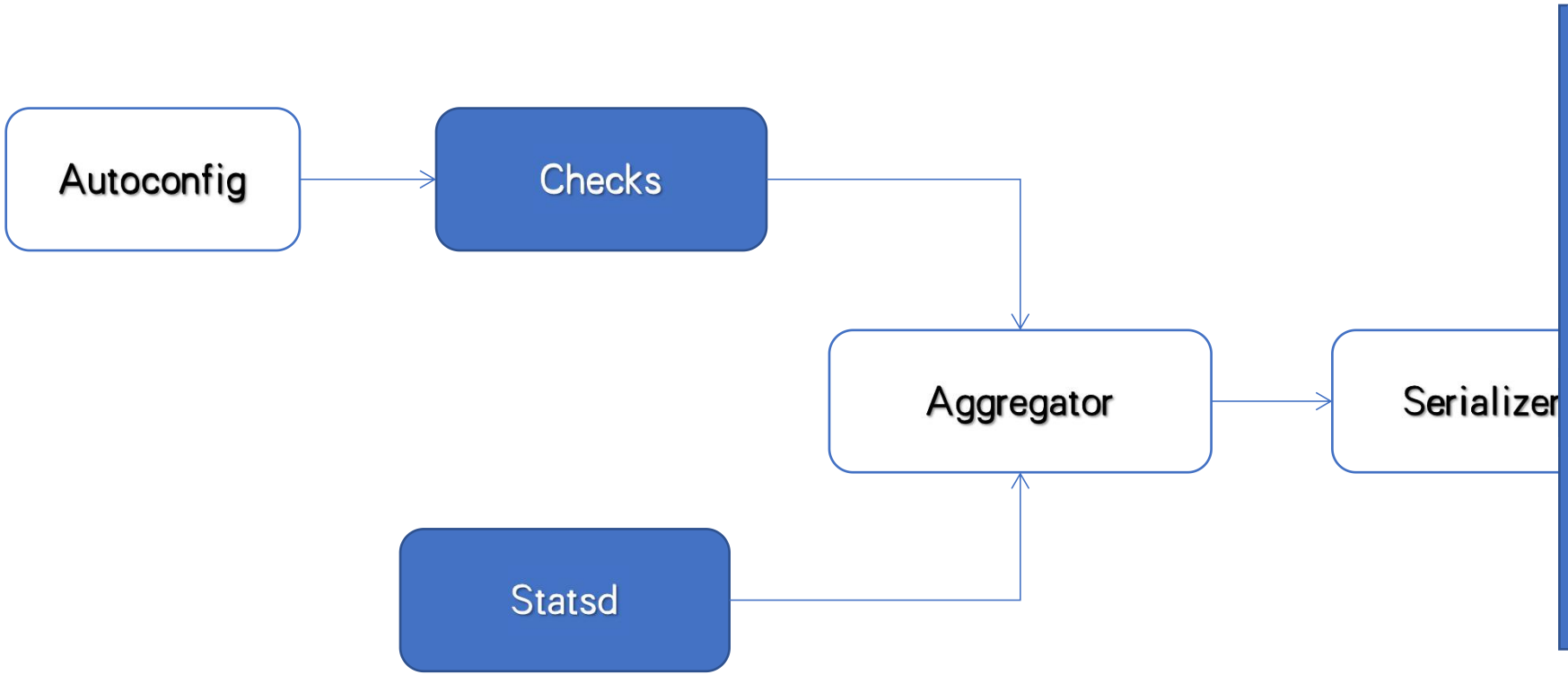
第六部分

夜莺设计实现 技术难点及细节

夜莺　技术难点及细节


Autoconfig
Checks
Aggregator
Serializer
Forwarder
Statsd
Autoconfig CollectRule
• 不确定的采集器规则
• 采集器的更新策略
• Add
• Del
• Keep

夜莺　技术难点及细节

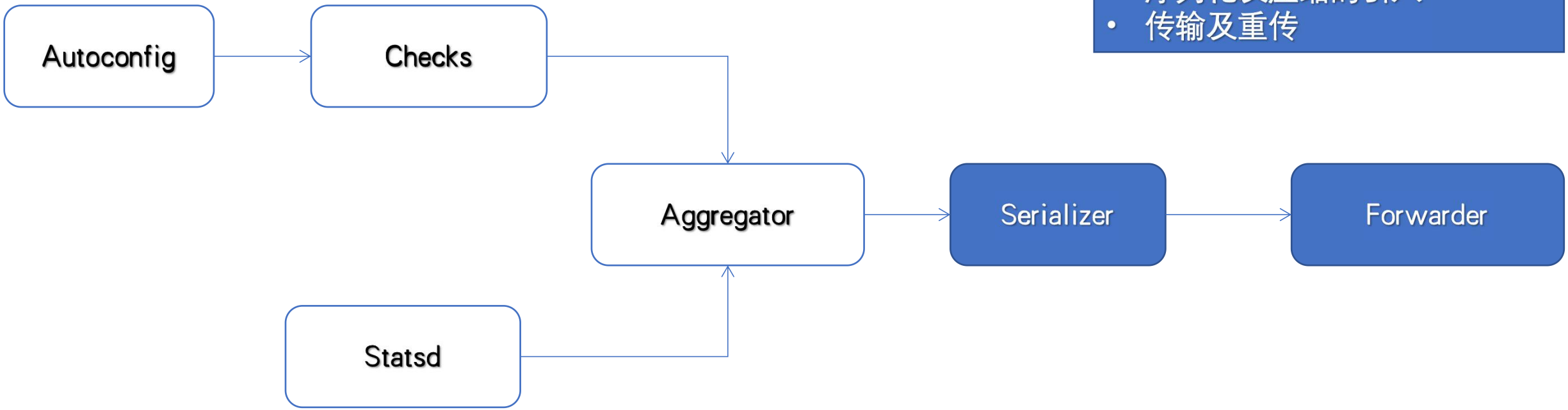


采集器

- 跨平台
  - Cpu
  - Process
- 跨语言
  - Python
  - Ruby
  - Go
  - Java
  - PHP
  - .NET

夜莺　技术难点及细节

数据序列化及传输问题
- 序列化及压缩的引入
- 传输及重传



Thank you