РАЗРАБОТКА ПЛАГИНОВ ДЛЯ ZABBIX AGENT 2

Вадим Ипатов
ZABBIX Разработчик команды интеграции

zabbix

Способы расширения функционала Zabbix Agent

Что за новый агент?

- Написан на Go
- Простота написания плагинов
- Обратная совместимость
- Постоянные соединения
- Сохранение состояния
- Может принимать трапы
- Конфигурация
- Логирование

- Плагины “из коробки”: PostgreSQL, MySQL, Redis, Memcached, Docker

Что за новый агент?

- Production ready, начиная с Zabbix 5.0!
- Доступен на Linux, Windows и MacOS

Зачем это всё?

Zabbix Agent - PostgreSQL

✔ Zabbix Agent

github.com/sergiotocalini/zapgix

share.zabbix.com/zapgix

Module for monitoring PostgreSQL

ibzbxpgsql (Lib-Zabbix-PostgreSQL) provides detailed and granular monitoring of PostgreSQL servers using a native Zabbix agent module

✔ Database Monitor

cavaliercoder.com/libzbxpgsql/

share.zabbix.com/postgresql-monitoring-for-zabbix

Monitoring agent for PostgreSQL

Tool for monitoring PostgreSQL with wide range of features

✔ Zabbix Agent Active

github.com/postgrespro/mamonsu

share.zabbix.com/mamonsu-monitoring-agent-for-postgresql

PostreSQL monitoring

Supports PostgreSQL 9.6+

✔ Low Level Discovery (LLD) ✔ Zabbix Agent Active

www.linuxelite.com.br/monitoramento/monitorando-postgresql-9-6-no-zabbix/

share.zabbix.com/postgresql-9-9

PosgreSQL partitioning

Scripts for partitioning postgresql

✔ DB Patch

github.com/cavaliercoder/zabbix-pgsql-partitioning

share.zabbix.com/postgresql-partitioning-scripts

PostgreSQL monitoring template for Zabbix

pg_monz enables various types of monitoring of PostgreSQL such as alive, resource, performance, etc. It supports some constitution patterns which includes single PostgreSQL pattern, HA pattern with Streaming Replication and load balancing pattern with pgpool-II

✔ LLD ✔ Zabbix Agent

pg-monz.github.io/pg_monz/index-en.html

share.zabbix.com/postgresql

PostgreSQL monitoring with Zabbix

The Template DB PostgreSQL was formerly known as pgCayenne. It is a set of UserParameter for PostgreSQL monitoring, which consists of Zabbix agent configuration and XML Template for web monitoring. It is using built-in PostgreSQL system views and functions. The agent requires the standard psql client ...

github.com/lesovsky/zabbix-extensions/tree/master/files/postgresql

Zabbix plugin to monitor RDBMS

ZabbixDBA is fast, flexible, and continuously developing plugin to monitor your RDBMS. ZabbixDBA uses threading of DBI connections which is good for monitoring of multiple database instances simultaneously. Just configure it, run daemon and it will do all the job

github.com/netrusov/ZabbixDBA

Zabbix PostgreSQL scripts

Scripts for partitioning the Zabbix database on PostgreSQL.

github.com/cavaliercoder/zabbix-pgsql-partitioning

PostgreSQL partitioning

Scripts to install, manage, and remove PostgreSQL partitioning for Zabbix DB

github.com/robbrucks/zabbix-postgresql-auto-partitioning

Docker for postgreSQL

Docker container for Zabbix DB on postrges, automatic partitioning.

github.com/blackcobra1973/zabbix-db-postgresql

Zabbix PostgreSQL template

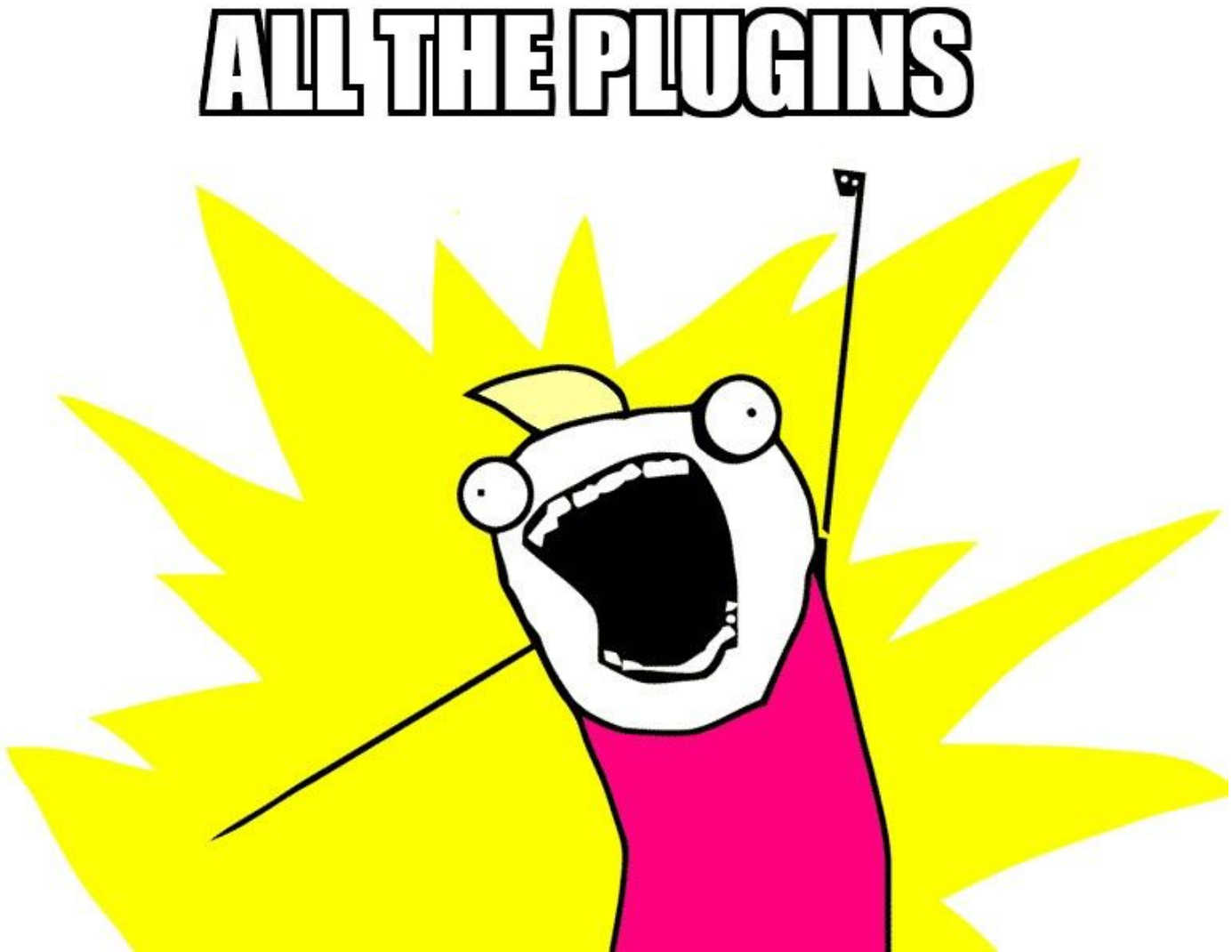
Agenless remote postgresql monitoring using zabbix

github.com/datorama/zabbix_postgresql_template

pgmon_2ndq is a set of postgresql monitoring functions implemented in the server.

The functions are installed in their own database and connect back to all databases in the server to get database specific infoA cronjob gets the monitoring info from these functions in a single database call and saves it to a file

github.com/postsql/pgmon_zabbix


ALL THE PLUGINS

```
package packageName

import "zabbix.com/pkg/plugin"

type Plugin struct {
    plugin.Base
}

var impl Plugin

func (p *Plugin) Export(key string, params []string, ctx
plugin.ContextProvider) (res interface{}, err error) {
    // Write your code here
    return
}

func init() {
    plugin.RegisterMetrics(&impl, "PluginName", "key", "Description.")
}
```

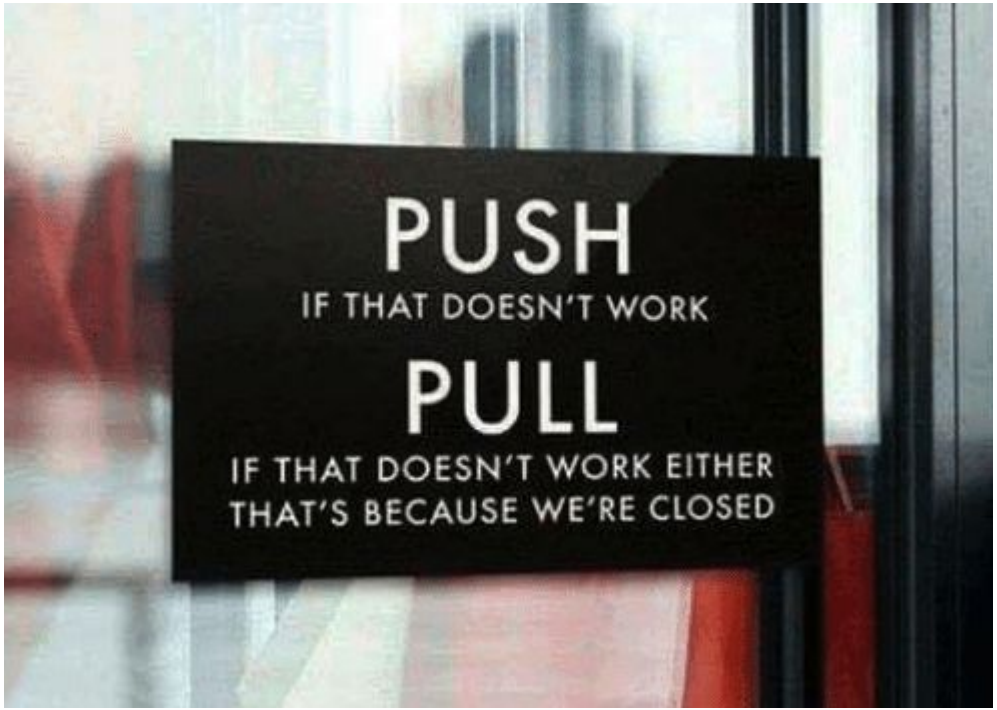
Типы плагинов

Внутренние: go/internal/agent/plugin_*.go

Внешние: go/plugins

Интерфейсы

- **Exporter** → pull
- **Watcher** → push
- **Collector**



ZABBIX

plugin.Exporter

```
// Exporter - interface for exporting collected metrics
type Exporter interface {
  // Export method exports data based on the key 'key' and
  // its parameters 'params'.
  Export(key string, params []string,
  context ContextProvider) (interface{}, error)
}
```

plugin.Exporter

- `MaxCapacity = 100`
- `SetCapacity(capacity int)`
- `Plugins.<PluginName>.Capacity = 1`

ZABBIX

plugin.Collector

```
// Collector - interface for periodical metric collection
type Collector interface {
  Collect() error
  Period() int
}
```

plugin.Watcher

```
// Watcher - interface for fully custom monitoring
type Watcher interface {
  // Watch method instructs plugin to watch for events based on
  // item configuration in 'requests'.
  Watch(requests []*Request, context ContextProvider)
}
```

Нам **watch/manager** строить и жить помогает!

Нужно больше интерфейсов!

– – –

- **Runner**
- **Configurator**

ZABBIX

plugin.Runner

```
// Runner - interface for managing background processes
type Runner interface {
  // Start method activates plugin.
  Start()
  // Stop method deactivates plugin.
  Stop()
}
```

ZABBIX

plugin.Configurator

```
// Configurator - interface for plugin configuration
// in agent conf files
type Configurator interface {
  // Configure method passes global and private plugin
  // configuration after it has been activated.
  Configure(globalOptions *GlobalOptions, privateOptions
interface{})
  // Validate method validates private plugin
  // configuration during agent startup.
  Validate(privateOptions interface{}) error
}
```

plugin.Configurator

```
KeepAlive int `conf:"optional,range=60:900,default=300"`
```

Миксуйте интерфейсы

– – –

system/cpucollector:

Collector: собирает и агрегирует данные

Exporter: отдаёт данные по запросу

Runner: [де]инициализирует структуры

ZABBIX

Hello, world!

---

1. Импортируем пакет plugin

```
package weather
import  "zabbix.com/pkg/plugin"
```

2. Определяем свою структуру, используя базовую

```
type Plugin struct {
  plugin.Base
}
var impl Plugin
```

Hello, world!

---

3. Реализуем интерфейс

```
func (p *Plugin) Export(key string, params []string,
ctx plugin.ContextProvider) (result interface{}, err error) {
  // https://github.com/chubin/wttr.in
  response, err := http.Get(fmt.Sprintf("https://wttr.in/~%s?format=%%t", params[0]))
  if err != nil {
     return nil, err
  }
  defer response.Body.Close()

  temp, _ := ioutil.ReadAll(response.Body)

  return string(temp)[0:len(temp)-4], nil
}
```

ZABBIX

Hello, world!

---

4. Регистрируем метрики

```
func init() {
  plugin.RegisterMetrics(&impl, "Weather", "weather.temp", "Returns
Celsius temperature.")
}
```

```
// impl — указатель на реализацию плагина
// name — имя плагина с заглавной буквы
// params — список метрик и описаний для них (key1, descr1, key2, descr2, keyN, descrN...)
func RegisterMetrics(impl Accessor, name string, params ...string)
```

Hello, world!

5. Добавляем плагин в список импортов в файлы plugins_<platform>.go

```
package plugins

import (
  _ "zabbix.com/plugins/kernel"
  _ "zabbix.com/plugins/log"
// ...
  _ "zabbix.com/plugins/weather"
)
```

Сборка

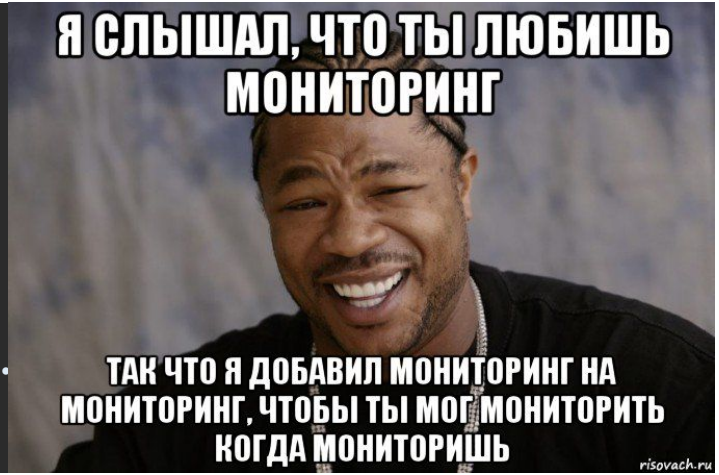
```
$ cd <zabbix-source>
$ ./bootstrap.sh; ./configure --enable-agent2; make install
```

That’s it!

```
$ zabbix_agent2 -t weather.temp[riga]
+20
```

Мониторинг мониторинга

```
$ zabbix_agent2 -R metrics
...
[Weather]
active: true
capacity: 0/100
tasks: 0
weather.temp: Returns Celsius temperature.
...
```



Что дальше?

- Загружаемые плагины
- Обновление конфигурации в рантайме
- Больше плагинов “из коробки” от команды Zabbix

Что ещё почитать/посмотреть?

- Templates guidelines
- An official guide to making and managing great templates
- Официальная документация
- Исходники плагинов на git.zabbix.com (src/go/plugins/)
- Writing watcher Zabbix Agent2 MQTT plugin in Go

СПАСИБО
ЗА ВНИМАНИЕ!

Вадим Ипатов
ZABBIX Разработчик команды интеграции

zabbix