Мне довольно часто приходится работать с нодами различных криптовалют, о чем я уже писал в отдельной статье. Сегодня хочу рассказать, как я настраиваю мониторинг нод (эфир, биткоин и др.) и конкретно состояние блокчейна с помощью zabbix. Если вам интересна эта тема, приглашаю к дальнейшему ознакомлению.

Содержание:

Введение

Для тех, кому интересно, как устанавливать и настраивать ноды популярных криптовалют, есть отдельная статья на этот счет — установка и настройка нод. Я расскажу, как мониторить состояние этих нод. Будут показаны 2 разных способа:

1. Мониторинг нод Ethereum (Эфира).
2. Мониторинг нод Bitcoin, Litecoin, Dash, Bitcoin Cash и т.д.

Последние ноды очень похожи между собой. Используют одни и те же консольные команды cli, вывод о состоянии блокчейна у них одинаковый. Они все по сути клоны биткоина, поэтому и мониторятся так же, как он. Ноды Эфира работают совсем по-другому, там и мониторинг другой.

Настраивать мониторинг нод криптовалют я будут с помощью системы мониторинга zabbix. Сразу делаю важное замечания. Предложенные в мониторинге скрипты и шаблоны сделаны для внутреннего использования. Я не приводил их к красивому и удобному виду, не оптимизировал код, не искал наиболее оптимальные варианты. Была задача настроить мониторинг и я решал ее сходу, с помощью тех идей, которые приходили в голову. В большей части это были единичные задачи, поэтому тратить время на оптимизацию и улучшения не было смысла. Важно, чтобы все работало так, как надо.

Сам по себе мониторинг криптонод очень примитивный. Надо просто парсить вывод команд о состоянии ноды и блокчейна. Вот и все. Делать это можно разными способами, кто как привык и умеет. Я сделал так, как умею сам, не претендуя ни на какие лавры. Просто делюсь с вами своими наработками. В сети я ничего подобного не видел, поэтому все с нуля делал сам.

Версия zabbix сервера, на котором все это работает — 4.0. Шаблоны и скрипты будут взяты из реальных работающих проектов, так что они 100% рабочие и проверенные.

Если у вас еще нет своего сервера для мониторинга, то рекомендую материалы на эту тему. Для тех, кто предпочитает систему CentOS:

1. Установка CentOS 7.
2. Настройка CentOS 7.
3. Установка и настройка zabbix сервера.

То же самое на Debian 9, если предпочитаете его:

1. Установка Debian 9.
2. Базовая настройка Debian 9.
3. Установка и настройка zabbix на debian.

Мониторинг ноды Ethereum — Эфира

Для начала привожу список параметров, которые я буду мониторить:

1. **eth_blockNumber** — число блоков в блокчейне. Если нода выдает 0, то срабатывает триггер.
2. Наличие запущенного сервиса ноды в системе. В моем случае используется клиент parity, мониторить буду его — **запущен или нет процесс**.
3. **eth_syncing** — состояние синхронизации. С его помощью можно узнать, не отстает ли нода в синхронизации блоков. Если отставание большое, срабатывает триггер.
4. Открытый tcp порт для rpc запросов. Если порт не отвечает, срабатывает триггер.

Расскажу подробно, как реализована каждая проверка. Пойдем по списку. Для проверки eth_blockNumber есть отдельный метод, который запускается вот так:

```
# curl -X POST -H "Content-Type: application/json" --data '{"jsonrpc":"2.0","method":"eth_blockNumber","params":[],"id":83}'
localhost:8545
```

Проверку можно делать как локально, так и удаленно. Я в данном случае все проверки буду делать локально, чтобы потом передавать информацию в zabbix-agent. Вывод будет примерно такой:

```
{"jsonrpc":"2.0","result":"0x65a0d9","id":83}
```

Нас интересует значение result. К сожалению, оно в 16-ти ричном формате. Я написал небольшой скрипт *eth-get.sh*, который парсит это значение и преобразует его в 10-ти ричный формат.

```
#!/bin/bash
curl -X POST -H "Content-Type: application/json" --data '{"jsonrpc":"2.0","method":"eth_blockNumber","params":[],"id":83}'
localhost:8545 | cut -f3 -d : | cut -f2 -d'"' > /etc/zabbix/scripts/result-hex.txt
a=`cat /etc/zabbix/scripts/result-hex.txt`
echo $(( ${a} )) > /etc/zabbix/scripts/result-dex.txt
```

Результат записывается в текстовый файл, а не выводится сразу из-за того, что в вывод попадает какая-то информация от curl. Уже не помню подробностей, но по-моему не получалось избавиться от ее прогресс бара, даже явно запрещая его ключами. В итоге пришлось записать значение в файл, а потом вторым скриптом выводить это значение и передавать в zabbix сервер. Второй скрипт *eth-chek.sh*

```
#!/bin/bash
```

```
a=`cat /etc/zabbix/scripts/result-dex.txt 2>/dev/null`
echo $a
```

После работы этого скрипта, в консоль должно выводиться только число с номером блока. Сразу скажу, что сделано очень костыльно. Мне, когда я делал, просто не пришло в голову, что zabbix умеет принимать сразу готовый json. Далее есть возможность сделать зависимый элемент и в нем же перевести 16-ти ричный формат в 10-ти ричный. Когда я делал данную проверку, не знал об этом. Сейчас по работе с json в zabbix у меня есть отдельная статья, где рассказано, как сделать то, что я описал быстро и красиво. Мониторинг bitcoin, который описан ниже, тоже сделан весь через парсинг json полностью на стороне zabbix server.

Следующим этапом я проверяю наличие службы parity в системе. Это делается с помощью встроенного функционала заббикса прямо в шаблоне. В агенте ничего настраивать не надо.

Дальше мониторим синхронизацию. Здесь все посложнее. Информация о синхронизации проверяется следующей командой:

```
curl --data '{"method":"eth_syncing","params":[],"id":1,"jsonrpc":"2.0"}' -H "Content-Type: application/json" -X POST
localhost:8545
```

Если все ОК, то вывод будет такой:

```
{"jsonrpc":"2.0","result":false,"id":1}
```

Нас интересует значение **result**. В данном случае **false** это нормальное состояние, когда все в порядке. Если идет процесс синхронизации, то вывод будет примерно такой:

```
{"id": 1,"jsonrpc": "2.0","result": {"startingBlock": "0x384","currentBlock": "0x386","highestBlock": "0x454"}}
```

Здесь нас интересуют значения **currentBlock** и **highestBlock**. Если разница между последним и первым более 50, считаем, что синхронизация отстает и срабатывает триггер. Здесь мне уже очень хотелось все отправить в zabbix сервер и распарсить там, но я не придумал, как это сделать. Формат json строки меняется. Пришлось опять все делать на клиенте скриптом и отправлять на сервер мониторинга уже подготовленные данные. Вот сам скрипт *eth_syncing.sh*:

```
#!/bin/bash

curl --data '{"method":"eth_syncing","params":[],"id":1,"jsonrpc":"2.0"}' -H "Content-Type: application/json" -X POST
localhost:8545 -o /etc/zabbix/scripts/eth_syncing.txt 2>/dev/null

if grep -q "false" /etc/zabbix/scripts/eth_syncing.txt; then
   echo "1"
else
    currentblock=`cat /etc/zabbix/scripts/eth_syncing.txt | cut -f4 -d : | cut -f1 -d , | cut -f2 -d'"'`
    highestblock=`cat /etc/zabbix/scripts/eth_syncing.txt | cut -f5 -d : | cut -f1 -d , | cut -f2 -d'"'`
    c=`echo $(( ${currentblock} ))`
    h=`echo $(( ${highestblock} ))`
    a=$(($h-$c))
    if [ $a -gt 50 ]
     then echo 0
    else
     echo 1
    fi
```

Скрипт парсит вывод. Если в выводе есть false, выводит в консоль 1, что в шаблоне будет означать OK. Дальше считает разницу между highestblock и currentblock. Если больше 50, то выводит 0. В шаблоне на это настроен триггер. Если разница меньше 50, то выводит 1, что означает OK.

Это что касается скриптов. Теперь создаем конфиг для заббикса с UserParameter. Для этого создаем файл */etc/zabbix/zabbix_agentd.d/eth.conf* следующего содержания.

```
UserParameter=eth_blockNumber,/etc/zabbix/scripts/eth-check.sh
UserParameter=eth_syncing.json,curl --data '{"method":"eth_syncing","params":[],"id":1,"jsonrpc":"2.0"}' -H "Content-Type:
application/json" -X POST localhost:8545 2>/dev/null
UserParameter=eth_syncing.status,/etc/zabbix/scripts/eth_syncing.sh
```

Первый и третий параметры передают вывод описанных скриптов. Второй передает полный json синхронизации на сервер. Это просто для информации, чтобы было проще продебажить момент работы последнего скрипта. Значение 0 или 1 не очень информативно, рядом будет полный json вывода.

После добавления параметров в zabbix-agent перезапустите его и проверьте в консоли, что ключи выводят правильные значения.

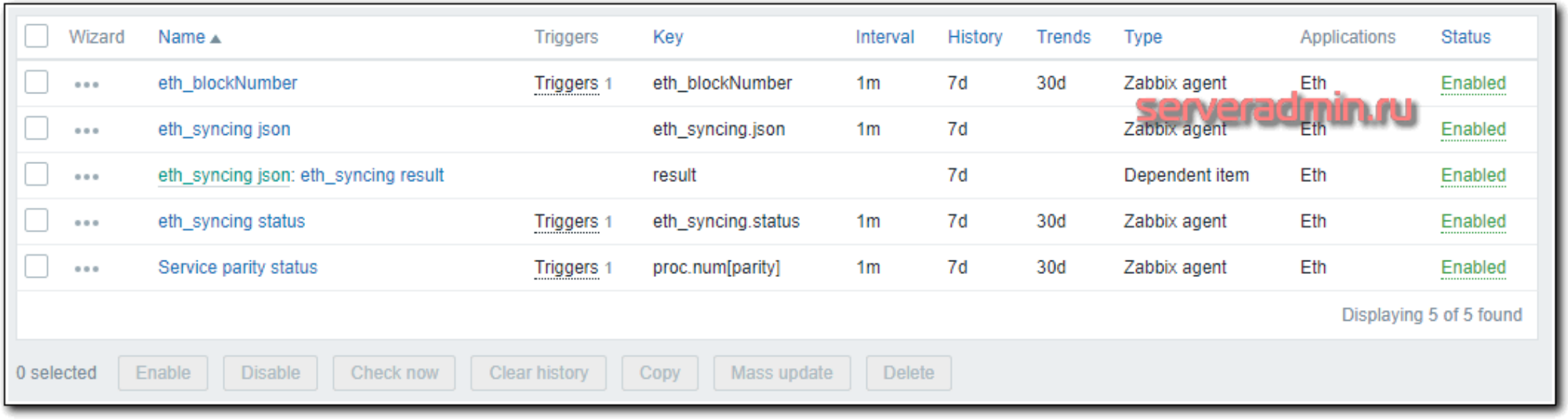
```
# zabbix_agentd -t eth_blockNumber
eth_blockNumber                               [t|4972806]
```

```
# zabbix_agentd -t eth_syncing.json
eth_syncing.json                              [t|{"jsonrpc":"2.0","result":false,"id":1}]
```

```
# zabbix_agentd -t eth_syncing.status
eth_syncing.status                            [t|1]
```

Если все в порядке и цифры реальные, то можно перемещаться на сервер и продолжать настройку там. Я подготовил шаблон для мониторинга Ethereum на основе описанных настроек — Cryptonode ETH check

Список итемов из шаблона.

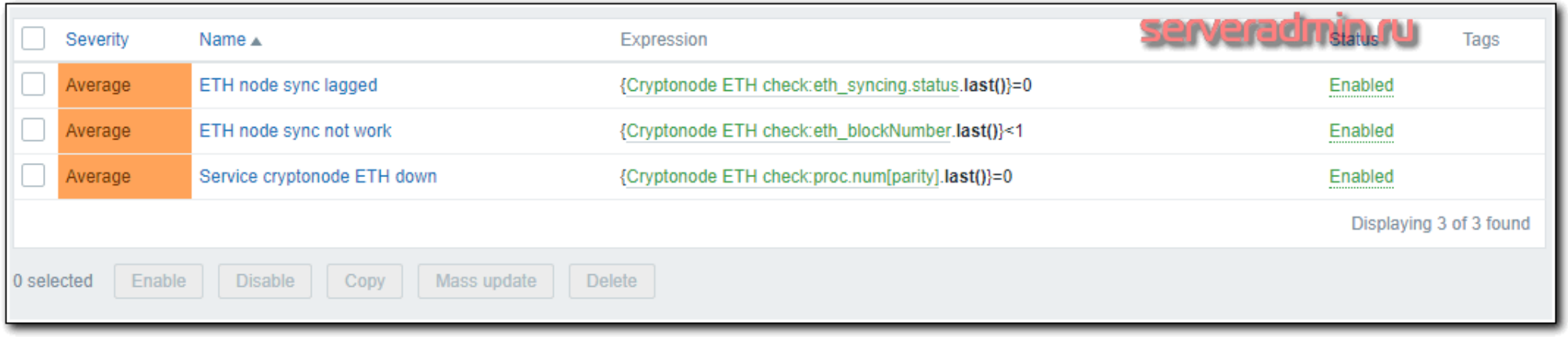


Wizard	Name ▲	Triggers	Key	Interval	History	Trends	Type	Applications	Status
	eth_blockNumber	Triggers 1	eth_blockNumber	1m	7d	30d	Zabbix agent	Eth	Enabled
	eth_syncing json		eth_syncing.json	1m	7d		Zabbix agent	Eth	Enabled
	eth_syncing json: eth_syncing result		result		7d		Dependent item	Eth	Enabled
	eth_syncing status	Triggers 1	eth_syncing.status	1m	7d	30d	Zabbix agent	Eth	Enabled
	Service parity status	Triggers 1	proc.num[parity]	1m	7d	30d	Zabbix agent	Eth	Enabled

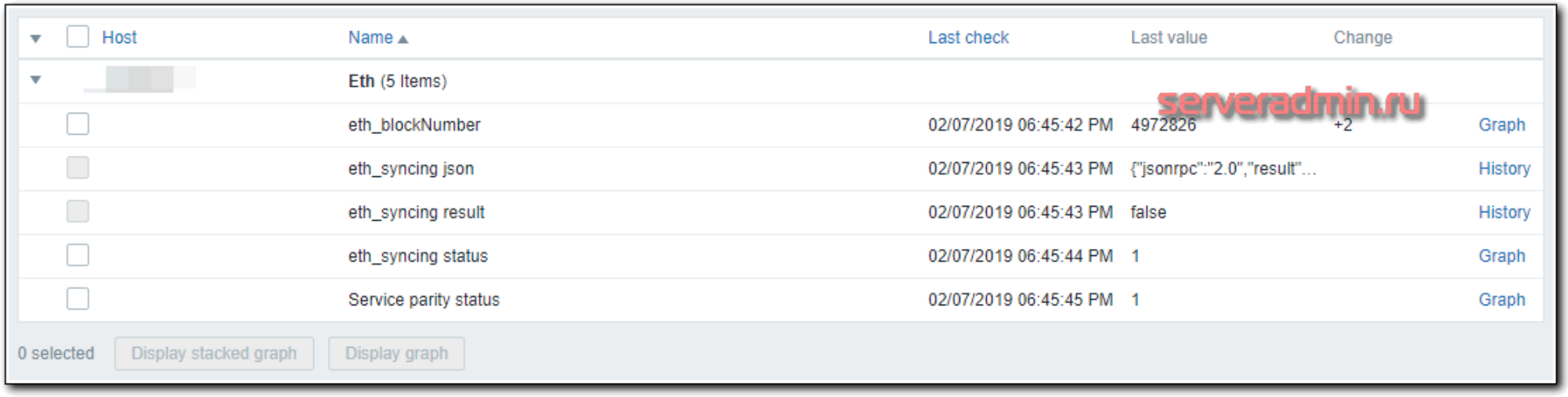
Displaying 5 of 5 found

0 selected | Enable | Disable | Check now | Clear history | Copy | Mass update | Delete

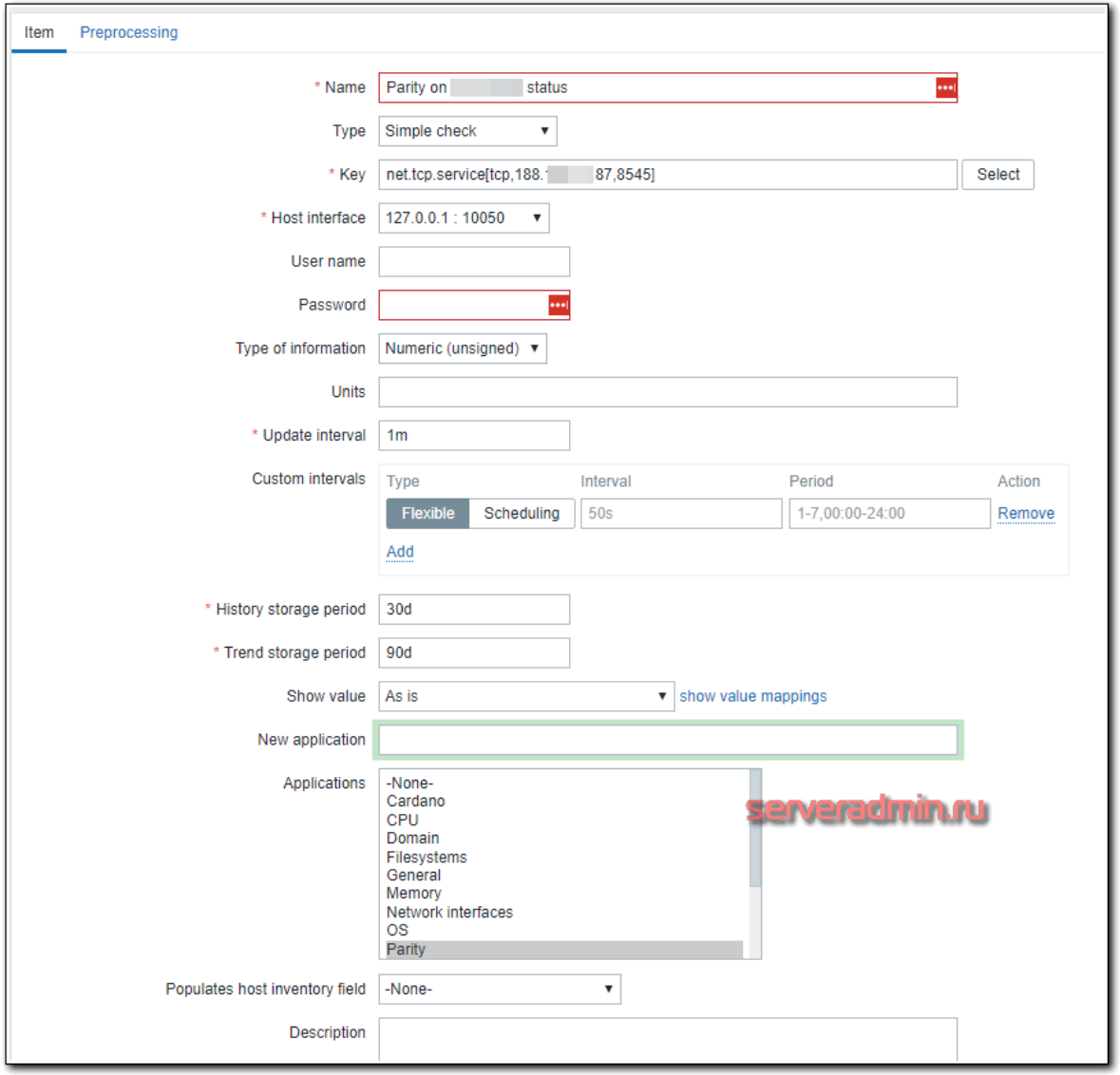
Список триггеров.



После подключения шаблона для мониторинга ноды Эфира, в Latest Data будет следующая информация.



Проверка доступности tcp порта для rpc запросов настраивается отдельно на том хосте, откуда вы хотите вести наблюдение. Я обычно с самого zabbix server настраиваю. Сделать это можно, создав вот такой элемент данных.


Item
Preprocessing
* Name
Parity on status
Type
Simple check
* Key
net.tcp.service[tcp,188.
87,8545]
Select
* Host interface
127.0.0.1 : 10050
User name
Password
Type of information
Numeric (unsigned)
Units
* Update interval
1m
Custom intervals
Type
Interval
Period
Action
Flexible
Scheduling
50s
1-7,00:00-24:00
Remove
Add
* History storage period
30d
* Trend storage period
90d
Show value
As is
show value mappings
New application
Applications
-None-
Cardano
CPU
Domain
Filesystems
General
Memory
Network interfaces
OS
Parity
serveradmin.ru
Populates host inventory field
-None-
Description

Используется **Simple check**. В поле key указывается ip адрес и порт для проверки. Подробнее об этом рассказывал в мониторинге служб linux. Не забудьте открыть на фаерволе доступ к порту со стороны zabbix сервера.

На этом про мониторинг Ethereum node все. Переходим к ноде bitcoin и производным от него.

Мониторинг bitcoin node

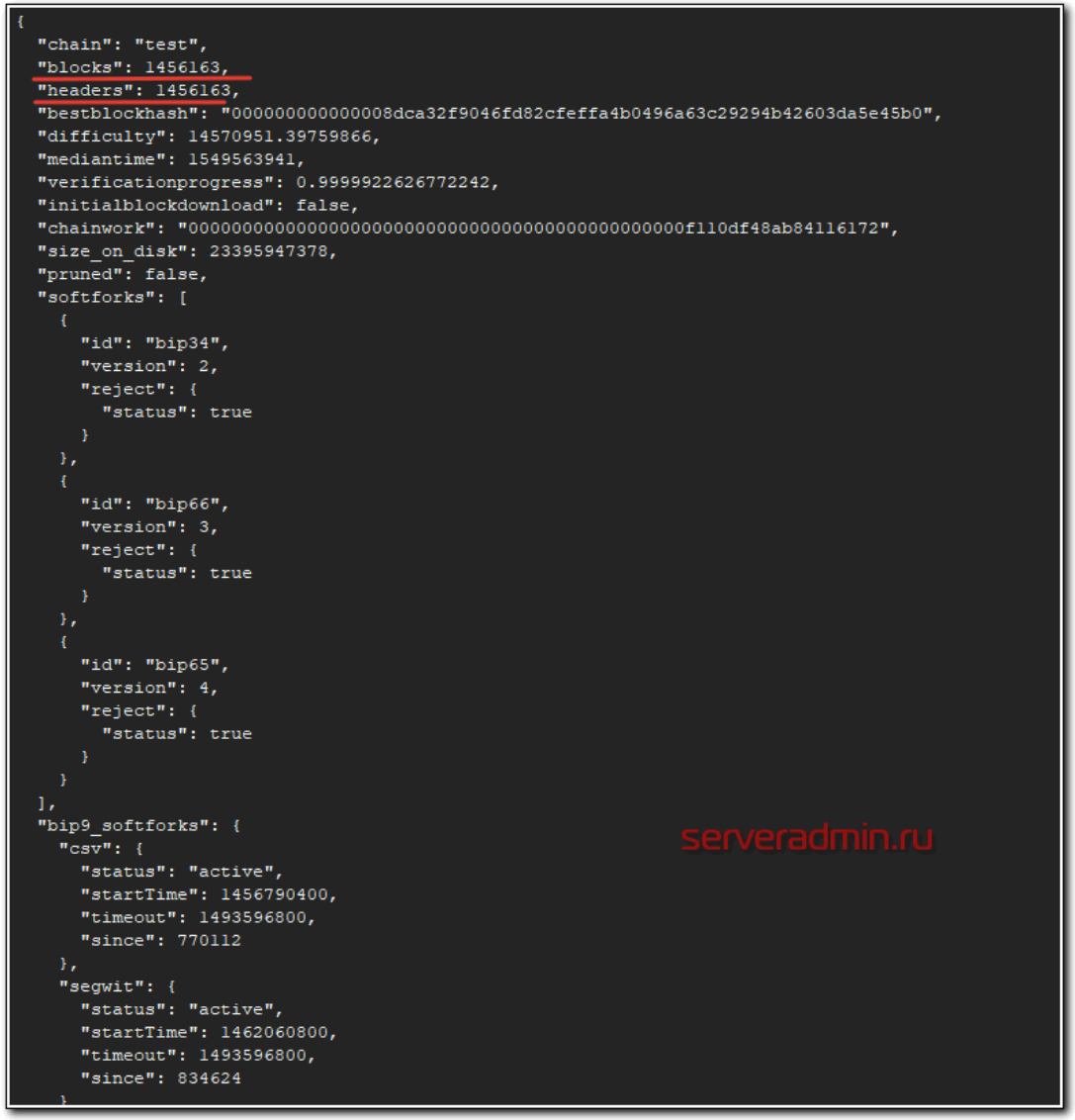
С мониторингом биткоин ноды все немного проще, так как для получения всей необходимой информации достаточно распарсить json, который не меняет свой формат в зависимости от ответа. Поэтому достаточно будет просто необходимую json строку передавать на сервер и там парсить с помощью зависимых элементов. Наблюдать будем за следующими параметрами:

1. Разница значений **headers** и **blocks**. Она не должна превышать определенного значения. Если превышает — срабатывает триггер.
2. Параметр **networkactive** должен возвращать значение **true**. Если это не так, срабатывает триггер.
3. Наличие процесса bitciond в системе.
4. Ответ на внешний запрос через rpc порт.

3 и 4 пункты сделаны точно так же, как и для ноды эфира. Не буду на этом останавливаться. Первый пункт настраивается через парсинг вывода следующей команды:

```
/usr/bin/bitcoin-cli -rpcuser=user -rpcpassword=password getblockchaininfo
```

Ответ должен быть примерно такой:


{
"chain": "test",
"blocks": 1456163,
"headers": 1456163,
"bestblockhash": "000000000000008dca32f9046fd82cfeffa4b0496a63c29294b42603da5e45b0",
"difficulty": 14570951.39759866,
"mediantime": 1549563941,
"verificationprogress": 0.9999922626772242,
"initialblockdownload": false,
"chainwork": "0000000000000000000000000000000000000000000000000f110df48ab84116172",
"size_on_disk": 23395947378,
"pruned": false,
"softforks": [
{
"id": "bip34",
"version": 2,
"reject": {
"status": true
}
},
{
"id": "bip66",
"version": 3,
"reject": {
"status": true
}
},
{
"id": "bip65",
"version": 4,
"reject": {
"status": true
}
}
],
"bip9_softforks": {
"csv": {
"status": "active",
"startTime": 1456790400,
"timeout": 1493596800,
"since": 770112
},
"segwit": {
"status": "active",
"startTime": 1462060800,
"timeout": 1493596800,
"since": 834624
}
serveradmin.ru

Нас интересует разница указанных значений. Они уже в десятиричном формате, так что проблем с их распознаванием нет. Все будет сделано в шаблоне, в том числе и настроен триггер на контроль разницы этих значений.

Второй пункт из списка на тему мониторинга сети настраивается парсингом вывода следующей команды:

```
/usr/bin/bitcoin-cli -rpcuser=user -rpcpassword=password getnetworkinfo
```

Вывод:

```
{
  "version": 170000,
  "subversion": "/Satoshi:0.17.0/",
  "protocolversion": 70015,
  "localservices": "000000000000040d",
  "localrelay": true,
  "timeoffset": 0,
  "networkactive": true,
  "connections": 8,
  "networks": [
    {
      "name": "ipv4",
      "limited": false,
      "reachable": true,
      "proxy": "",
      "proxy_randomize_credentials": false
    },
    {
      "name": "ipv6",
      "limited": true,
      "reachable": false,
      "proxy": "",
      "proxy_randomize_credentials": false
    },
    {
      "name": "onion",
      "limited": true,
      "reachable": false,
      "proxy": "",
      "proxy_randomize_credentials": false
    }
  ],
  "relayfee": 0.00001000,
  "incrementalfee": 0.00001000,
  "localaddresses": [
  ],
  "warnings": "Warning: unknown new rules activated (versionbit 28)"
}
```

Весь этот json тоже отправляем на сервер и там парсим и проверяем параметр **networkactive**. Он должен быть **true**.

Таким образом в zabbix-agent на нужно добавить следующие параметры:

```
UserParameter=blockchaininfo,/usr/bin/bitcoin-cli -rpcuser=user -rpcpassword=password getblockchaininfo
UserParameter=networkinfo,/usr/bin/bitcoin-cli -rpcuser=user -rpcpassword=password getnetworkinfo
```

Я их добавил в отдельный конфиг — /etc/zabbix/zabbix_agentd.d/btc.conf Перезапускаем агент и проверяем:

```
# zabbix_agentd -t blockchaininfo
# zabbix_agentd -t networkinfo
```

В выводе должны увидеть полные json строки. Если все в порядке, переходим к zabbix серверу для настройки ноды биткоина там.

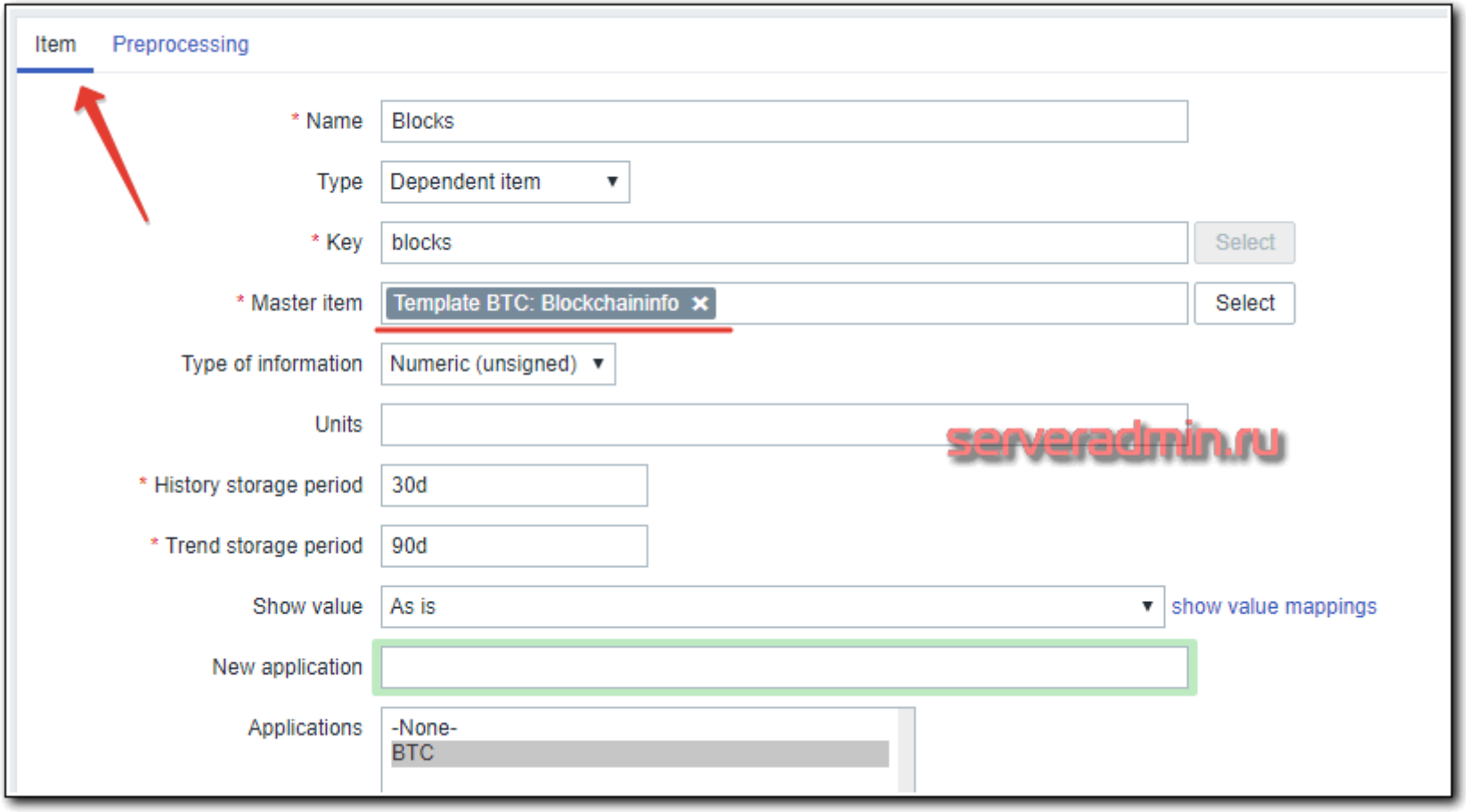
Для этого импортируем мой готовый шаблон для bitcoin node — Cryptonode BTC check

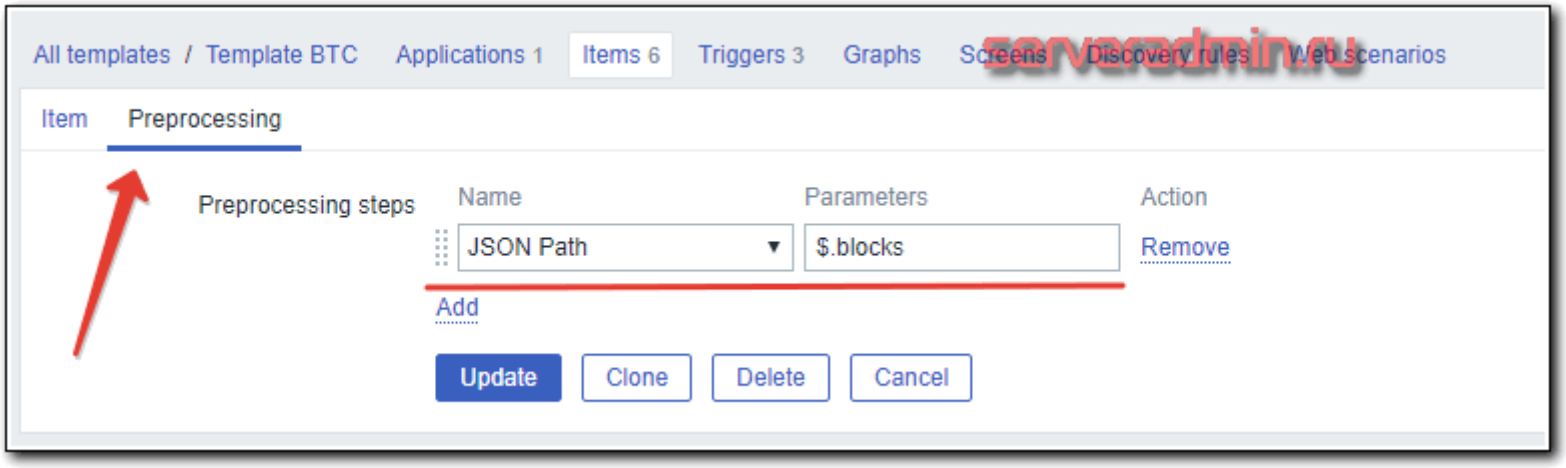
Там присутствуют следующие итемы:

Wizard	Name ▲	Triggers	Key	Interval	History	Trends	Type	Applications	Status
	Blockchaininfo		blockchaininfo	1m	30d		Zabbix agent	BTC	Enabled
	Blockchaininfo: Blocks	Triggers 1	blocks		30d	90d	Dependent item	BTC	Enabled
	Blockchaininfo: Headers	Triggers 1	headers		30d	90d	Dependent item	BTC	Enabled
	Networkinfo: Networkactive	Triggers 1	networkactive		30d		Dependent item	BTC	Enabled
	Networkinfo		networkinfo	1m	30d		Zabbix agent	BTC	Enabled
	Service bitcoind status	Triggers 1	proc.num[bitcoind]	1m	10d	30d	Zabbix agent	BTC	Enabled

Displaying 6 of 6 found

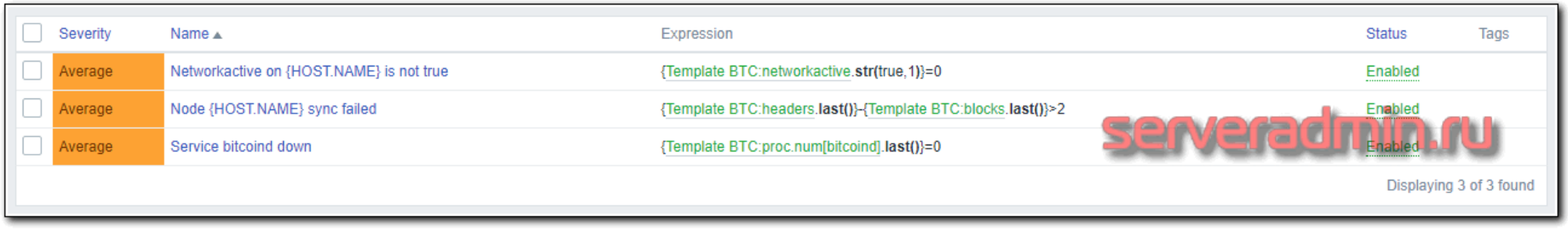
Вот пример зависимого итема, который получает значение blocks из json строки.


Item
Preprocessing
* Name
Blocks
Type
Dependent item
* Key
blocks
Select
* Master item
Template BTC: Blockchaininfo
Select
Type of information
Numeric (unsigned)
Units
serveradmin.ru
* History storage period
30d
* Trend storage period
90d
Show value
As is
show value mappings
New application
Applications
-None-
BTC



Более подробно о парсинге json строк читайте в отдельной статье, ссылку на которую я приводил в начале.

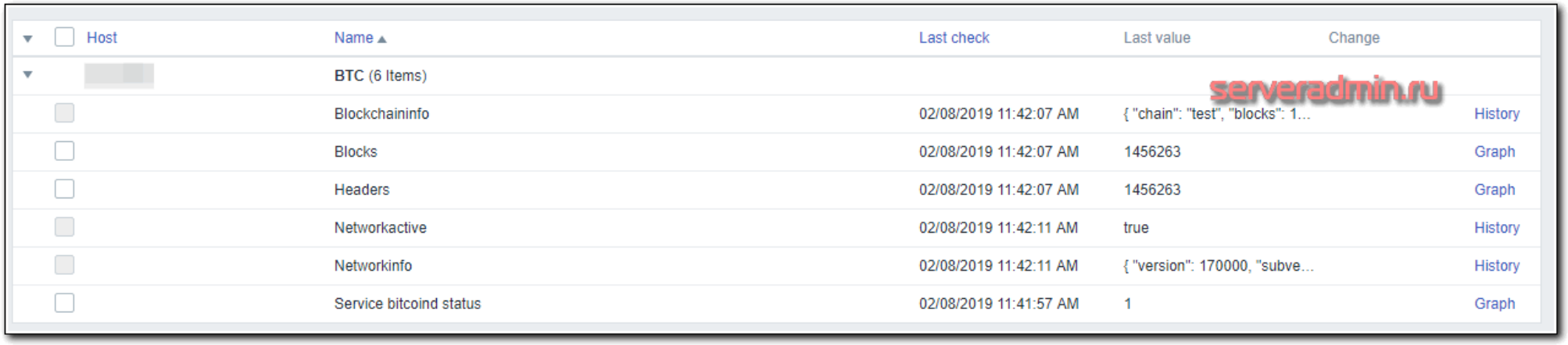
Список триггеров в шаблоне:



Severity	Name ▲	Expression	Status	Tags
Average	Networkactive on {HOST.NAME} is not true	{Template BTC:networkactive.**str**(true,1)}=0	Enabled	
Average	Node {HOST.NAME} sync failed	{Template BTC:headers.**last()**}-{Template BTC:blocks.**last()**}>2	Enabled	
Average	Service bitcoind down	{Template BTC:proc.num[bitcoind].**last()**}=0	Enabled	

Displaying 3 of 3 found

Тут все понятно из описания триггеров. Комментировать нечего. После применения шаблона к хосту с биткоин нодой, получите следующие данные.



Host	Name ▲	Last check	Last value	Change	
	BTC (6 Items)				
	Blockchaininfo	02/08/2019 11:42:07 AM	{ "chain": "test", "blocks": 1...		History
	Blocks	02/08/2019 11:42:07 AM	1456263		Graph
	Headers	02/08/2019 11:42:07 AM	1456263		Graph
	Networkactive	02/08/2019 11:42:11 AM	true		History
	Networkinfo	02/08/2019 11:42:11 AM	{ "version": 170000, "subve...		History
	Service bitcoind status	02/08/2019 11:41:57 AM	1		Graph

При открытии History к json элементам, можно посмотреть полностью всю строку.

Все остальные клоны биткоина мониторятся похожим образом. Отличаться может консольная команда, например, litecoin-cli, dash-cli и т.д. Так же названия системных служб будут отличаться. Для настройки мониторинга этих нод достаточно скопировать шаблон и конфиги bitcoin и отредактировать.

Мониторинг внешнего порта для rpc команд настраивается точно так же, как для ethereum, не буду повторяться. Достаточно только поменять ip адрес и номер порта.

Заключение

> Не понравилась статья и хочешь научить меня администрировать? Пожалуйста, я люблю учиться. Комментарии в твоем распоряжении. Расскажи, как сделать правильно!

Очередной наглядный пример простоты и удобства работы с мониторингом zabbix. С его помощью можно настроить мониторинг чего угодно. Главное, данные подать в удобочитаемом виде.

Очень нравится реализованная в нем работа с json. Не помню точно, с какой версии она появилась, но я стал пользоваться только недавно, когда появилась необходимость. Сейчас формат json очень популярен.

У меня накопился неплохой опыт работы с нодами различных криптовалют. Если кому-то нужна помощь или советы по работе с ними, задавайте в комментариях к статье про настройку криптонод или в темах на форуме.