Puppet konfiguriert Nagios automatisch

2013-10-23

Bernd Strößenreuther
mailto:bs@mathias-kettner.de

Open Source Monitoring Conference 2013
Nürnberg

Mathias Kettner GmbH
Linux und Open Source

Kellerstraße 29, 81667 München • Tel.: 089 / 18 90 42 10 • info@mathias-kettner.de • www.mathias-kettner.de

Lizenz

Mathias Kettner GmbH

- Open-Source Monitoring
  - Check_MK - umfassendes Addon für Nagios
  - OMD - voll integrierte Monitoringlösung
- Schulungen zu Check_MK und Open Source
- Firmensitz: München

Puppet arbeitet deklarativ

- Beschreibe den Zielzustand

Erster Ansatz

- Config-Files, Dienste, Pakete usw. werden eins zu eins über Puppet definiert
- Templates zur Vereinheitlichung

Zu überwachender Server

```
class apache{
   package { 'httpd':
      ensure => installed;
   }
   file { '/etc/httpd/conf/httpd.conf':
      owner   => 'root',
      group   => 'root',
      mode    => '0644',
      source  => 'puppet:///apache/httpd.conf',
      require => Package['httpd'],
      notify  => Service['httpd'];
   }
   service {'httpd':
      ensure => running,
      enable => true;
   }
}
```

Monitoring-Maschine

```
class nagios-server {
   file { '/etc/nagios/objects.puppet/
           services.http.cfg':
      owner => 'root',
      group => 'root',
      mode  => '0644';
   }
   nagios_service { "check_http_webserv1":
      check_command       => 'check_http_port_path!80!/',
      use                 => 'generic-service',
      host_name           => 'webserv1',
      notification_period => '24x7',
      service_description => 'HTTP GET /',
      target              => '/etc/nagios/objects.puppet/
                              services.http.cfg‘;
   }
   nagios_service { "check_http_webserv2":
      # ..
   }
}
```

Puppets Potential voll ausschöpfen

- Ressourcen werden logisch zusammengehörend definiert
- „Single Source“

Zu überwachender Server

```
class apache {
   service {'httpd':
      ensure    => running,
      enable    => true,
   }
   @@nagios_service { "check_http_${hostname}":
      check_command       => 'check_http_port_path!80!/',
      use                 => 'generic-service',
      host_name           => $hostname,
      notification_period => '24x7',
      service_description => 'HTTP GET /',
      target              => '/etc/nagios/objects.puppet.
                              autogen/services.http.cfg';
   }
}
```

Monitoring-Maschine

```
class nagios-server {
   file { '/etc/nagios/objects.puppet.autogen/
           services.http.cfg':
      owner => 'root',
      group => 'root',
      mode  => '0644';
   }
}

# collect resources and populate
# /etc/nagios/objects.puppet.autogen/*.cfg
Nagios_service <<||>>
```

Ergebnis

```
[booboo@nagios-server ~]$ cat services.http.cfg

# HEADER: This file was autogenerated at
# HEADER: Fri Dec 14 13:53:26 +0100 2012
# HEADER: by puppet. While it can still be managed
# HEADER: manually, it is definitely not recommended.
define service {
    ## --PUPPET_NAME-- (called '_naginator_name' in
    ## the manifest) check_http_webserv1
    use                     generic-service
    service_description     HTTP GET /
    check_command           check_http_port_path!80!/
    host_name               webserv1
    notification_period     24x7
}
```

Voraussetzungen

- Voraussetzungen für Exported Resources:

```
[master]
storeconfigs = true
```

- Siehe

  http://docs.puppetlabs.com/guides/exported_resources.html bzw.
  http://docs.puppetlabs.com/puppet/2.7/reference/lang_exported.html

- Wo persistieren?
- Empfehlung: PuppetDB, siehe

  http://docs.puppetlabs.com/puppetdb/1.4/

Was konfiguriere ich wo?

- Basis-Monitoring der Maschine (CPU, RAM, Disks, …) und den Host-Check bei der Maschine selbst
  - z. B. Klasse basemonitoring beim basenode einbinden
- Dienste-spezifische Monitorings in der Klasse, die auch den Dienst definiert
  - z. B. check_http in der Klasse httpd
- Für alles darüber hinaus: individuelle Lösung finden

Tags nutzen

- Alle Ressourcen vom Typ nagios_service werden realisiert:

```
Nagios_service <<||>>
```

- Nur ausgewählte:

```
Nagios_service <<| tag == 'datacenter1' |>>
```

Tags zuweisen

- Tags zuweisen:

```
@@nagios_service { "check_http_${hostname}":
    check_command       => 'check_http_port_path!80!/',
    tag                 => 'datacenter1',
    # […]
}
```

- oder dynamisch mit einem Custom Fact:

```
@@nagios_service { "check_http_${hostname}":
    check_command       => 'check_http_port_path!80!/',
    tag                 => $datacenter,
    # […]
}
```

Nagios-Resourcen in Puppet

- nagios_command
- nagios_contact
- nagios_contactgroup
- nagios_host
- nagios_hostdependency
- nagios_hostescalation
- nagios_hostextinfo
- nagios_hostgroup
- nagios_service
- nagios_servicedependency
- nagios_serviceescalation
- nagios_serviceextinfo
- nagios_servicegroup
- nagios_timeperiod

Siehe http://docs.puppetlabs.com/references/2.7.latest/type.html

Teil 2

- Von und mit Sascha Brechmann