KONFIGURATIONSMANAGEMENT **MIT PUPPET**

TIM SCHMELING

4b:65:72:73:74:69:6e:2c:20:69:63:68:20:6c:69:65:62:65:20:64:69:63:68:21

WAS IST **PUPPET?**

- SW für Konfigurationsmanagement
- Geschrieben in Ruby
- Client-Server Architektur
- Betriebssystemunabhängig
  (Fokus auf Unix-artige OS)
- Open Source oder Enterprise?

WARUM **PUPPET?**

- Zeiteinsparungen
- Zentrales Konfigurationsmanagement
- Versionskontrolle
- Einheitliche Konfiguration
- Fehler vermeiden

VORAUSSETZUNGEN

- Unterstütztes OS
- Ruby
- Facter
- Zu verwaltende Netzerkgeräte
- Ein Ziel!

AUFGABEN EINES **ADMIN**

- Installation
  - OS Installation
  - Konfiguration des OS
  - Software installieren und konfigurieren
- Betrieb
  - Änderungen an Konfigurationen
  - Useraccounts verwalten

WIE FUNKTIONIERT **PUPPET?**

- Puppet Master
  - Serverdaemon
  - Zertifizierungsstelle (certificate authority)
  - Integrierter Webserver (WebRICK)
  - Apache/nginx mit mod_passenger

WIE FUNKTIONIERT **PUPPET?**

- Puppet Agent
  - Clientdaemon pollt alle 30 Minuten
  - Manuell gesteuert

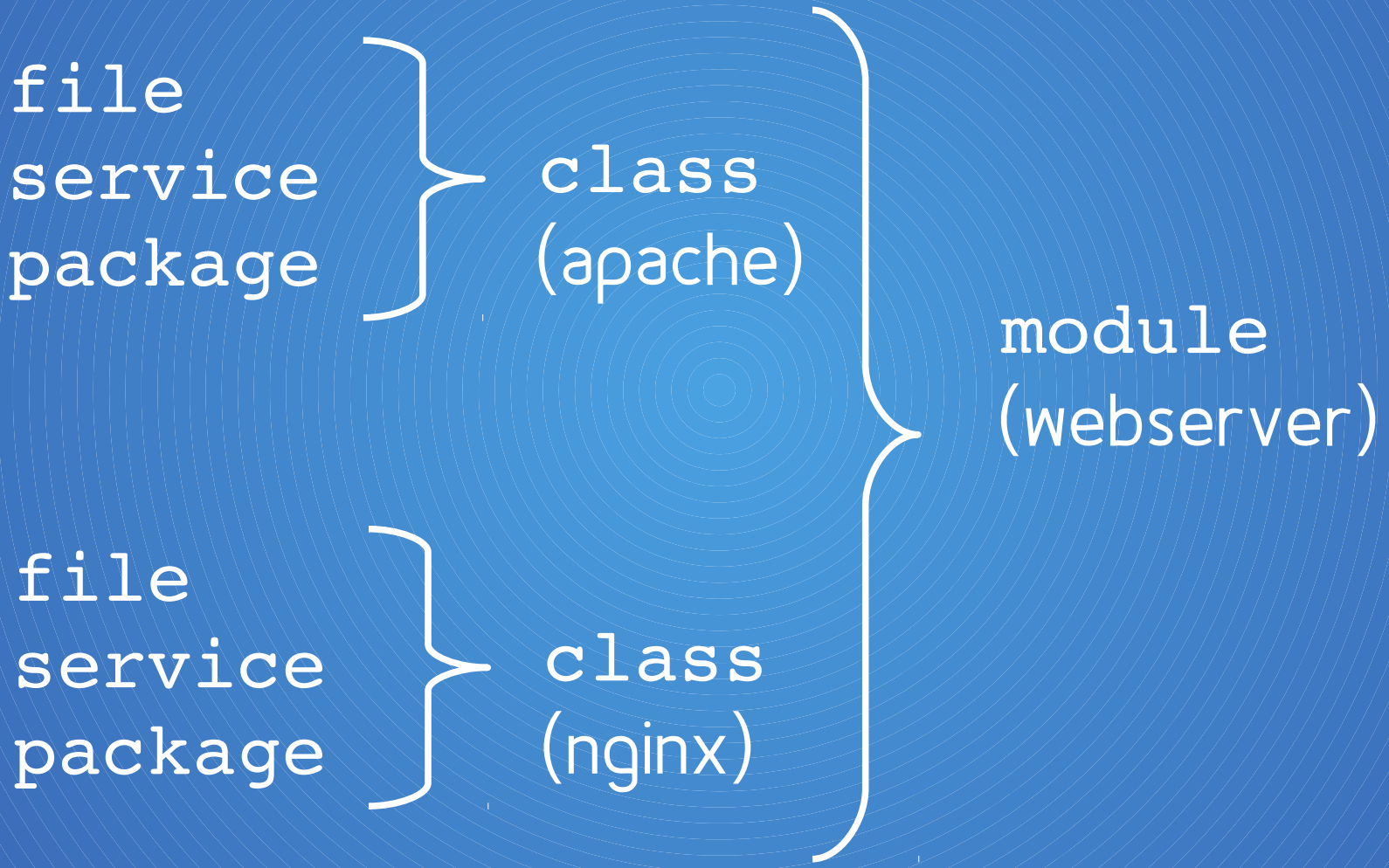
BEGRIFFE SCHNELL **ERKLÄRT**

- Manifests
- Node
- Puppet Master
- Ressourcen
- Facter
- Templates
- Module

WAS SIND **RESSOURCEN?**

- Dateien / Verzeichnisse (**`file`**)
- Dienste (**`service`**)
- Softwarepakete (**`package`**)
- Benutzeraccounts (**`user`**)
- Host-Einträge (**`host`**)
- SSH-Host-Keys (**`sshkey`**)

RESSOURCEN GLIEDERN, **MODULE**



TEMPLATES

```
Herzlich Willkommen
Hostname: <%= hostname %>
RAM:      <%= memorytotal %> GB

----------------------------------

Herzlich Willkommen
Hostname: node1
RAM:      32 GB

root@node1:/root#
```

TEMPLATES

```
file {"/etc/motd":
   ensure  => present,
   content => template("system/etc/motd.erb"),
}
```

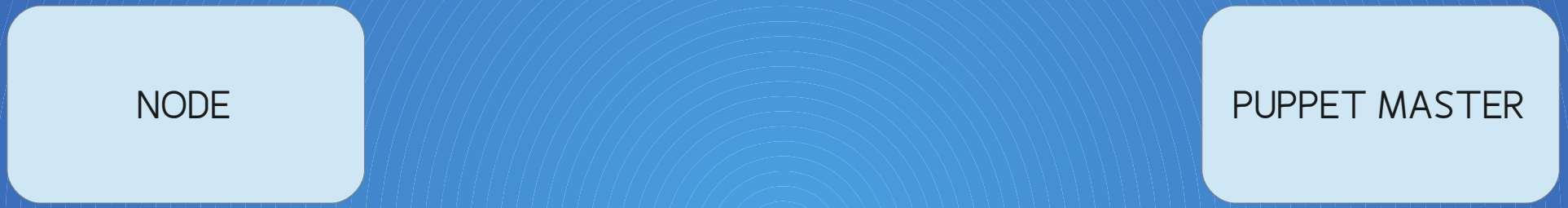
TEMPLATES

```
file {"/etc/motd":
   ensure  => present,
   content => template("system/etc/motd.erb"),
}


file { $operatingsystem ? {
     Debian => "/var/run/motd",
     RedHat => "/etc/motd",
   }:
   ensure  => present,
   content => template("system/etc/motd.erb"),
}
```

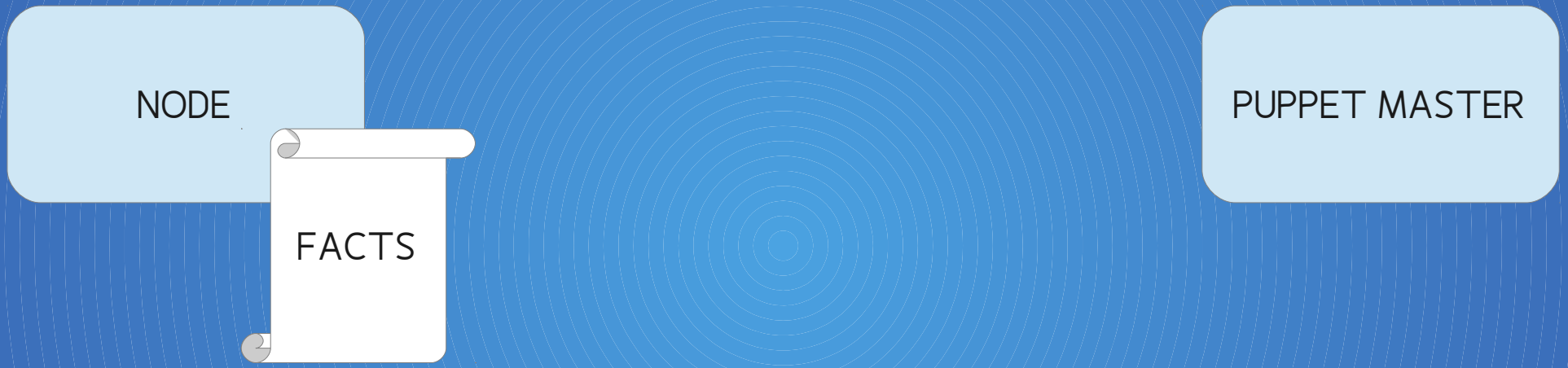
EXEC, DIE "BÖSE" **RESSOURCE?**

```
exec { "generate_sendmail_config":
 command => "m4 /etc/mail/sendmail.mc > /etc/mail/sendmail.cf",
 subscribe   => File["/etc/mail/sendmail.mc"],
 path        => ["/usr/sbin", "/usr/bin", "/sbin", "/bin"],
 refreshonly => true,
}


file { "/etc/mail/sendmail.mc":
 ensure  => present,
 notify  => Exec["generate_sendmail_config"],
 content => template("sendmail/etc/mail/sendmail.mc.erb"),
}
```

WIE FUNKTIONIERT **PUPPET?**



WIE FUNKTIONIERT **PUPPET?**



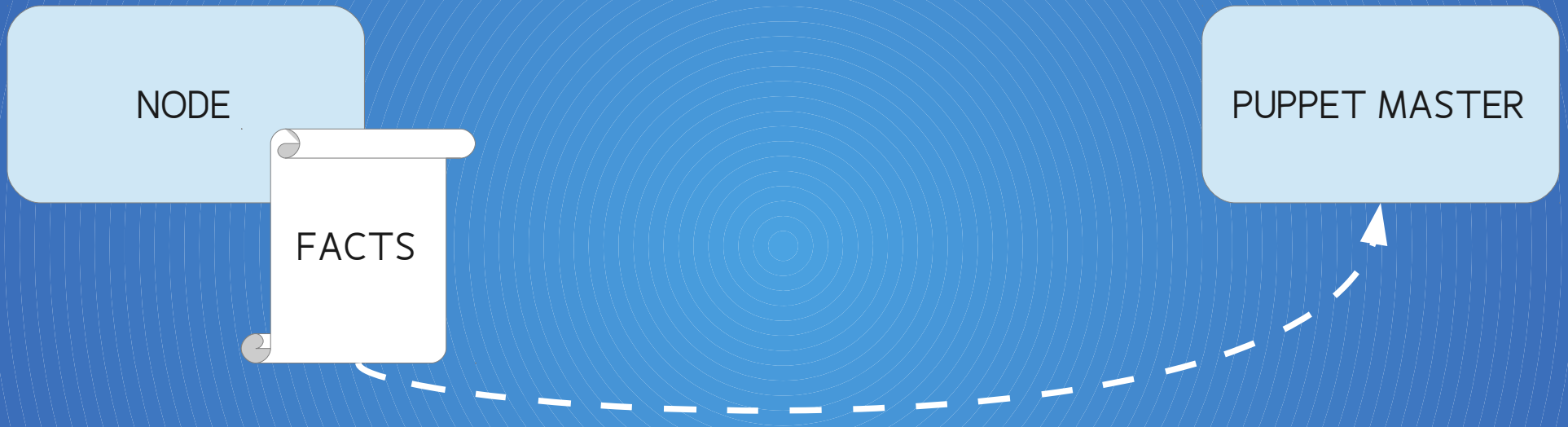
WIE FUNKTIONIERT **PUPPET?**

NODE

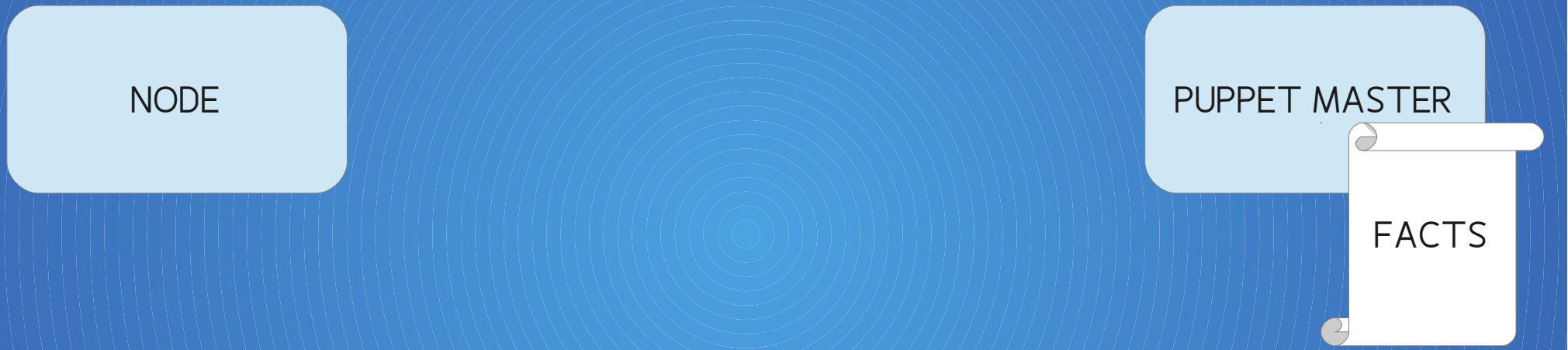
MASTER

```
hostname => puppet-test
id => root
interfaces => eth0,lo
ipaddress => 192.168.2.88
ipaddress_eth0 => 192.168.2.88
ipaddress_lo => 127.0.0.1
is_virtual => true
kernel => Linux
kernelmajversion => 3.2
kernelrelease => 3.2.0-4-486
kernelversion => 3.2.0
lsbdistcodename => wheezy
lsbdistdescription => Debian GNU/Linux 7.1 (wheezy)
lsbdistid => Debian
lsbdistrelease => 7.1
lsbmajdistrelease => 7
macaddress => 08:00:27:03:e5:f0
macaddress_eth0 => 08:00:27:03:e5:f0
manufacturer => innotek GmbH
memoryfree => 212.53 MB
memorysize => 375.73 MB
memorytotal => 375.73 MB
netmask => 255.255.255.0
netmask_eth0 => 255.255.255.0
netmask_lo => 255.0.0.0
network_eth0 => 192.168.2.0
network_lo => 127.0.0.0
operatingsystem => Debian
operatingsystemrelease => 7.1
osfamily => Debian
path => /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
physicalprocessorcount => 1
processor0 => Intel(R) Core(TM)2 Duo CPU     T7250  @ 2.00GHz
processorcount => 1
```

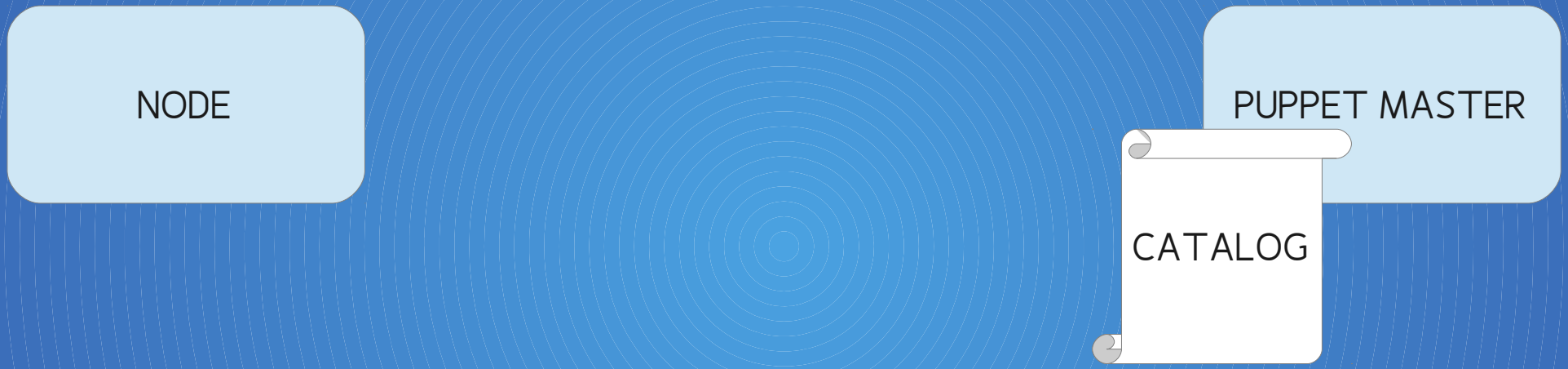
WIE FUNKTIONIERT **PUPPET?**



WIE FUNKTIONIERT **PUPPET?**



WIE FUNKTIONIERT **PUPPET?**

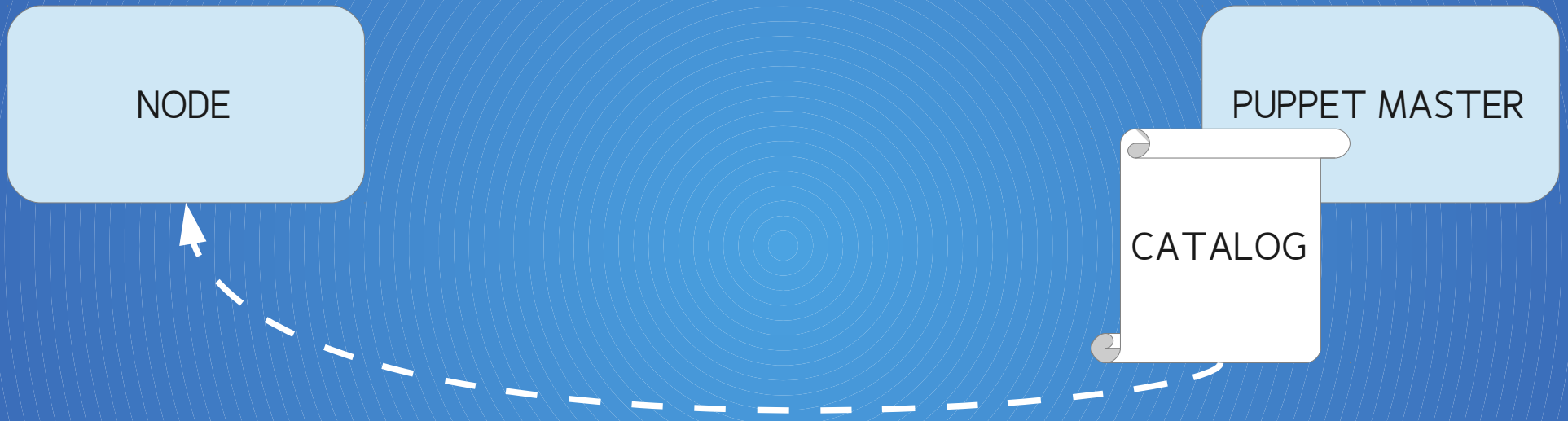


WIE FUNKTIONIERT **PUPPET?**

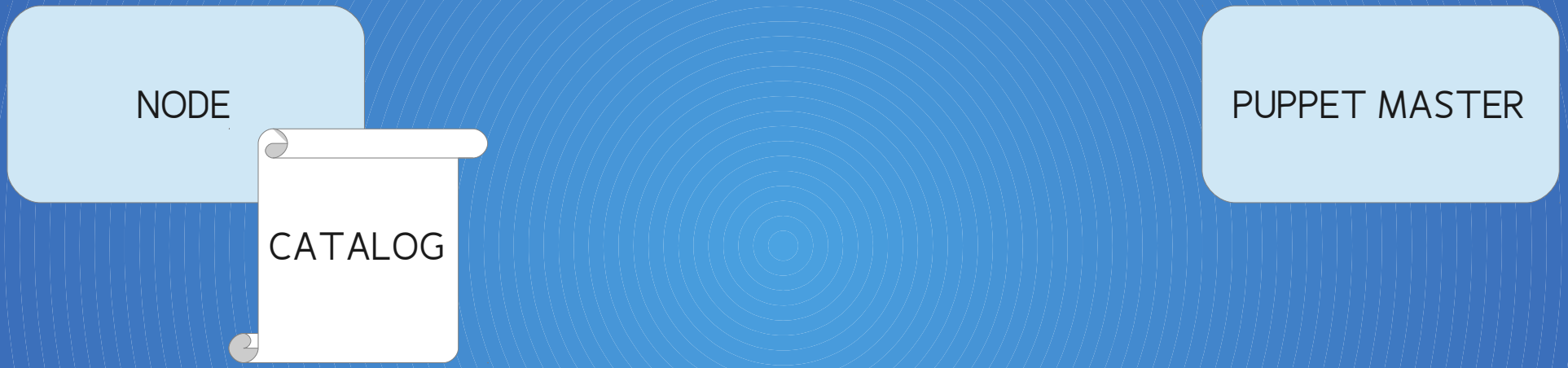

```
"resources": [
  {
    "line": 26,
    "title": "/home/support/.ssh/authorized_keys",
    "type": "File",
    "parameters": {
      "group": "support",
      "ensure": "present",
      "content": "AAAADAQABAAABAQ..... peter@workstation45\nAAAAB3NzaC1kc3M.....
      "owner": "support",
      "require": "File[/home/support/.ssh]",
      "mode": "0600"
    },
    "tags": [
      "file",
      "ssh::authrzd_key",
      "ssh",
      "authrzd_key",
      "support",
      "class",
      "node",
      "puppet-test"
    ],
    "exported": false,
    "file": "/etc/puppet/environments/test/modules/ssh/manifests/init.pp"
  },
  {
    "line": 5,
    "title": "support",
    "type": "Ssh::Authrzd_key",
    "parameters": {
      "authkeys": [
        "AAAADAQABAAABAQ..... peter@workstation45",
        "AAAAB3NzaC1kc3M..... tim@workstation1",
```


ET MASTER

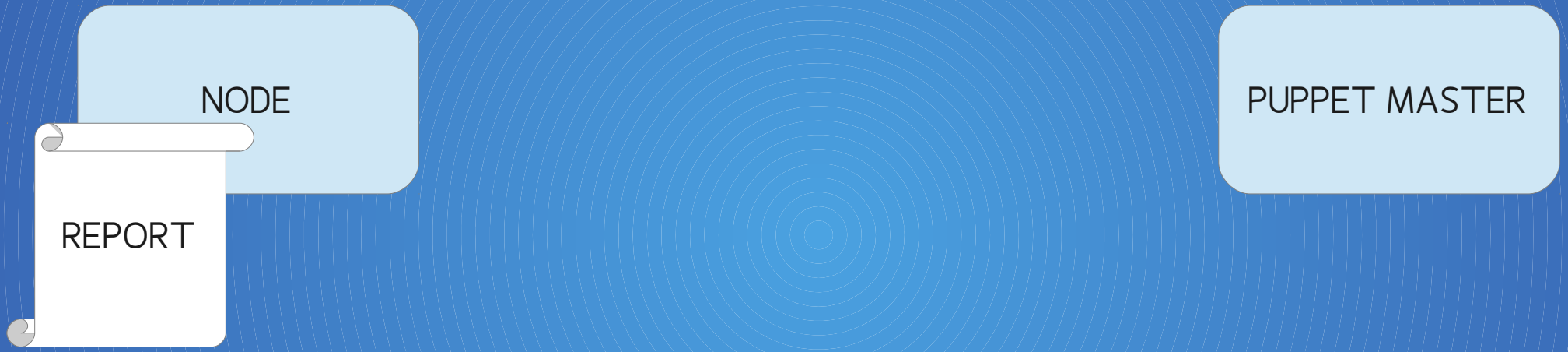
WIE FUNKTIONIERT **PUPPET?**



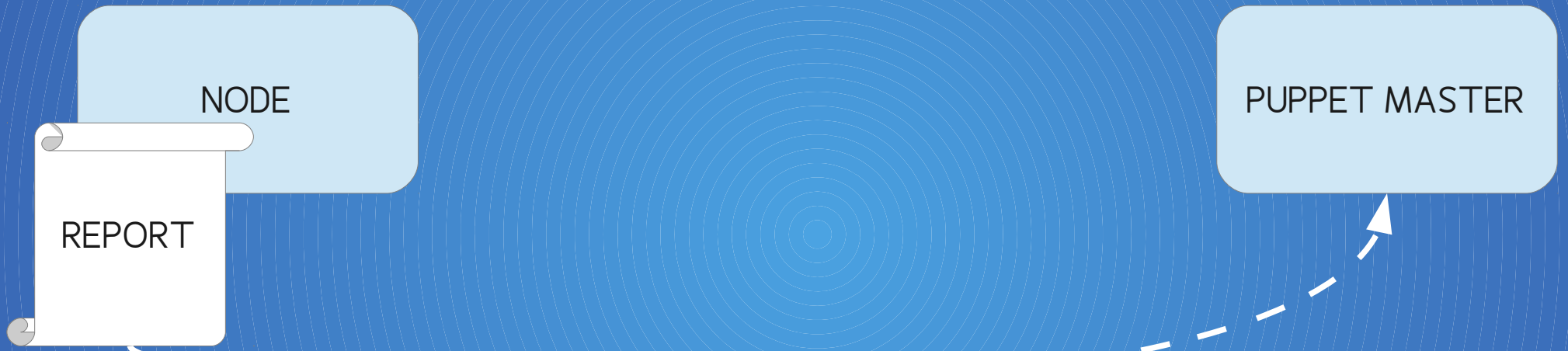
WIE FUNKTIONIERT **PUPPET?**



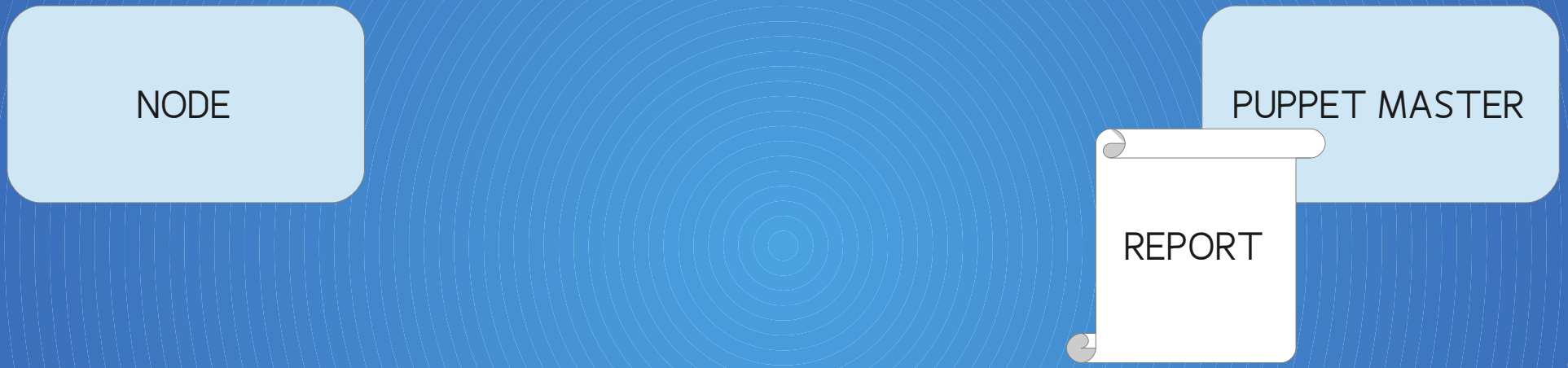
WIE FUNKTIONIERT **PUPPET?**



WIE FUNKTIONIERT **PUPPET?**



WIE FUNKTIONIERT **PUPPET?**



ENVIRONMENTS

- In Anlehnung an SW-Entwicklung
- Standard „production“
- unbegrenzt viele
- Aus der Praxis: „test“, „pre-prod“, „production“
- Gesteuert über Node oder Puppet Master

ENVIRONMENTS

Puppet Master (**puppet.conf**)

```
[production]
manifest = $confdir/env/$environment/manifests/site.pp

[pre-prod]
manifest = $confdir/env/$environment/manifests/site.pp

[test]
manifest = $confdir/env/$environment/manifests/site.pp
```

Puppet Agent / Node (**puppet.conf**)

```
[agent]
  environment = test
```

BEZIEHUNGEN, **METAPARAMETER**

- `before`
  Ressource VOR der anderen ausführen
- `require`
  Ressource NACH der anderen ausführen
- `notify`
  Resssource(n) benachrichtigen, wenn geändert
- `subscribe`
  Änderungen einer anderen Ressource überwachen

BEZIEHUNGEN, **METAPARAMETER**

- `before`
  Ressource VOR der anderen ausführen
- `r`
  Re
- `n`
  Re
- `s`
  Ä

```
package { 'openssh-server':
  ensure => present,
  before => File['/etc/ssh/sshd_config'],
}

file { '/etc/ssh/sshd_config':
  ensure  => file,
  source  => 'puppet:///modules/sshd/sshd_config',
  require => Package['openssh-server'],
}
```

```
file { '/etc/ssh/sshd_config':
  ensure => file,
  source => 'puppet:///modules/sshd/sshd_config',
  notify => Service['sshd'],
}

service { 'sshd':
  ensure    => running,
  enable    => true,
  subscribe => File['/etc/ssh/sshd_config'],
}
```

- `notify`
  Resssource(n) benachrichtigen, wenn geändert
- `subscribe`
  Änderungen einer anderen Ressource überwachen

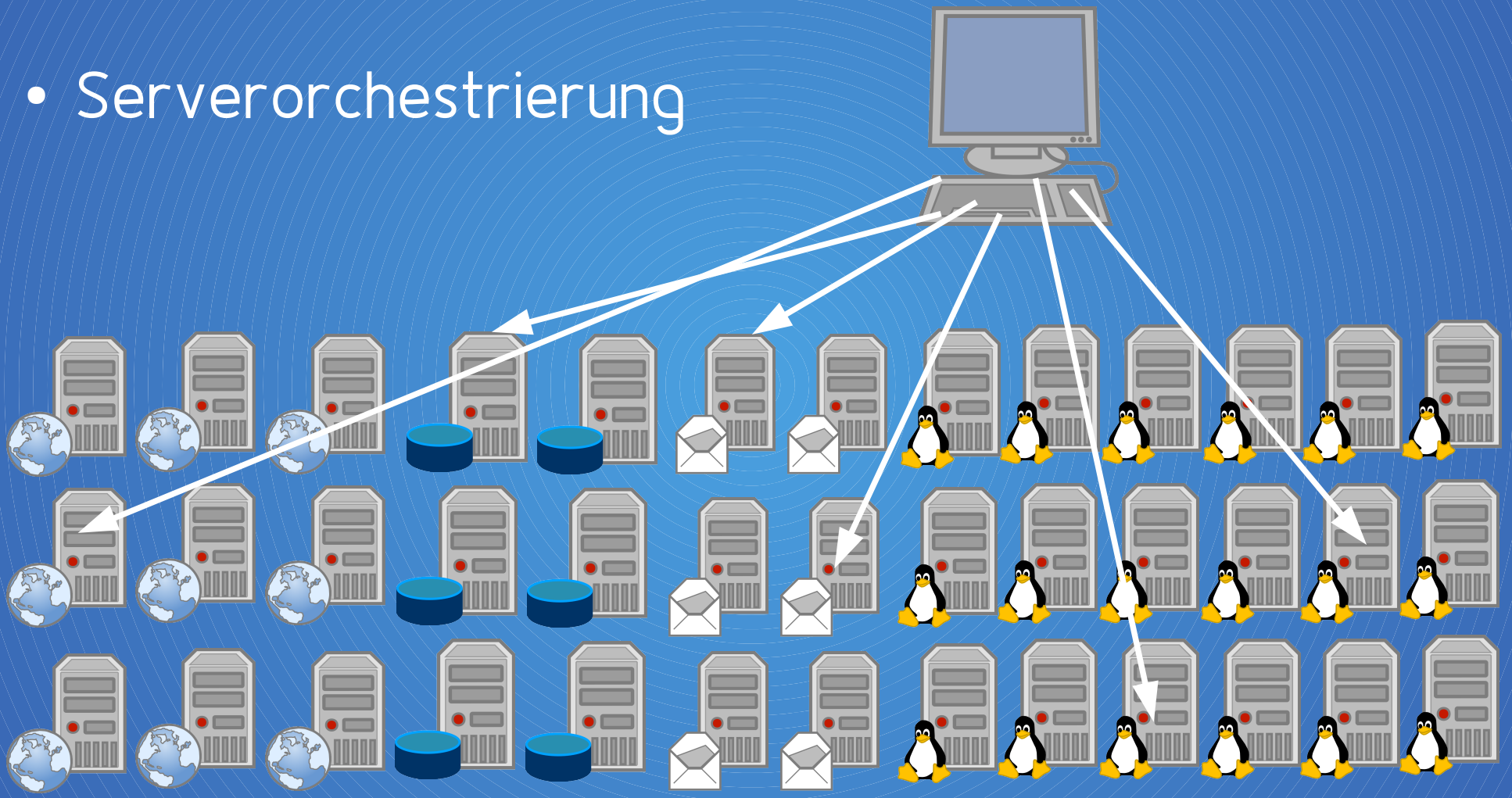
STAGING / **STAGES**

- Keine Beziehungen / Abhängigkeiten!
- Ausführung „steuern“

```
stage { 'pre_main':
 before => Stage['main'],
}
class { 'repository':
 stage => pre_main,
}
```

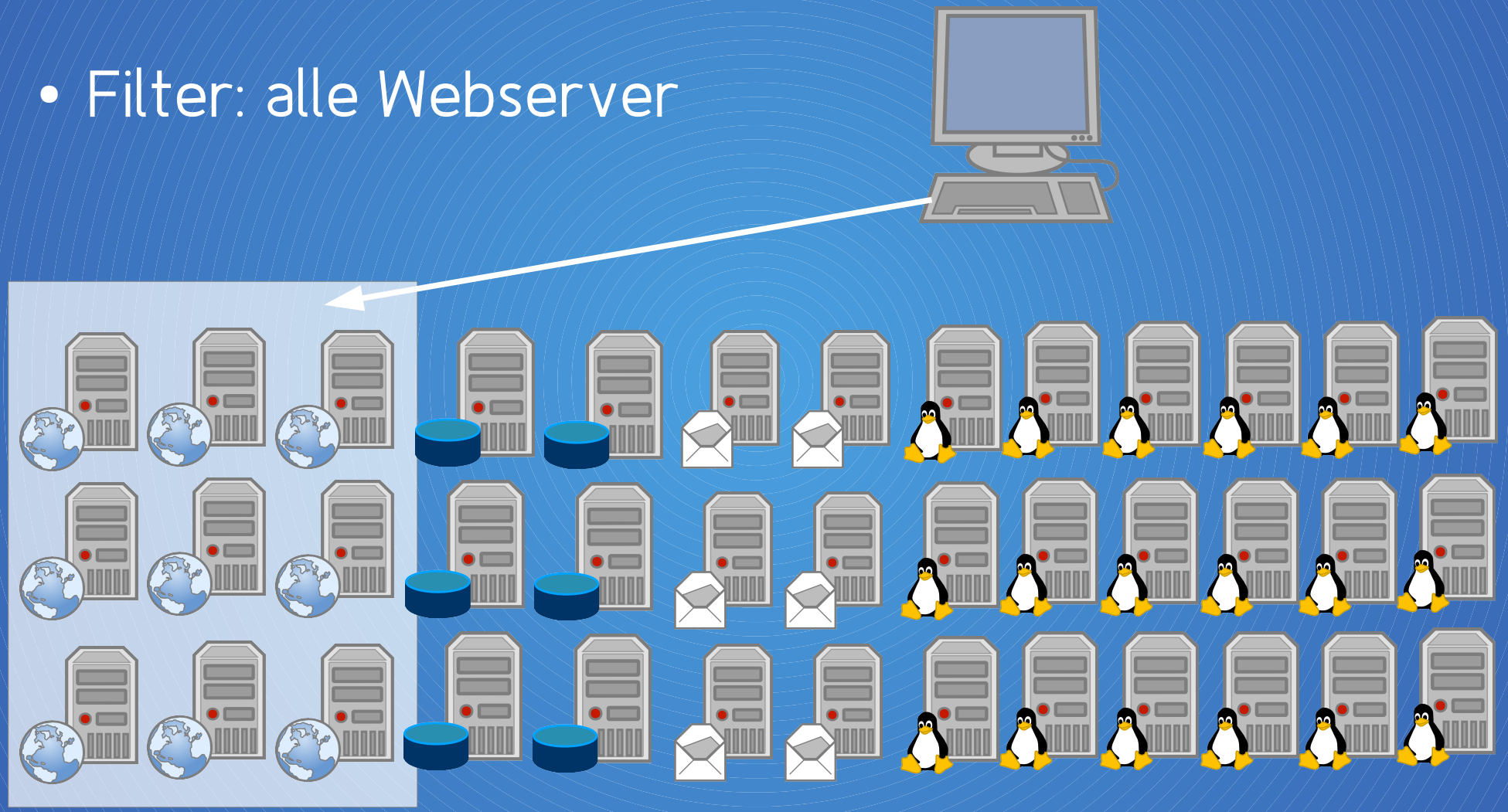
SERVERORCHESTRIERUNG, **MCOLLECTIVE**

- Serverorchestrierung



SERVERORCHESTRIERUNG, **MCOLLECTIVE**

- Filter: alle Webserver



WEITERE **INFORMATIONEN**

- docs.puppetlabs.com
- Puppet-Buch, opensourcepress.de
  Veröffentlichung Anfang 2014

DANKE FÜR IHR **AUFMERKSAMKEIT**

FRAGEN?