IT-Services & Consulting

Konfigurationsmanagement mit Chef

NETexpress GmbH, 08.03.2018

solutions & more

Inhalt

- Grundlagen
- Administration mit Chef
  - Testing
  - Module
  - Cookbooks
- Chef vs. Puppet
- Demo

Grundlagen

1. **Installation eines Servers**
2. **Kunde bemerkt Fehler**
3. **Fehlerbehebung, eine Zeile in Konfiguration vergessen**
4. **Kunde bemerkt weiteren Fehler**
5. **Fehlerbehebung, oben geänderte Zeile nicht 100%ig korrekt**
6. **Kunde bemerkt wieder Fehler…**

Lösung:

- Configuration Management/Continuous Delivery/Deployment
- Ziel: Infrastructure as Code, Idempotenz, Reproduzierbarkeit

Opscode Chef:

- OpenSource
- Läuft unter Linux, Unix, MacOS, Windows
- Client/Server-Modus (hosted/selfhosted) oder Standalone
- Ruby
- Alternativen: Puppet, Salt, Ansible, CFEngine

Vorteile:

- Benötigte Zeit für Installation/Konfiguration/Tests wird auf Dauer reduziert
- Änderungen auf vielen Systemen sind eine Sache von Minuten
- CMDB (lt. ITIL) ist bei einem Chef-Server enthalten (OHAI ermittelt die Konfiguration des Systems)
- Dokumentation kann auf ein Minimum beschränkt werden
- Prinzipiell reicht ein Backup der Userdaten
- Fehler werden während des Lifecycles der verwalteten Systeme im Optimalfall einmal begangen

Grundlagen

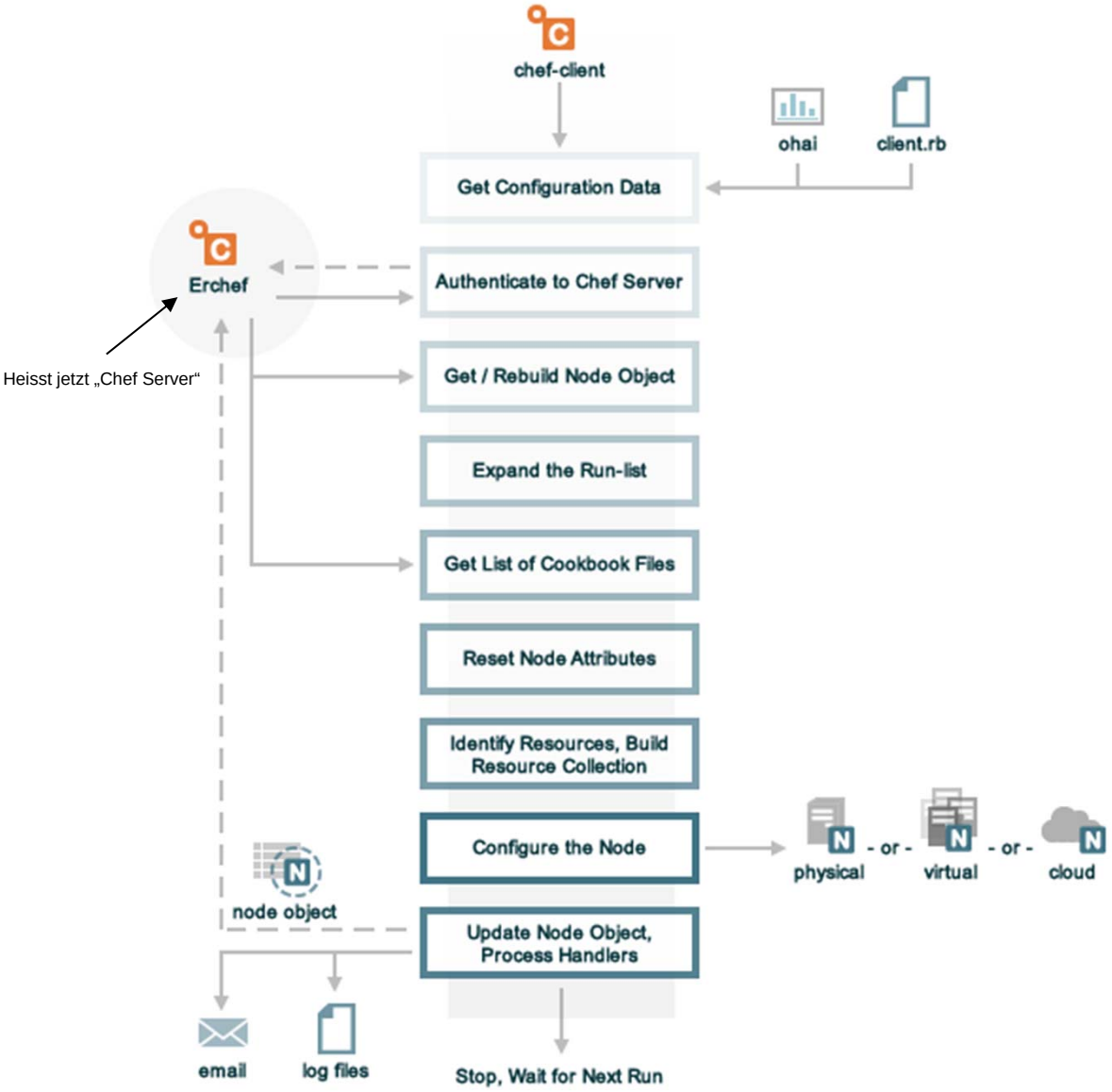
- Applikationen installieren/konfigurieren
- Services starten/stoppen
- Cronjobs anlegen
- Benutzer verwalten
- …..

Grundlagen

- Zugriff auf Registry und Umgebungsvariablen
- Benutzen der Powershell (-> DSC)
- Aufrufen von MSI-Installern
- ACLs setzen
- Microsoft Azure Cloud steuern
- Beispiel:

```
registry_key "HKLM\\Software\\MyApp\\MyConfig" do
  values [{
    :name => "Testkey",
    :type => :multi_string;
    :data => linux\0ist\0beste\0
  }]
  architecture :x86_64
  action :create
  not_if do ::File.exists?('c:/pleasenot') end
End
```

Grundlagen


chef-client
ohai
client.rb
Get Configuration Data
Erchef
Heisst jetzt „Chef Server“
Authenticate to Chef Server
Get / Rebuild Node Object
Expand the Run-list
Get List of Cookbook Files
Reset Node Attributes
Identify Resources, Build Resource Collection
Configure the Node
physical
- or -
virtual
- or -
cloud
node object
Update Node Object, Process Handlers
email
log files
Stop, Wait for Next Run

Andere haben sich vielleicht schon den Kopf über das Problem zerbrochen!

Administration mit Chef - Testing

- ~~Lokal~~
- Vagrant (manuell)

  (Volle) Tests auf „Wegwerf-VMs“

- Test kitchen (automatisch)

  Volle Tests via Vagrant, Docker, LXC

- Foodcritic

  LINT-Check (Code style, Syntax, häufige Fehler)

- ChefSpec

  Ausführung

Administration mit Chef - Testing

- Verwaltung von virtuellen (Test-)Maschinen (VMWare, VirtualBox, LXC..)
- Vordefinierte „Wegwerf-VMs“

```
[maxher@pc291-vm1 kw50_expertsession_chef]$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'add'...
==> default: Matching MAC address for NAT networking...
==> default: Setting the name of the VM: kw50_expertsession_chef_default_1449822007418_8378
0
WARNING: Nokogiri was built against LibXML version 2.9.2, but has dynamically loaded 2.9.3
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 80 => 1337 (adapter 1)
    default: 22 => 2222 (adapter 1)
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
==> default: Mounting shared folders...
    default: /vagrant => /home/maxher/Development/kw50_expertsession_chef
    default: /tmp/vagrant-chef/1ffb105e78bdbfd257bc2c1a3389a160/cookbooks => /home/maxher/D
evelopment/kw50_expertsession_chef/repo
==> default: Running provisioner: chef_solo...
==> default: Detected Chef (latest) is already installed
Generating chef JSON and uploading...
==> default: Running chef-solo...
==> default: [2015-12-11T08:22:07+00:00] INFO: Forking chef instance to converge...
==> default: Starting Chef Client, version 11.6.0
==> default:
==> default: [2015-12-11T08:22:08+00:00] INFO: *** Chef 11.6.0 ***
==> default: [2015-12-11T08:22:12+00:00] INFO: Setting the run_list to ["recipe[varnish::de
fault]"] from JSON
==> default: [2015-12-11T08:22:12+00:00] INFO: Run List is [recipe[varnish::default]]
==> default: [2015-12-11T08:22:12+00:00] INFO: Run List expands to [varnish::default]
==> default: [2015-12-11T08:22:12+00:00] INFO: Starting Chef Run for localhost
==> default: [2015-12-11T08:22:12+00:00] INFO: Running start handlers
```

Administration mit Chef - Komponenten

- **Workstation**

  Eigener Rechner, zur Verwaltung der Infrastruktur

- **Server**

  Zentraler Chef-Server, der die Konfiguration zentral im Netzwerk zur Verfügung stellt

- **Client**

  Nodes die von Chef gemanaged werden

Administration mit Chef - Module

- knife

  Zugriff auf die API (RESTful JSON), „Admintool“

- ohai

  Sammeln von Systeminformationen (RAM, CPU…), speichern in Databag

- chef-client

  Konfiguriert Zielmaschinen lt. Spezifikation

- chef-shell

  Debugging

[

Chef server

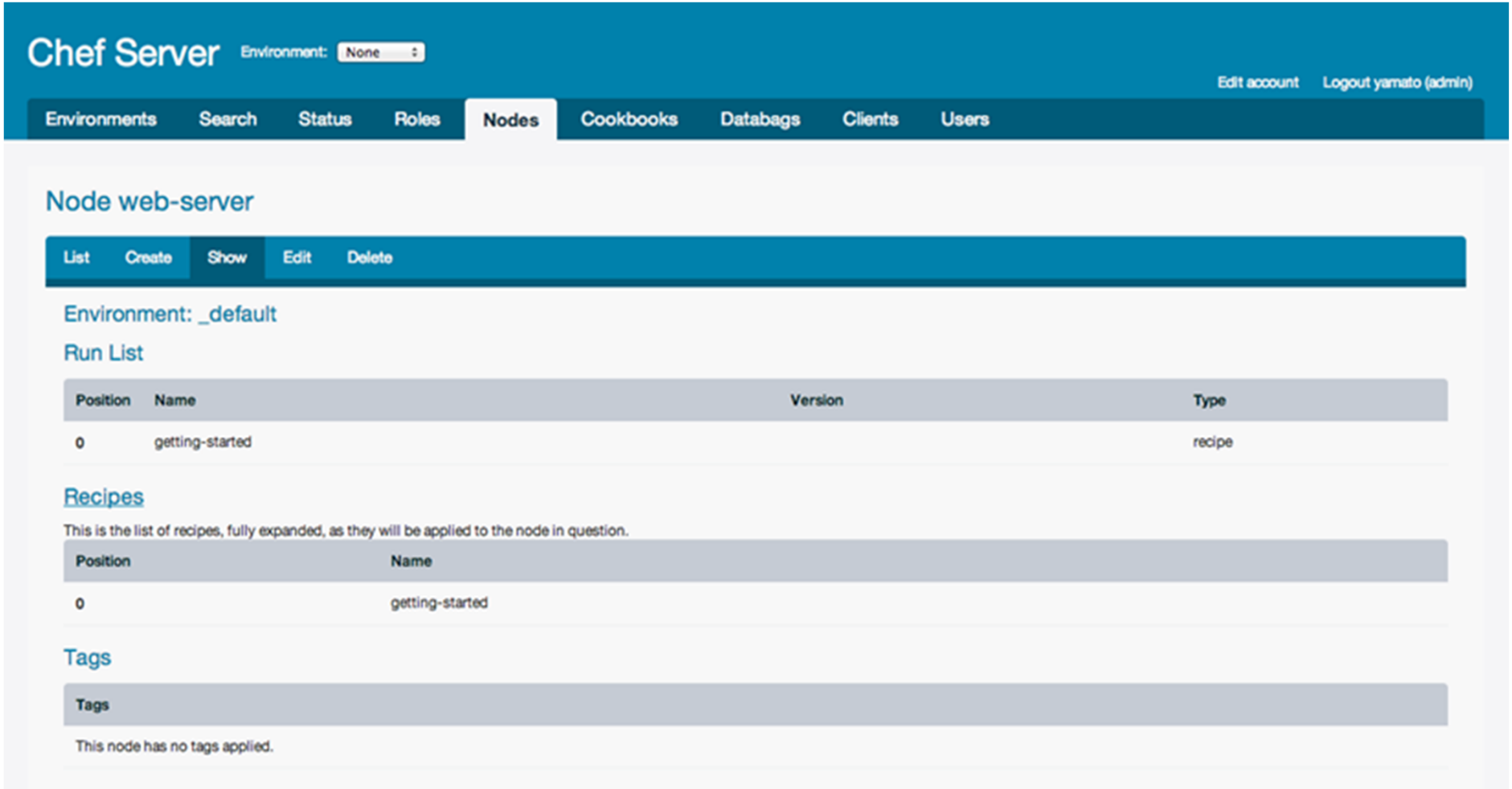
Zentrale Verwaltung

chef-apply

Einzelne RECIPES ~~ausführen~~ testen

]

Administration mit Chef - Modul Server



Administration mit Chef - Modul OHAI

- Sammeln von Systeminformationen des Rechners

  CPU, Netzwerk, Arbeitsspeicher, Dateisysteme, OS, FQDN, PHP/Python-Version, SSH-Host-Key...

- Erweiterbar durch (eigene) Plugins

- Aktualisierung bei jedem Chef-Run

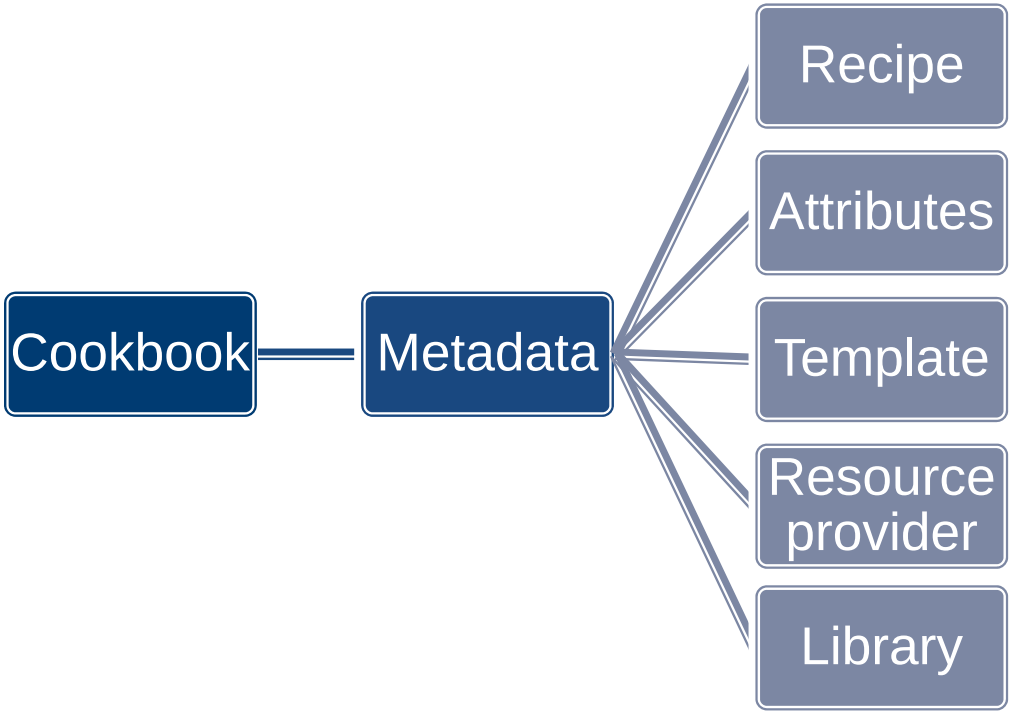

```
"languages": {
  "mono": {
    "version": "4.0.5",
    "builddate": "4.0.5.1/1d8d582 Thu Nov "
  },
  "perl": {
    "version": "5.22.0",
    "archname": "x86_64-linux-thread-multi"
  },
  "c": {
    "gcc": {
      "version": "20150618",
      "description": "gcc-Version 5.1.1 20150618 (Red Hat 5.1.1-4) (GCC) "
    },
    "glibc": {
      "version": "2.22",
      "description": "GNU C Library (GNU libc) stable release version 2.22, by Roland McGrath et al."
    }
  },
  "ruby": {
    "platform": "x86_64-linux",
    "version": "2.2.3",
    "release_date": "2015-08-18",
    "target": "x86_64-redhat-linux-gnu",
    "target_cpu": "x86_64",
    "target_vendor": "redhat",
```

- Führt Konfigurationsaufgaben auf Nodes aus
- ~~Läuft nicht automatisch~~ (cronjob)

```
Generating chef JSON and uploading...
==> default: Running chef-solo...
==> default: [2015-12-08T15:35:05+00:00] INFO: Forking chef instance to converge...
==> default: Starting Chef Client, version 11.6.0
==> default:
==> default: [2015-12-08T15:35:05+00:00] INFO: *** Chef 11.6.0 ***
==> default: [2015-12-08T15:35:09+00:00] INFO: Setting the run_list to ["recipe[proxy::default]", "recipe[
apache2::default]", "recipe[pixel::default]", "recipe[iptables::default]"] from JSON
==> default: [2015-12-08T15:35:09+00:00] INFO: Run List is [recipe[proxy::default], recipe[apache2::defaul
t], recipe[pixel::default], recipe[iptables::default]]
==> default: [2015-12-08T15:35:09+00:00] INFO: Run List expands to [proxy::default, apache2::default, pixe
l::default, iptables::default]
==> default: [2015-12-08T15:35:09+00:00] INFO: Starting Chef Run for localhost
==> default: [2015-12-08T15:35:09+00:00] INFO: Running start handlers
==> default: [2015-12-08T15:35:09+00:00] INFO: Start handlers complete.
==> default: Compiling Cookbooks...
==> default: [2015-12-08T15:35:14+00:00] INFO: Ignoring apache2::mod_authn_core. not available until apach
e 2.4
==> default: Converging 94 resources
==> default: Recipe: proxy::default
==> default:   * template[/etc/yum.conf] action create
==> default: [2015-12-08T15:35:15+00:00] INFO: template[/etc/yum.conf] backed up to /var/chef/backup/etc/y
um.conf.chef-20151208153515
==> default: [2015-12-08T15:35:15+00:00] INFO: template[/etc/yum.conf] updated file contents /etc/yum.conf
==> default:
==> default:     - update content in file /etc/yum.conf from a403d7 to d6f0c2
==> default:         --- /etc/yum.conf  2013-12-17 12:27:03.000000000 +0000
==> default:         +++ /tmp/chef-rendered-template20151208-2654-1mlblqu       2015-12-08 15:35:15.041505
798 +0000
```

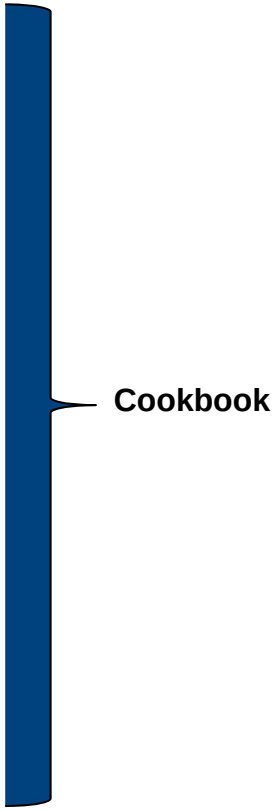
Cookbook

```
[maxher@pc291-vm1 tmp]$ ls apache2/
attributes    CONTRIBUTING.md  Gemfile     LICENSE       recipes    test
Berksfile     definitions      Guardfile   metadata.rb   spec       TESTING.md
CHANGELOG.md  files            libraries   README.md     templates
[maxher@pc291-vm1 tmp]$
```



Administration mit Chef - Cookbooks

- Metadata = Abhängigkeiten, Author, Changelog…
- Recipe = Installations/Konfigurationsanweisungen
- Template = Vorlage (z.B. Konfigurationsdatei)
- File = Statische Datei (z.B. Readme)
- Resource provider = Resources erweitern (Chef DSL)
- Library = Eigene Resources hinzufügen (Ruby)
- Attribute = Eigenschaften von Nodes



Administration mit Chef - Cookbooks

[

- Runlist = Nach Ausführungszeitpunkt sortierte Liste mit Roles/Recipes

- Environment = dev/stage/prod sauber Trennbar

- Data bag = Globale Variablen, gespeichert als JSON (können optional verschlüsselt werden)

]

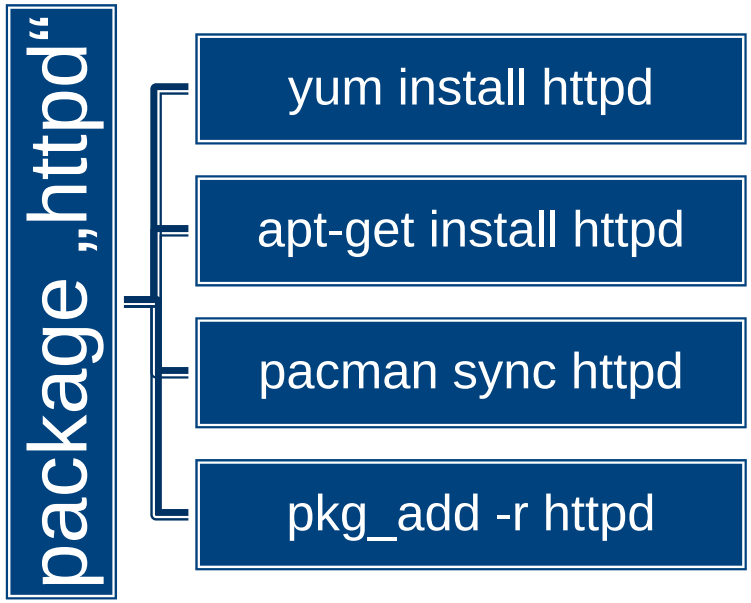
- Resources sind „Anweisungen“ für Aktionen, Verwaltung von:
  - Verzeichnissen,
  - Paketen,
  - User/Gruppen,
  - Netzwerk,
  - Software-RAIDs,
  - Routen,
  - Services,
  - Cronjobs
  - …..
- Änderungen nur wenn nötig

Beispiel:

```
include_recipe postfix::connector

%w{epel-release httpd php5 mysql-client}.each do |pkg|
  package pkg do
    action :install
  end
end

service “httpd“ do
  supports :restart => true, :reload => true
  action [:enable, :start]
end

hostnames = search("node", "role:webserver")
case node['platform_family']
when 'debian‘
  template “/etc/httpd/httpd.conf“ do
    source “httpd_conf.erb“
    owner “root“
    group “root“
    mode “0755“
    notifies :restart, “service[httpd]“, :delayed
    variables(
      :hostname => hostnames,
      :port => ‚443‘
    )
  end
when 'rhel'
  Chef::Log.info('Hey I'm #{node[:
```

Resource / Name / Parameter

- Resources plattformunabhängig (Provider)



- Es kann mit Ruby-Code gearbeitet werden!
- Ein Cookbook kann mehrere Recipes haben (Client/Server-Config)
- Cookbooks können Databags bearbeiten

Chef vs. Puppet

- **Cookbook (Chef) vs. Manifest (Puppet)**
- **Festlegen der Reihenfolge im Cookbook (Chef), imperativ vs. Soll-Zustand in Manifest beschreiben (Puppet), deklarativ**
- **Verschlüsselte Data bags (Chef) vs. Drittanbieterplugin notwenig (Puppet)**
- **Konfigurationsdateien „überbügeln“ (Chef) vs. Konfigurationsdateien anpassen (Puppet)**
- **Tools für Test-driven Development mitgeliefert (Chef) vs. Plugins nötig (Puppet)**

Egal welches Tool, Konfigurationsmanagement lohnt sich!

Demo

Chef run:

- Apache Webserver
- Virtual Host für www.netexpress.de
- www.netexpress.de Dateien
- Firewalld-Konfiguration

Chef – Mehr?

→ **Learn Chef** *(kostenfreie Test-VMs für Tutorials)*

https://learn.chef.io/

→ **Just enough Ruby for Chef**

https://docs.chef.io/ruby.html

→ **Chef Resources**

https://docs.chef.io/resources.html

→ **Vagrant**

https://www.vagrantup.com/

→ **Vagrant Baseboxes**

http://www.vagrantbox.es

→ **NixOS**

http://nixos.org/

NETexpress Network Solutions GmbH
Lochhamer Schlag 17
D-82166 Gräfelfing

Tel.: +49/89/8 98 68-400
Fax: +49/89/8 98 68-444

info@netexpress.de
www.netexpress.de

© 2018 NETexpress GmbH
Ein Unternehmen der PIXEL Group