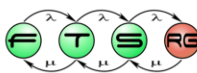


# Automatikus infrastruktúra menedzsment és alkalmazástelepítés

Szatmári Zoltán



Utolsó módosítás: 2013.04.08.

## DEMO Egyszerű alkalmazás telepítése

- Telepítés kézzel
  - „Release unit”
  - Másolás utáni egyéb beállítások: pl. registry
- Telepítő script
  - Telepítés automatizálása
  - Általános és gép/felhasználó-specifikus beállítások szétválasztása
- Felügyelet??
  - Hova, mikor, ki, miért telepítette
  - Most pontosan mi is van fent?
  - Milyen infrastruktúrával tudok számolni?

# Tartalom

- Automatikus konfigurációkezelés
  - Környezetfüggő konfiguráció
  - Dinamikus konfigurációk

# Motiváció

- Nagyméretű infrastruktúra menedzsmentje
  - Központosított megoldás
- Hasonló konfigurációs igények, ismétlődő feladatok
  - Sablon alapú technológia
- Automatikus alkalmazástelepítés
  - Felügyelő és beavatkozó komponensek
- Dinamikus infrastruktúra menedzsment
  - Automatikus igény szerinti alkalmazás telepítés

# Konfigurációmenedzsment eszköz

- Lehetővé teszi
  - Konfigurációs beállítások deklaratív megadását
  - Igény esetén a konfiguráció elvégzését
- Legtöbbször tartalmaz valamilyen CMDB megoldást
  
- Technológiák
  - CFEngine
  - Puppet
  - **Chef**
  - stb.

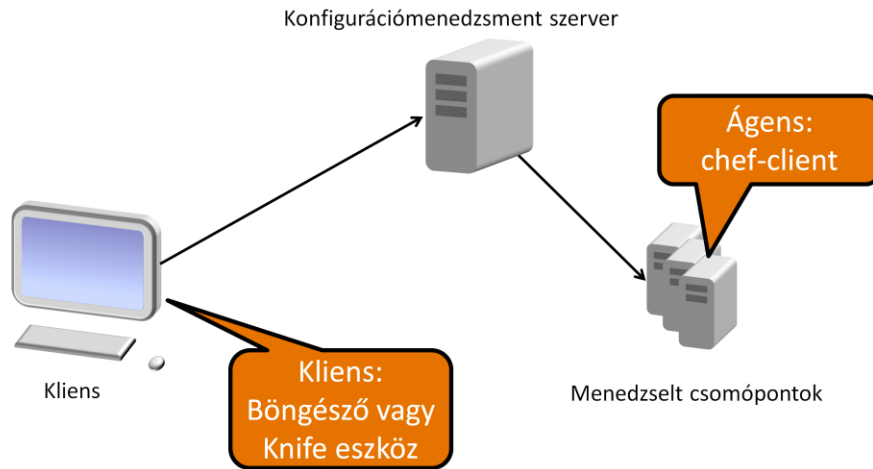
## Konfigurációmenedzsment eszköz

- Jellemző komponensek:
  - Központi konfigurációmenedzsment szerver
  - Menedzselt csomópont ágens
- Hol történik a döntés a végrehajtandó lépésekről?
  - Puppet: a szerver kész konfigurációs lépéseket állít elő
  - CFEngine, Chef: az ágens hozza meg a döntést

## Chef – főzzünk egy infrastruktúrát

- Infrastruktúra automatizációs megoldás
- Deklaratív konfigurációleírás támogatása
  - Azt mondjuk, mit szeretnénk, nem azt, hogyan
  - Cookbooks, recipes
- Központi infrastruktúra adatbázis
  - Attribútumok, futási listák (run list)

# Chef architektúra





## DEMO Webes alkalmazás telepítése

- Szükséges erőforrások
  - Webszerver (Apache), PHP, stb.
  - Webes alkalmazás
  - Konfigurációs beállítások
- 1 gép esetén kézzel, 10 vagy 100 esetén már automatizáltan



# Deklaratív konfigurációmegadás

- **Recept (recipe)**
  - Erőforrások deklaratív megadása
  - Ruby nyelv
- **Szakácskönyv (cookbook)**
  - Receptek
  - Attribútumok
  - Sablonok
  - Stb.

## Deklaratív konfigurációmegadás

- **Szerep (role)**
  - Receptek felsorolása
- **Csomópont (node)**
  - Szerepek
  - Receptek
  - Attribútumok

## Receptek

```
package "apache2"  
package "apache2-mpm-prefork"  
  
a2enmod "ldap" do  
  file "ldap.load"  
  notifies :reload, "service[apache2]"  
end  
  
service "apache2" do  
  supports :status => true, :restart =>  
    true, :reload => true  
  action :enable  
end
```



## Fontosabb erőforrások

- Csomag
- Felhasználó
- Csoport
- Cronjob
- SVN repository
- Mount
- IPConfig

## Fontosabb erőforrások

- Fájlok
  - Cookbookban definiált
  - Távoli URL-en elérhető
- Könyvtárak
  - Cookbookban definiált

## Fontosabb erőforrások

- Sablonok
  - Paramétrezhető fájlok
  - Különböző típusú paraméterek
    - Egész érték
    - String érték
    - Objektum

```
# ports.conf by chef
<% @ports.each do |l| -%>
Listen <%= l %>
<% end -%>
```

## Fontosabb erőforrások

```
template "/etc/apache2/ports.conf" do
  source "apache2/ports.conf.erb"
  mode 644
  owner "root"
  group "root"
  variables(
    :ports => node[:apache2][:ports]
  )
  notifies :reload, "service[apache2]"
end
```



# Változók

- Mitől lesz ez testre szabható?
- Változók definiálása
  - Cookbook szinten
  - Role szinten
  - Node szinten

- Attribútum beállítása
  - Role-tól függően
  - Node-tól függően
  
- Pl.:
  - HTTP port beállítása
  - Alkalmazás paramétereinek beállítása

Role, node

## Chef search

- Mitől lesz környezetfüggő a konfiguráció?
- Konfiguráció adatbázisban információk vannak a hosztokról
  - IP cím
  - Hoszt neve
  - Lefuttatott receptek
  - OS típus, verzió
  - Stb.

## Chef search

- CMDB információk felhasználása
  - Receptekben
  - Sablonokban
- Pl.:

```
hosts =  
  search(:node, "recipes:irf::app")  
  .map { |n| webhost_data(n) }
```

## DEMO Webes terheléselosztó beállítása

- Egyszerű HTTP proxy (HAProxy)
  - Telepítés
  - Konfigurálás
    - Működő webserverek felsorolása

knife search node "role:webserver,,

## Modern, skálázható alkalmazások

- Cloud környezet
- Igény szerinti rendszerkonfiguráció
  - Pl.: Terhelés függő webserverek mennyisége
  - VM gyorsan igényelhető és eldobható
- Alkalmazásnak is támogatnia kell
  - Állapotmentes komponensek
  - Laza csatolás
  - Minden komponens kívülről konfigurálható
- Lásd félév második felében: Virtualizáció és Cloud

## DEMO AWS EC2

- EC2 Instance indítása
  - Instance létrehozása
  - Chef bootstrap
  - Szerepek beállítása



23



```
knife ec2 server create --groups=all --ssh-user=ubuntu --ssh-key=irf --identity-  
file=~/.irf.pem
```

## Állapotgép alapú megközelítés

- Visszafele is működik?
  - El tudom távolítani a telepített erőforrásokat?
  - Le tudom állítani a szolgáltatásokat?
  
- Mi történik, ha valami futás közben változik?
  - Hiba lép fel?
  - Túlterhelés következik be?



# Motiváció

## Adottságok

- Dinamikusan változó terhelés
  - időszakos
  - tervezett
- Konfigurációk telepítése/karbantartása
- Hibás viselkedés automatikus észlelése

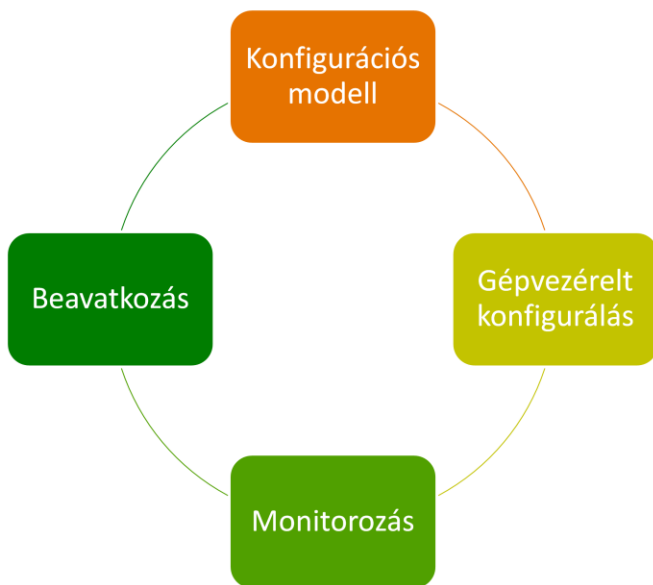
## Szükséges támogatás

- Jól skálázható (scalable)
  - alkalmazások
  - infrastruktúra
    - Cloud
- Gép által vezérelt megvalósítás
- **Monitorozás**

# Megközelítések

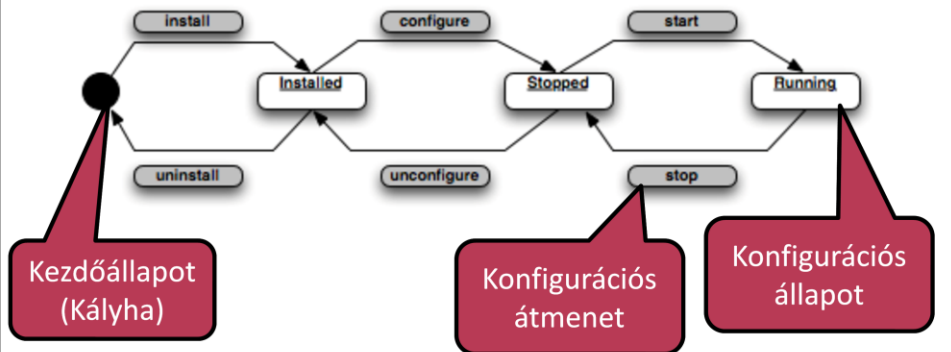
- **Állapotgép alapú megközelítés**
  - Eszköz példa
    - ZooKeeper
    - **GLU (az előadáson ezzel foglalkozunk)** <https://github.com/linkedin/glu>
  - Megvalósítás alapja
    - A szolgáltatások konfigurációjának állapotgépként való leírása
  - Mikor jó
    - Statikus és dinamikus konfigurációra is
    - Statikusnál jelentős lehet az overhead
  
- **Célkonfiguráció deklarálása és állapot fenntartása**
  - Eszköz példa
    - Puppet <http://www.puppetlabs.com/>
  - Megvalósítás alapja
    - Erőforrásháló alapján
  - Mikor jó
    - Viszonylag statikus konfiguráció

# A konfigurációs rendszer főbb feladatai



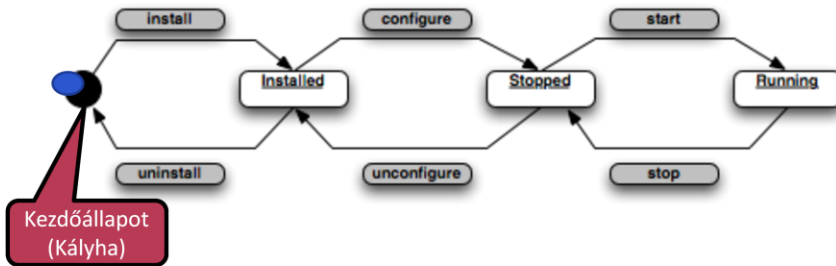
# Állapotgép alapú megoldás – GLU

- Konfigurációs állapotgép  
= egy véges állapotgép (Finite State Machine)



# Véges állapotgép

- 1 token van a rendszerben
- Nem keletkeznek és nem is tűnnek el tokenek
- A token jelöli ki az aktuális állapotot
- (Digitből ismerősnek kell lennie)



## Miért jó ez az egész?

- Automatikusan tudunk telepíteni 10, 100, 1000... gépet
- Ha ügyesen írjuk meg a szkripteket, akkor szinte autonóm rendszert kapunk
- Hol használják?
  - GLU
    - LinkedIn
  - Chef
    - Amazon EC2
    - Stb.

## További információ

- [Chef: systems integration framework](http://www.opscode.com/chef)
- [GLU: Deployment Automation Platform](https://github.com/linkedin/glu)



<http://www.opscode.com/chef>  
<https://github.com/linkedin/glu>