PEOPLE COME FIRST
INFORMATIKAI SZAKÉRTŐK EGYESÜLETE

Nagyvállalati rendszerek

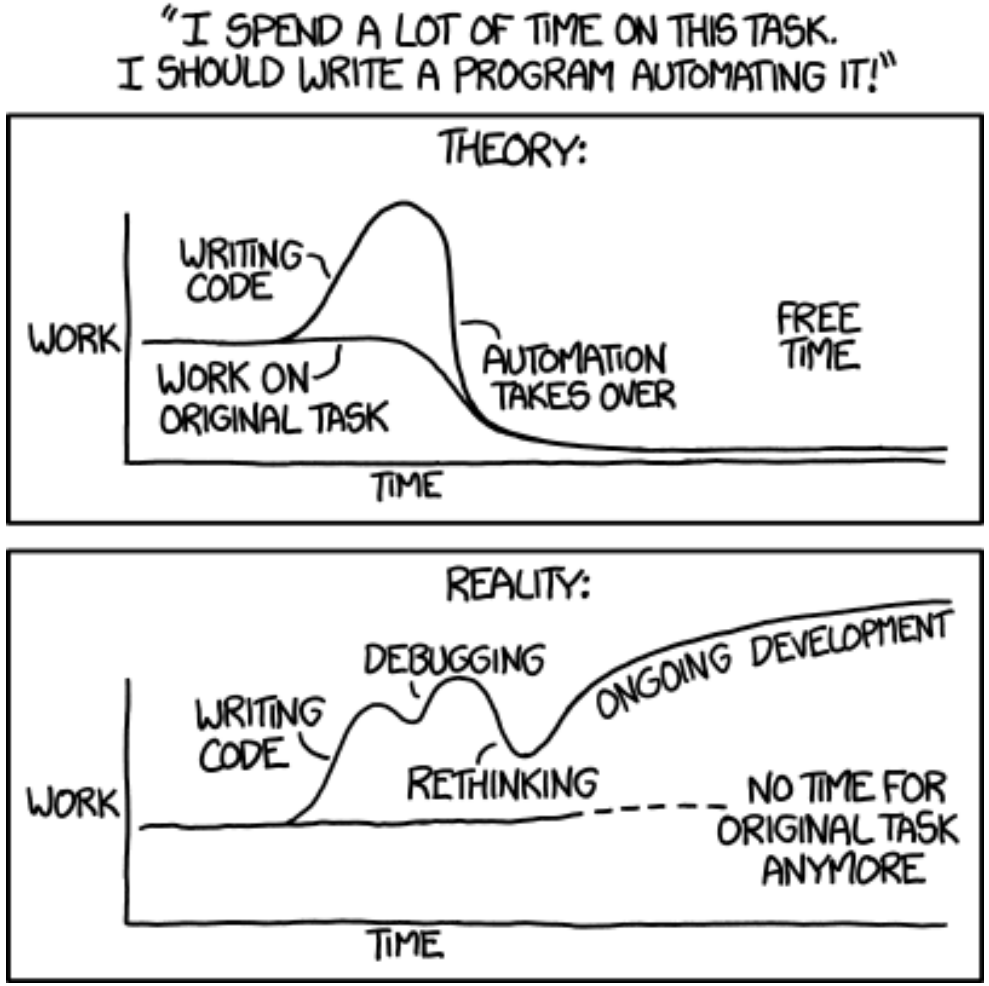
Tematika

- Miért érdemes automatizálni?
- Mit érdemes automatizálni?
- Hogyan és mivel automatizáljunk
- Infrastruktúra automatizáló eszközök
- Saltstack

Miért érdemes automatizálni?

- Munkaidőt lehet általa spórolni
  - Más feladatokkal többet lehet foglalkozni
- Gyorsabbá válhat tőle egy folyamat
  - Next, next, finish…
- Csökkenthető az emberi hiba előfordulási valószínűsége

Automatizálás



Mit érdemes automatizálni?

- Időigényes feladatokat
  - Adatbázis mentés
- Gyakori feladatokat
  - Log fájlok karbantartása
  - Frissítések telepítése
- Bonyolult feladatokat
  - Riportok
- Rutin feladatokat

Automatizálási eszközök

- Legelterjedtebb: scriptek
- Monitorozó eszközök
- Workflow Engine-ek
- Alkalmazáson belüli képességek
- Ütemező szoftverek
  - Jellemzően scriptekkel kombinálva
- Dokumentáció

Egy folyamat automatizálásának lépései

- Előfeltétel: a folyamat ismerete
- Első lépés: a folyamat dokumentálása
- 3 fázis: futtatási előtt, közben és után
  - Kell-e értesítés indulásról/hibáról stb.
- Legyen részfeladatokra bontva
- A feladathoz illeszkedő eszközt használjunk
- A lehető legkevesebb eszközt használjuk

Infrastruktúra automatizálás

- Jellemzően hardware felett kezdődik
  - Cloud szolgáltatóknál van hardware-re is
- Operációs renszer telepítése
- Szoftverek telepítése és beállítása
- Felvétel monitorozásba

OS telepítés automatizálása

- OS image-ek használata
  - Virtuáls gépeknél VM sablon/klónozás
  - Fizikai gépeknél OS képfájl “terítése”
- PXE Network Boot
- Bizonyos operáció rendszerknél beépített mechanizmussal
  - CoreOS - Ingition

Élet az OS telepítés után

- Jellemzően “A” telepítő script segítségével
- Probléma 1: különböző disztribúciók kezelése
  - httpd vs apache2
- Probléma 2: módosításra nem alkalmas
  - Csomag verzió váltásért ne kelljen OS újratelepítés

Infrastructure as Code

- Nem csupán a szoftver, az infrastruktúra is “kódból” álljon elő
- Valamilyen deklaratív módon legyen leírva az infrastruktúra
- Legyenen eszközök arra, hogy ebből a deklaratív formátumból előállítható legyen a kívánt rendszer

Megoldás: deklaratív eszközök

- Azt mondjuk meg, mit szeretnénk
  - “legyen feltelepítve X csomag”
- A keretrendszer kideríti, hogyan kell, és meg is csinálja
- Állapot alapú megközelítés
  - Csak akkor módosítsunk, ha szükséges
  - Kell mechanizmus az állapot ellenőrzésére

Javasolt eszközök

- Puppet (Ruby)
- Chef (Ruby)
- Ansible (Python)
- SaltStack (Python)

SaltStack

- Eredetileg távoli utasításvégrehajtásra
  - cmd.run modul
- YAML formátumban megadható állapotok
- Master/Minion komponensek
  - De van masterless és minionless mód is
- Könnyen bővíthető (Python)

Salt Minion

- A célgépen fut
- A Mastertől megkapott állapotok/parancsok alapján dolgozik
- A feladatok eredményét visszaküldi a Masternek

Salt Master

- Ide csatlakoznak be a Minionok
  - Titkosított forgalom (salt-key)
- Központi event bus
  - Minden Minion lájta
  - Scope: targeting alapján
- Állapotleírók jellemzően helyben
  - Van lehetőség pl. távoli git repository használatára

SaltStack működési módok

- Master/Minion
- Masterless
  - Kézzel kell a leírókat hordozni, de nincs szükség service-ekre
- Minionless
  - SSH-n keresztül másolódik át a szükséges információ a célgépre
  - Jelentős átfedés az Ansible-el

Salt Grains

- A gépről a Minion által szolgáltatott információk
- Lehet dinamikus (rendszeridő)
- Lehet statikus (disztribúció neve)
- Saját magunk is definiálhatunk Graineket
  - Tipikus felhasználás: gépek címkézése (csoportosítás)

Salt Targeting

- A futtatott parancs “scope”-ját határozza meg (mely minionokra legyen érvényes)
- Mindenkire: “*”
- Adott gépre: “hostname.domainname”
- Listára: “host1,host2”
- Grain alapján:
  - Debian gépekre: “G@os:Debian”
  - x86-64-es gépekre: “G@cpuarch:x86_64”

Salt State

- Elvárt állapot leírására szolgál
- YAML formátum
- Futtatás deklarációs sorrendben
  - De: függőségeket is tud kezelni
- Újra felhasználhatóak (include stb.)

Salt State példa

```
/etc/httpd/extra/httpd-vhosts.conf:
  file.managed:
    - source: salt://webserver/httpd-vhosts.conf
apache:
  pkg.installed: []
  service.running:
    - watch:
    - file: /etc/httpd/extra/httpd-vhosts.conf
    - require:
    - pkg: apache
```

Sablonok: Jinja2

- Fájlok tartalmának dinamikus előállítására
- YAML feldolgozás előtt fut
- State-ek generálására is használható
- Tipikus felhasználás: konfigurációs fájlok előállítása

Salt Pillar

- Master oldali kulcs-érték tároló
- Fájl alapú (de: lehet pl. Redis-t használni)
- YAML formátum

Jinja példa

```
motd:
  file.managed:
  {% if grains['os'] == 'FreeBSD' %}
    - name: /etc/motd
  {% elif grains['os'] == 'Debian' %}
    - name: /etc/motd.tail
  {% endif %}
    - source: salt://motd
```

```
motd:
  file.managed:
    - name: /etc/motd.tail
    - source: salt://motd
```

Salt Formula

- Előre megírt State-ek, jelentős sablon használattal
- https://github.com/saltstack-formulas

Salt Mine

- A Minionok csak a saját Grainjeiket érik el
  - Ha más Minionról származó adat kell: Salt Mine
- Master oldalon tárolódik, innen kérdezik le a Minionok
- A Master meghatározott időközönként frissíti az információt

Salt Mine

- A Minionok csak a saját Grainjeiket érik el
  - Ha más Minionról származó adat kell: Salt Mine
- Master oldalon tárolódik, innen kérdezik le a Minionok
- A Master meghatározott időközönként frissíti az információt