UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Programrendszerek fejlesztése CI/CD Jenkins/Docker/Ansible

Dr. Bilicki Vilmos
Szegedi Tudományegyetem
Informatikai Tanszékcsoport
Szoftverfejlesztés Tanszék

A mai előadás tartalma

- 12 faktoros alkalmazás
- Ansible
- Jenkins
- CI/CD alapok
- CI cső
- Automatikus Elfogadás Tesztelés
- CD cső
- Jenkins + Docker Swarm

Vas korszak és Felhő korszak

- Vaskorszak (megbízható HW megbízhatatlan SW)
  - Az erőforrások hardverhez kötöttek
  - Az infrastruktúra kezelése manuális munka
  - A változások munkaigényesek és hetekbe telnek
  - A folyamatok sokszor csak papíron vannak meg
- Felhő korszak (megbízhatatlan HW megbízható SW)
  - Virtuális erőforrások
  - A rutinfeladatok szoftverrel megoldhatóak
  - A változások másodpercek alatt megvalósíthatóak

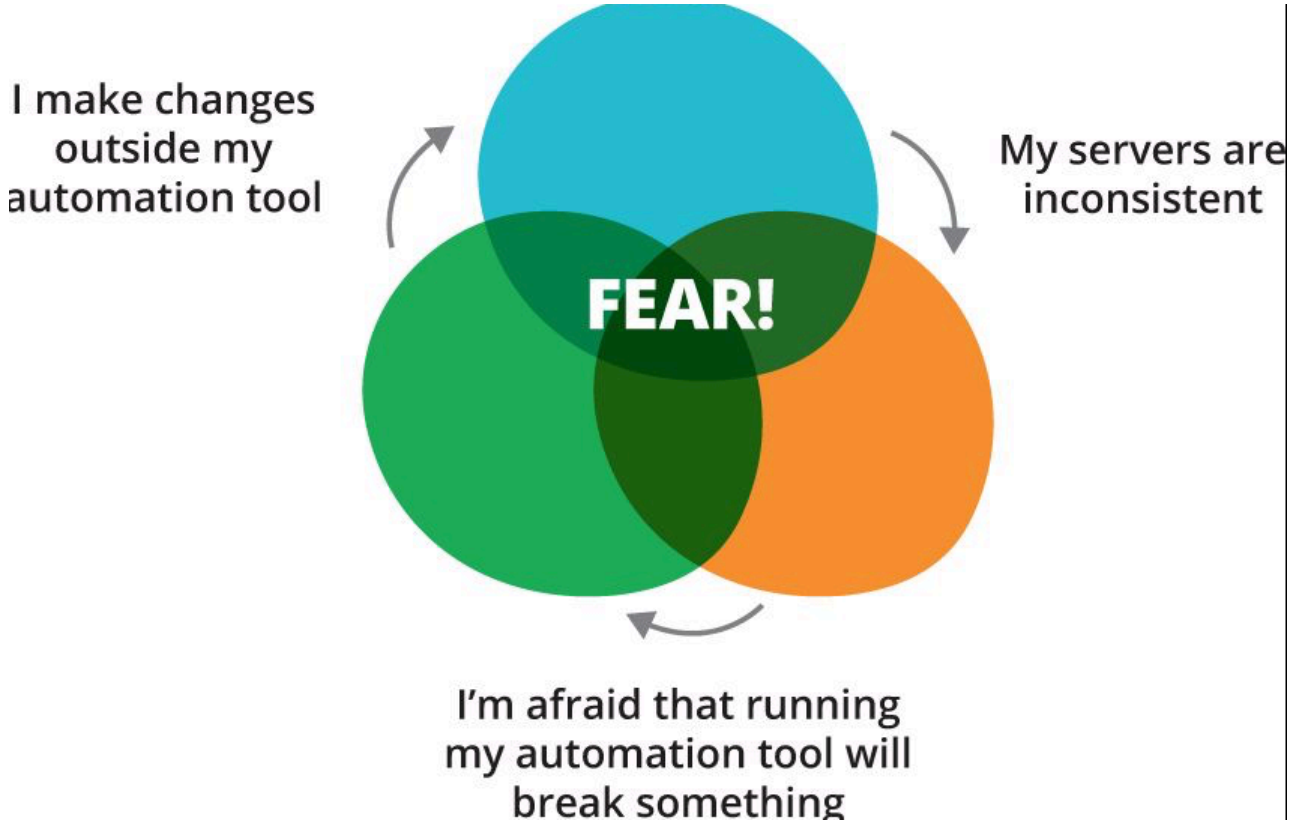
Kihívások

- Technológiák
  - Virtualizáció
  - Felhő
  - Konténerek
  - Szerver automatizáció
- Elvileg
  - IT feladatok egyszerűsítése
    - Erőforrások: létrehozása, törlése, menedzsmentje
    - Problémák gyorsabb fellelése és megoldása
    - Rutinfeladatokra fordított idő csökkenése
- Gyakorlatilag
  - Az erőforrások lérehozásának egyszerűsítése azok elszaporodásához vezet
  - Változások kezelése inkonzisztens rendszerekhez vezet
  - Klasszikus folyamatok, módszerek megmaradtak

Problémák

- Szerverek elszaporodása
  - Egyszerűen gyorsan lehet szervert létrehozni
  - Frissítése?
- Konfigurációs csúszás
  - Konzisztens kiindulási szerverekkel (majd adott hibajavításhoz adott szervert kiegészítenek)
  - Idővel más-más szervereken más-más igény lesz (pl.: Oracle adott Java, Jira más Java)
  - 3 rendszergada 3 féle módon telepít egy IIS-t
  - Egy szervernek agy a terhelése és ezt elkezdik tuningolni
- Hópehely szerverek
  - Minden más szervertől különbözik
  - Speciális, nem replikálható
  - A tudás nincs menedzselve (ne nézz rá, ne érintsd meg)
- Törékeny infrastruktúra
  - Könnyen meghibásodik és nehéz javítani
- Erózió: javítások, merevlemez betelése, ...

Automatizálás aggályok



Infrastruktúra mint kód

- Infrastruktúra automatizálási megközelítés ahol a szotfverfejlesztés legjobb gyakorlatát alkalmazza
- IaC szoftver fejlesztés csak ehhez megfelelő nyelven
- Konzistens, megismételhető rutinok
- A változások a definíciókat érintik és azok lesznek kitelepítve
- Úgy kezeljük az infrastruktúrát mintha szoftver lenne
  - Verziókövető rendszerek
  - Automatizált teszt könyvtárak
  - Teszt vezérelt fejlesztés
  - Folyamatos integráció
  - Folyamatos kibocsájtás
- Ahol alkalmazzák: Google, Amazon, Netflix, ...

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Infrastruktúra mint kód

- Az IT infrastuktúra nem gátolja, hanem támogatja a változásokat
- A változások rutinfeladatok, nem stresszes epizódok
- Az IT csapat nem foglalkozik rutin és repetatív feladatokkal
- A felhasználók kezelik az erőforrásokat az IT csapat támogatása nélkül
- A csapatok könnyen megoldják a hibákat, nem aza feltételezés, hogy hiba megelőzhető
- A javítások folyamatosan telepítve vannak, nem nagy és kockázatos nagy kibocsájtások formájában
- A problémákra a megoldások a megvalósítás és tesztelés által vannak kipróbálva, nem megbeszéléseken, dokumentumokon keresztül

Infrastuktúra mint kód

- Nem csak felhő környezetben
  - Fejlesztő gépén
  - Fejlesztő szerveren
  - Staging
  - Production

Elvek

- Az infrastruktúra bármelyik eleme gyorsan és kicsi ráfordítással reprodukálható
- A rendszerek kidobhatóak (vas vs. felhő)
  - Bárány és nem házikedvenc
- A rendszerek konzisztensek
- A folyamatok megismételhetőek
  - Erős szkriptelős kultúra
- A terv folyamatosan változik

Gyakorlatban

- Használjunk definíciós fájlokat
- Öndokumentáló folyamatok
- Mindent verziózni kell
  - Követhetőség
  - Visszaállítás
  - Korreláció (hibakeresés)
  - Láthatóság
  - Reakcióképesség
- Folyamatos tesztelés
- Kicsi változások a nagyok helyett
  - Kisebb munka kevesebb tesztelés is elegendő
  - Ha nem sikerül egyszerűbb visszaállítani
  - Egyszerűbb a hibát is lokalizálni
  - Nagy változást kis hiba is megállíthat
  - Javítások és új funkciók érkezése motiváló, nagy visszalépések demotiválóak
- Folyamatos működtetés

```
server: dbnode
  base_image: centos72
  chef_role: dbnode
  network_segment: prod_db
  allowed_inbound:
    from_segment: prod_app
    port: 1521
  allowed_inbound:
    from_segment: admin
    port: 22
```

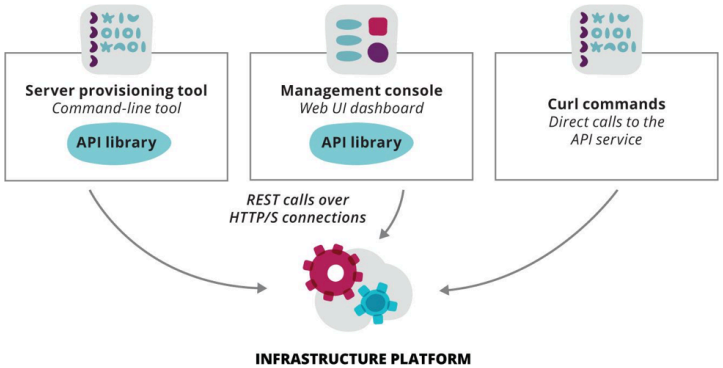
```
79  def testTask(testName, label=null, extraEnv=null) {
80    def needsXunit = testName != "test-docker-py"
81    return this.makeTask(label ?: "docker", testName, extraEnv ?: []) {
82      // Need to refresh timestamps else junit archiver will die.
83      sh "find 'bundles/${this.versionString}/${testName}' -type f -print0 | xargs -0 touch"
84      if (needsXunit) { this.go2xunit(testName) }
85      this.junitArchive(testName)
86    }
87  }
88
89  def integrationTask(label) {
90    return this.testTask("test-integration-cli", label, ["CI_TASK=test-integration-cli/JENKINS_LABEL=${label}"])
91  }
```

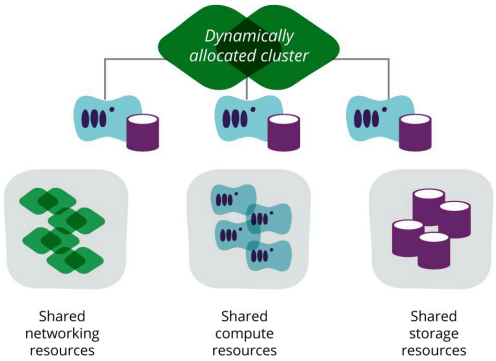
Robosztus infrastruktúrán túl

- Nem törékeny infrastruktúra
- Az incidensek hatása a rendszerre:
  - Javítás, fejlődés
  - Amikor valami elromlik akkor nem simán a kijavítása a cél, hanem a rendszer átalakítása, hogy a hasonló esetekkel a jövőben könnyebben megbirkózzon
- A rendszer bármely eleme kis erőbefektetéssel , gyorsan újraépíthető
- Minden rendszere elem naprakész, minden javítást tartalmaz
- A normál infrastruktúra igények a működtető személyzet nélkül teljesíthetőek
- A javítási ablakok ritkák, vagy nincsennek (minden változás munkaidőben normál ügymenet mellett történik)
- A csapat naplózza a hibaelhárításhoz szükséges időt
- A csapat érzi, hogy értéket teremt

Dinamikus Infrastuktúra platformok

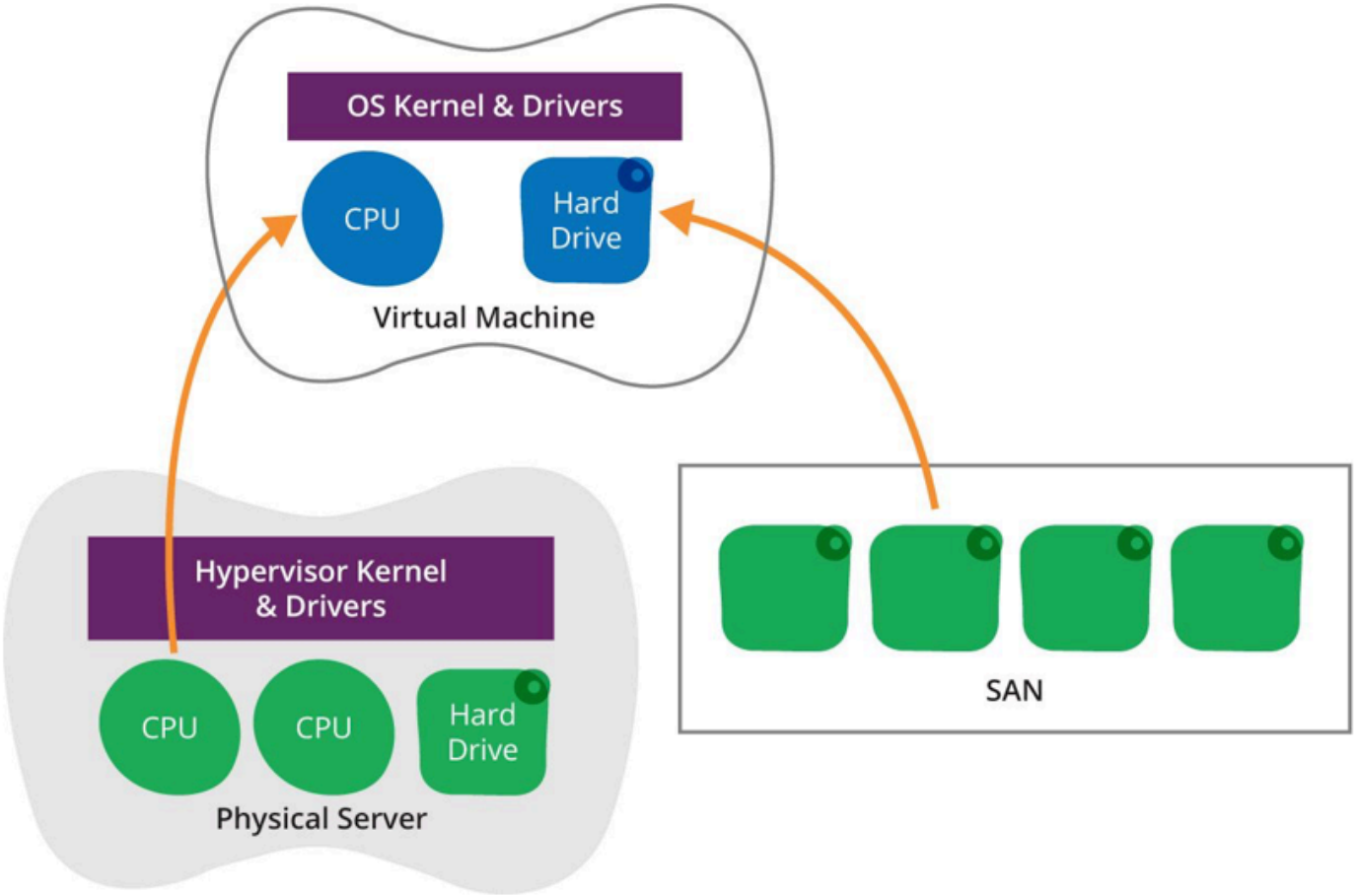
- Az erőforrások (számítás, hálózat, tároló, ...) programozható felületen is elérhetőek
- Követelmények
  - Programazható
  - Igény szerinti
  - Önkiszolgáló






```
require 'aws-sdk'

@client = Aws::EC2::Client.new(:region => 'eu-west-1')

@client.run_instances({
  image_id: 'ami-903686e7',
  instance_type: "t1.micro",
  min_count: 1,
  max_count: 1
})

@client.describe_instances()[:reservations].each { |reservation|
  reservation[:instances].each { |instance|
    puts "Instance #{instance.instance_id} is #{instance.state.name}"
  }
}
```

Értenünk kell a lehetőségeinket



Infrastruktúra definiáló eszközök

- GUI vs API
  - Szkriptelhető felület
  - Felügyelet nélküli mód
    - Akár szabványos bevitelen akár máson is meg lehessen szólítani
    - Struktúrált kimenet
    - Jelszó, ... automatizálható
  - API jellemzők
    - Idempotens
    - Futás előtti ellenőrzés
    - Utólagos ellenőrzés
    - Látható hiba
    - Paraméterezhető
  - Külső konfiguráció
  - Verziókezelés

```
echo "spock:*:1010:1010:Spock:/home/spock:/bin/sh" \
    >> /etc/passwd
```

```
user { "spock":
  ensure => present,
  gid    => "science",
  home   => "/home/spock",
  shell  => "/bin/sh"
}
```

```
user "spock"
  state active
  gid   "science"
  home  "/home/spock"
  shell "/bin/sh"
```

```
aws_instance: web_server
  name: web-${var.environment}
  ami: ${var.web_server_ami}
  instance_type: ${var.instance_type}
```

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Létrehozás, Provisionig

- Erőforrások csatolása az elemhez
- Elem létrehozása
- Szoftverek telepítése
- Elem konfigurálása
- Az elem regisztrálása az infrastuktúrában

Két megközelítés

- Procedurális szkriptek

```
# spin qa app
```

```ruby
require 'aws-sdk'
require 'yaml'

usage = 'spin <qa|stage|prod> <web|app|db>'

abort("Wrong number of arguments. Usage: #{usage}")
unless ARGV.size == 2
environment = ARGV[0]
server_type = ARGV[1]

#
# Set subnet_id based on the environment defined on
the command line.
# The options are hardcoded in this script.
subnet_id = case environment
when 'qa'
  'subnet-12345678'
when 'stage'
  'subnet-abcdabcd'
when 'prod'
  'subnet-a1b2c3d4'
else
  abort("Unknown environment '#{environment}'.
Usage: #{usage}")
end
```

```ruby
#
# Set the AMI image ID based on the server type specified on the command
line.
# Again, the options are hardcoded here.
image_id = case server_type
when 'web'
  'ami-87654321'
when 'app'
  'ami-dcbadcba'
when 'db'
  'ami-4d3c2b1a'
else
  abort("Unknown server type '#{server_type}'. Usage: #{usage}")
end

#
# Use the AWS Ruby API to create the new server
ec2 = Aws::EC2::Client.new(region: 'eu-west-1')
resp = ec2.run_instances(
  image_id: image_id,
  min_count: 1,
  max_count: 1,
  key_name: 'my-key',
  instance_type: 't2.micro',
  subnet_id: subnet_id
)

#
# Print the server details returned by the API
puts resp.data[:instances].first.to_yaml
```

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Két megközelítés

- Deklaratív szkriptek
- Nem cselekvés, hanem az eredmény van leírva
  - A hogyan a rendszer dolga (mi van már meg, ...)

```
variable "environment" {
  type = "string"
}

variable "subnets" {
  type = "map"

  default = {
    qa = "subnet-12345678"
    stage = "subnet-abcdabcd"
    prod = "subnet-a1b2c3d4"
  }
}

resource "aws_instance" "web" {
  instance_type = "t2.micro"
  ami = "ami-87654321"
  subnet_id = "${lookup(var.subnets, var.environment)}"
}
```

```
# terraform apply -var environment=qa web_server.tf
```

Infrastruktúra vs Szerver konfiguráció

- Létrehozza szervet és szerepkörét
- A szerver konfigurációját a szerver konfiguráció valósítja meg
  - Konfigurációs nyilvántartás (registry)
    - Dinamikus infrastruktúrában
    - Nem klasszikus CMDB
      - » CMDB audit és javítás anti minta

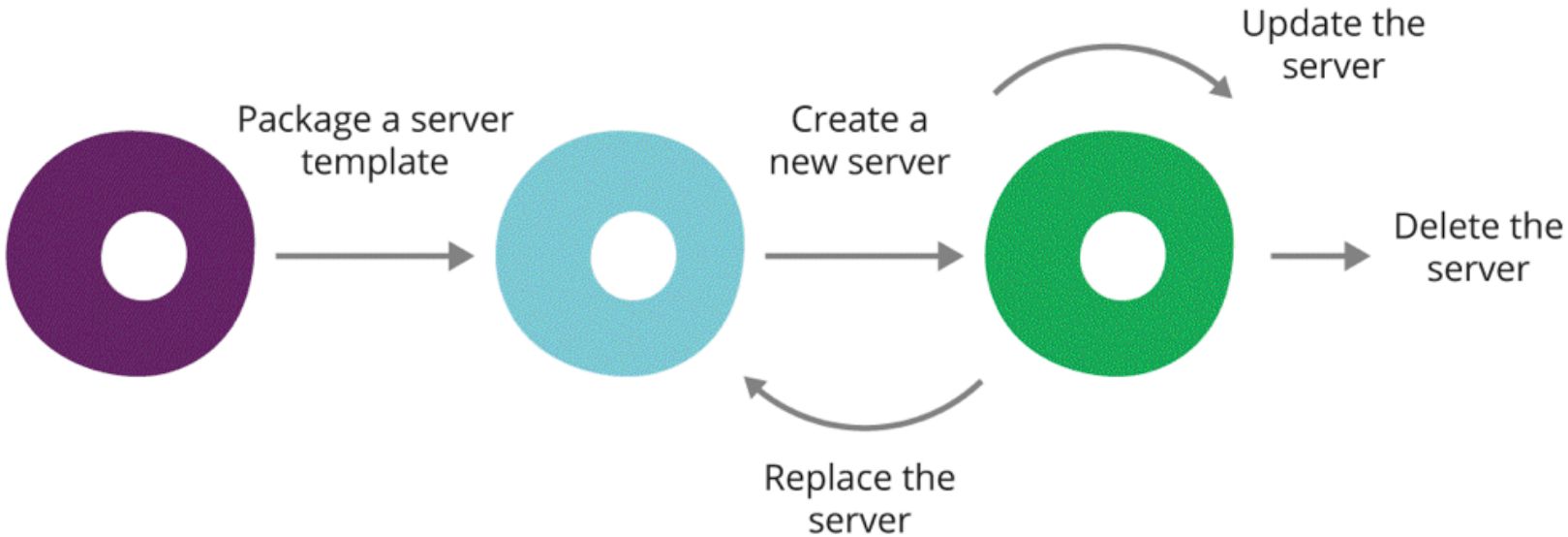
```
resource "aws_instance" "web" {
  instance_type = "t2.micro"
  ami = "ami-87654321"

  provisioner "chef"  {
    run_list = [ "role::web_server" ]
    attributes_json = {
      "dns_servers": [
        "192.168.100.2",
        "192.168.101.2"
      ]
    }
  }
}
```

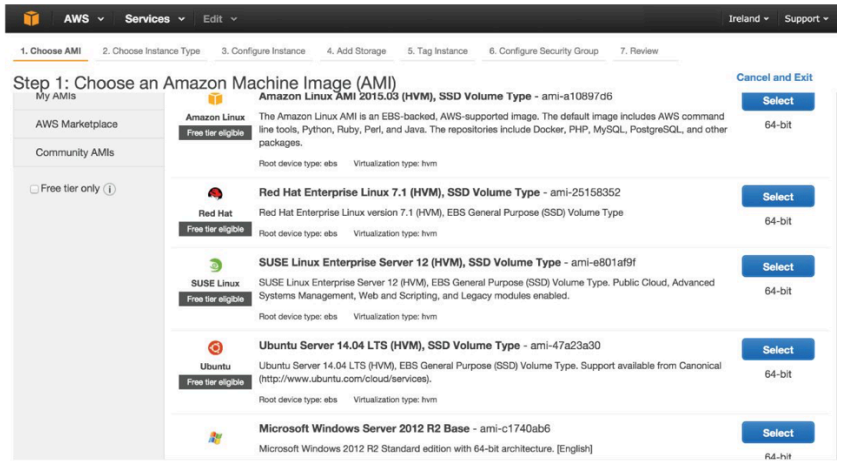
UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Szerver konfigurációs eszközök

- Célok
  - Egy új szerver igény szerint azonnal létrehozható
  - Egy új szerver humán beavatkozás nélkül is létrehozható
  - Amikor egy szerveren változtatni kell akkor ez megtehető humán beavatkozás nélkül
  - A változások minden érintett szerveren megtörténnek, megismételhető, konzisztens, transzparens és öndokumentáló módon
  - A változásokat leíró folyamatoka egyszerű és biztonságos változtatni
  - Automatizált tesztek futnak minden változás után
  - A változások verziózottak

Szever menedzsment életciklus





Konfiguráció kiszervezése

```
default['monitoring']['servers']['sydney'] = 'monitoring.au.myco'
default['monitoring']['servers']['dublin'] = 'monitoring.eu.myco'
```

```
my_datacenter = node['data_center']
template '/etc/monitoring/agent.conf' do
  owner 'root'
  group 'root'
  mode 0644
  variables(
    :monitoring_server => node['monitoring']['servers'][my_datacenter]
  )
end
```

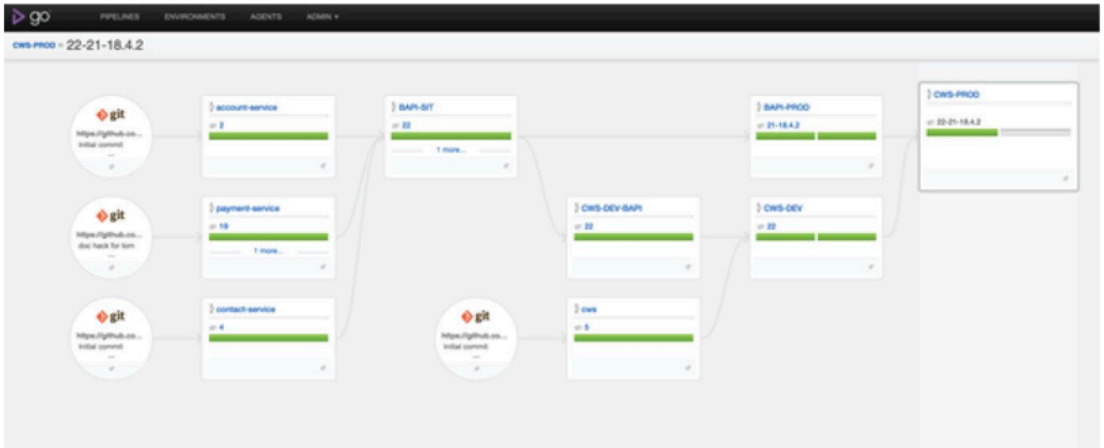
Általános infrastruktúra szolgáltatások

- A konkrét szolgáltatást megvalósító szerveken túl általános elemek is kellenek
  - DNS
  - Monitorozás
    - Állapot
    - Események
  - Riasztások (Információ sugárzó)
  - Naplózás
  - Szolgáltatás fellelés
    - Szerver oldali (LB)
    - Kliens oldali (Szolgáltatás tár)
  - Elosztott folyamat menedzsment
    - Szerver szerepkör
    - Konténerek
  - ...
- Szolgáltatások megosztása csapatok között
  - Dedikált preferáltabb

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

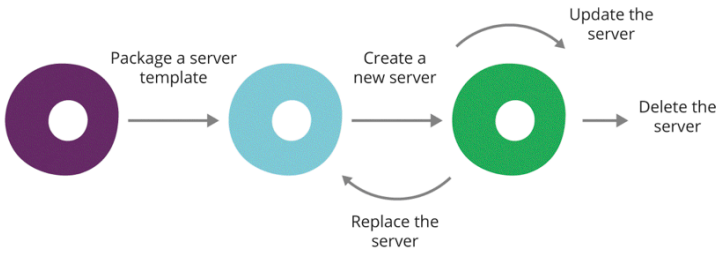
Szoftverek telepítése

- Saját szoftverek tartoznak ide
- Több szervert is érinthet
- Újraindítások kellhetnek
- Van olyan eset amikor manuálisan kell megtenni
  - Hópehely kibocsájtások
  - Ritka kibocsájtások
  - Nem konzisztens környezetek
- CD cső



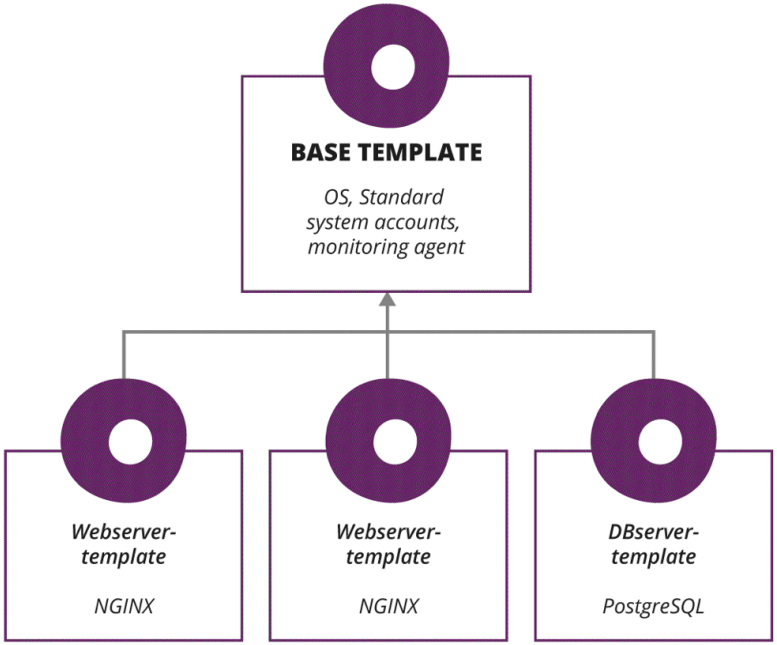
Szerver létrehozás minták

- Tipikus folyamat
  - HW allokáció
  - Szerver példány létrehozása
  - Lemezek partícionálása
  - Operndszer telepítése
  - Szoftverek telepítése
  - Adott elemek konfigurálása
  - Hálózat beállítása
  - Szerver regisztrálása
- Létrehozás
- Módosítás
- Csere (hatása van másra is: DNS, naplózás, ...)
  - Nem változtatható szerver
- Törlés (adatok mentése, ...)
- Egyébb
  - Visszatérés egy hibából
  - Szerver tár átalakítása
  - HW átkonfigurálása



Szerver létrehozás minták

- Területek
  - Szoftver
  - Konfiguráció
  - Adat (a tartalma a konfig. eszközök dolga)
- Szerepkörök, sablonok




```
#!/bin/bash

aws ec2 run-instances \
    --image-id ami-12345678 \
    --count 1 \
    --instance-type t2.micro \
    --key-name MyKeyPair \
    --security-groups open-https-port
```

Szerver létrehozás minták

- Anti minták
  - Kézzel létrehozott szerver
  - Klónozott szerver
  - Hópehely gyár (kézzel futtatja adott paraméterekkel)
- Szerver létrehozás lehetőséges szkriptbe
- Szerver sablonok

```
$ aws ec2 run-instances \
 --image-id ami-12345678 \
 --region eu-west-1 \
 --instance-type t2.micro \
 --key-name MyKeyPair
```

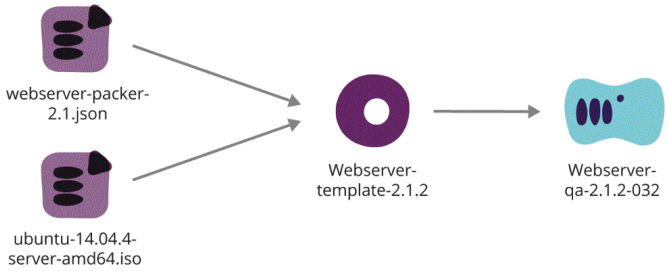
Új szerverek elindítása

- A használatba vétel előtt és a létrehozás után számos tevékenység lehet
  - Firssítések telepítése
  - Adatok átmozgatása
  - ...
- Két stratégia
  - Push
    - Elindulás után
    - SSH, ...
  - Pull
    - A lemezképet létrehozó szkriptet módosítjuk megfelelően
- Teszteljünk le minden új példányt

```
"ServerHost" : {
  "Type" : "AWS::EC2::Instance",
  "Properties" : {
    "ImageId" : {
      "Ref" : "ServerAmi"
    },
    "InstanceType" : "t2.small",

    "UserData" : {
      "Fn::Base64" : {
        "Fn::Join" : [ "", [
          "#!/bin/bash\n",
          "/usr/local/sbin/server-setup.sh ",
          { "Ref": "ServerRole" }
        ]
      ]
    }
  }
}
```

Sablon menedzsment

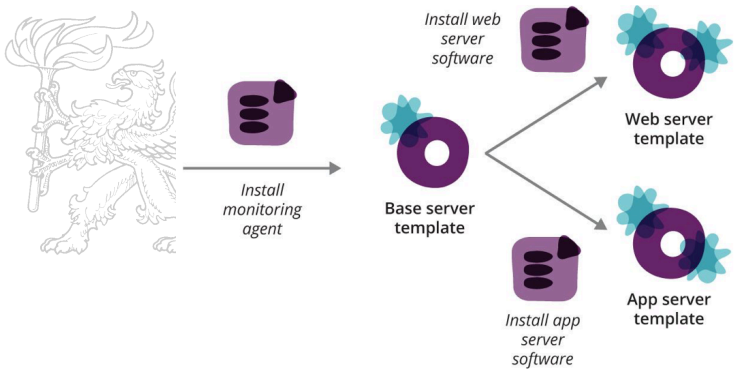
- Újramelegítjük a sablont
  - Új sablon a régire építve
  - Felesleges lépések lehetnek benne (szoftver telepítése majd eltávolítása)

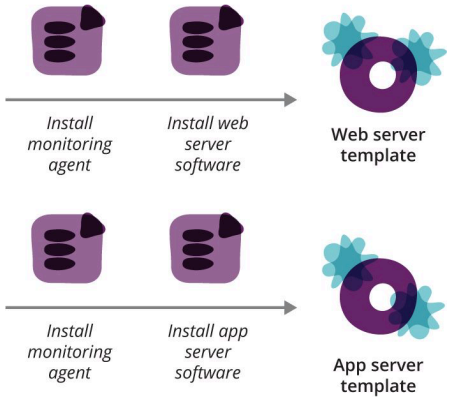


- Új sablonok készítése
  - Verziózva `{TEMPLATE_ROLE}-$\{YYYYMMDDHHSS}`.
- Tudnunk kell, hogy a futó szerverek mely sablonnal készültek

Szerepkörhöz kötött minták

- A szerverek létrehozása gyorsabb átláthatóbb
- Rétegelt sablon
  - Az adott szerepkörök is rétegelve egymásra épülve lehetnek
- Szkriptek megosztása





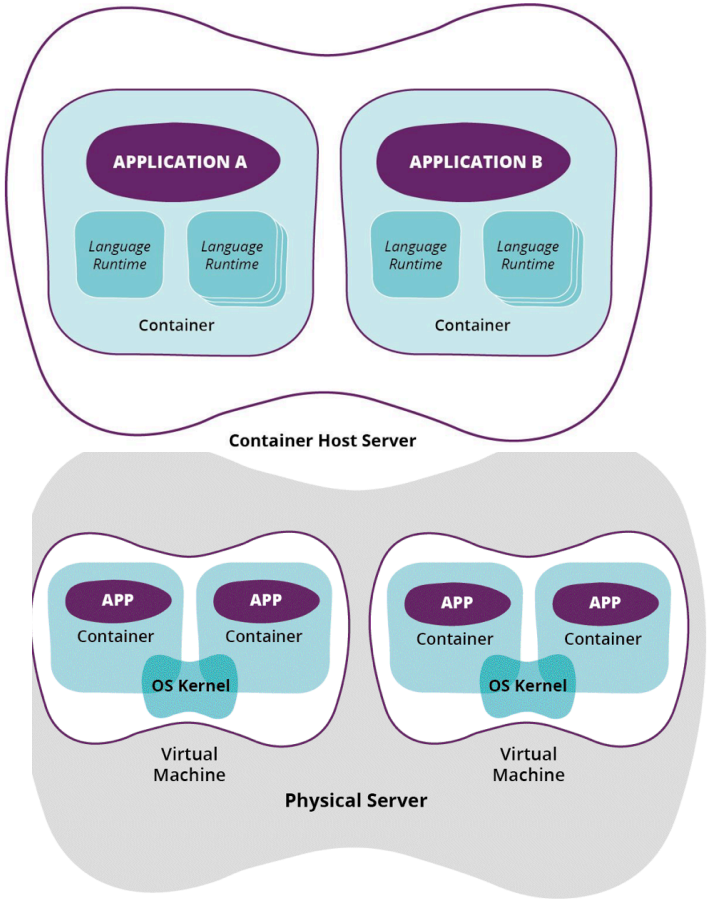
UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Szerver változtatás

- Az automatizált folyamat a legegyszerűbb és leghatékonyabb
- A változások minden meglévő szerverre kimennek
- Az egyforma szerverek egyformák maradnak, nem szabad inkonzisztenciának előfordulnia
- A változások beavatkozás nélkül végrehajtódnak
- A változások megvalósításához szükséges munka nem függ a szerverek mennyiségétől
- A hibák gyorsan láthatóak

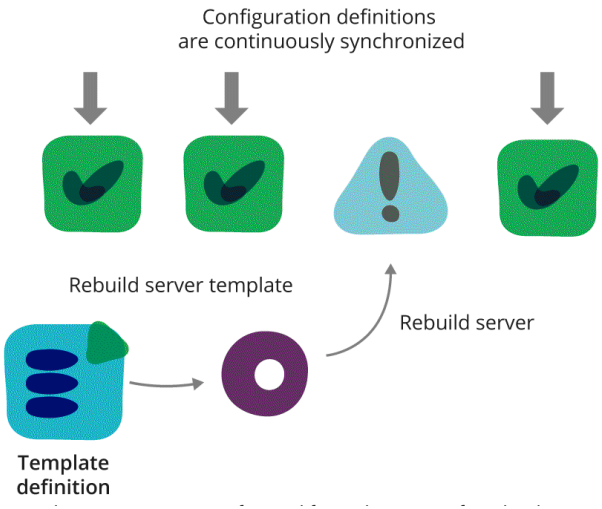
Szerver változás modellek

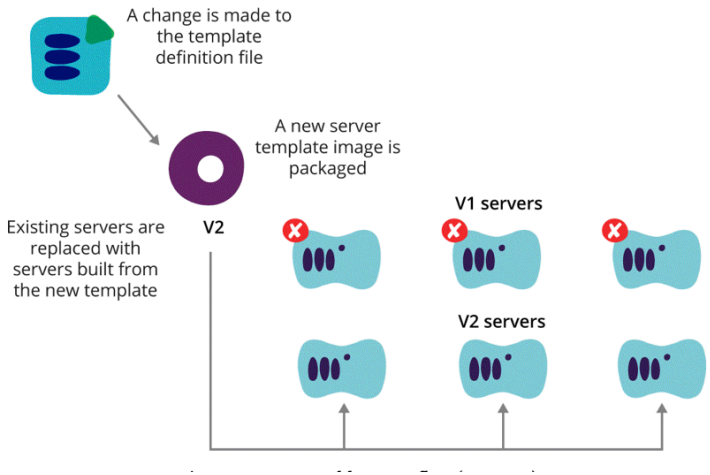
- Ad-Hoc változás menedzsment
  - Akkor változtatjuk amikor kell
- Konfiguráció szinkronizáció
  - Adott eszközzel érvényesíti a változásokat (pl.: Chef)
- Megváltoztathatatlan infrastruktúra
  - Nem módosít, hanem teljesen kicserél
- Konténerizált szolgáltatások
  - Unikernel
  - Könnyűsúlyú konténerekbe csomagolt szolgáltatások
  - A host így stabilabb, kevesebb változtatása



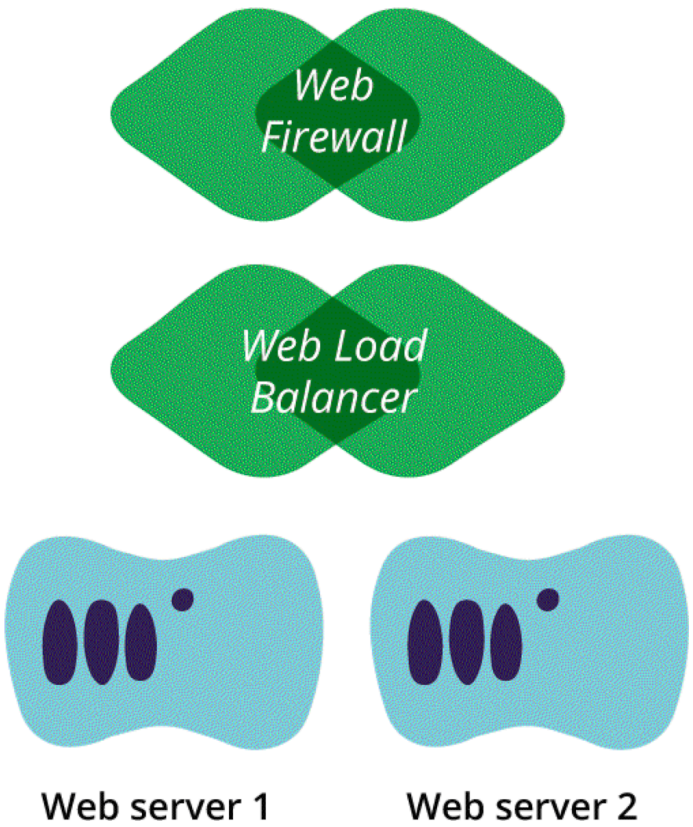
Szerver változtatások

- Minimalizált szerver sablon
- Nem menedzselt szerver változás (nem garantált, hogy minden azonnal szinkron lesz)
- Cseréljük le a szervert, ha változik a sablon
- Főnix szerver: rutinszerű szerver csere (garantált konzisztencia, adott élettartam után garantált csere)
- Tranzakcionális változás menedzsmenet





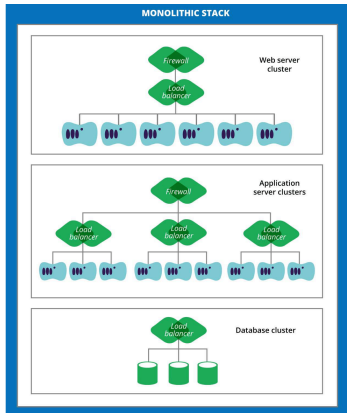
Infrastruktúra létrehozása

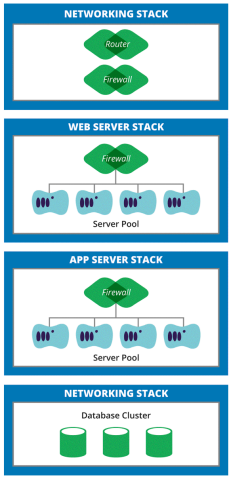


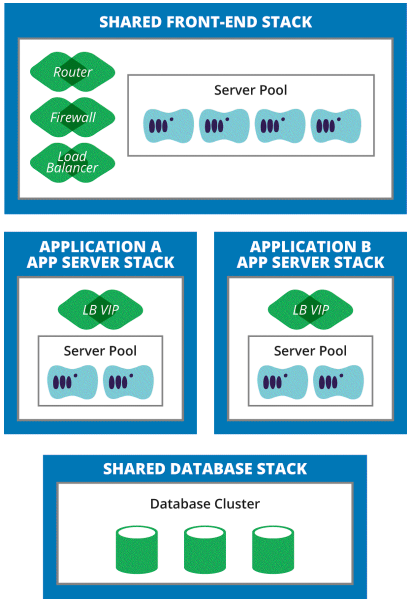

```
resource "aws_instance" "web_server_1" {
  ami = "ami-47a23a30"
  instance_type = "t2.micro"
}

resource "aws_instance" "web_server_2" {
  ami = "ami-47a23a30"
  instance_type = "t2.micro"
}

resource "aws_elb" "web_load_balancer" {
  name = "weblb"

  listener {
    instance_port = 80
    instance_protocol = "http"
    lb_port = 80
    lb_protocol = "http"
  }

  instances = [
    "${aws_instance.web_server_1.id}",
    "${aws_instance.web_server_2.id}"
  ]

  security_groups = ["${aws_security_group.web_firewall.id}"]
}

resource "aws_security_group" "web_firewall" {
  name = "web_firewall"

  ingress {
      from_port = 80
      to_port = 80
      protocol = "tcp"
      cidr_blocks = ["0.0.0.0/0"]
  }
}
```

Infrastruktúra menedzsment

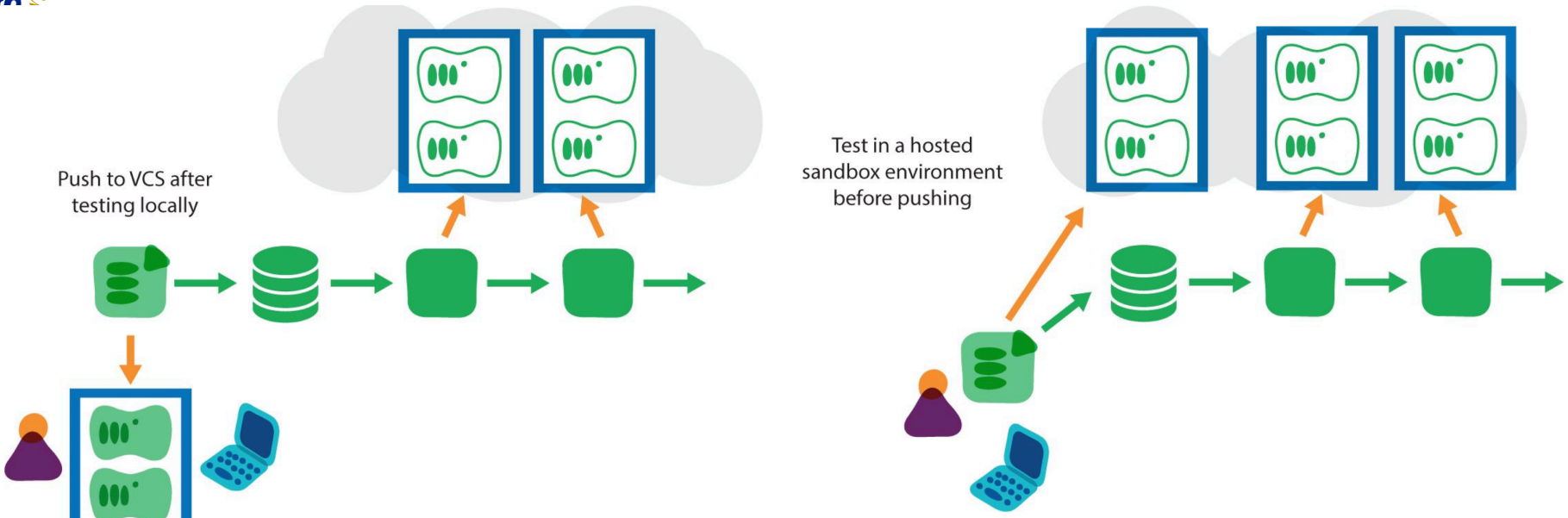
- Teljes környezetek menedzselése
  - Dev, Prod, Staging, …
- Hópehely infrastruktúrák
- Monolitikus infrastruktúra
  - minden egy helyen van specifikálva
- Osztott környezetek
- Osztott elemek





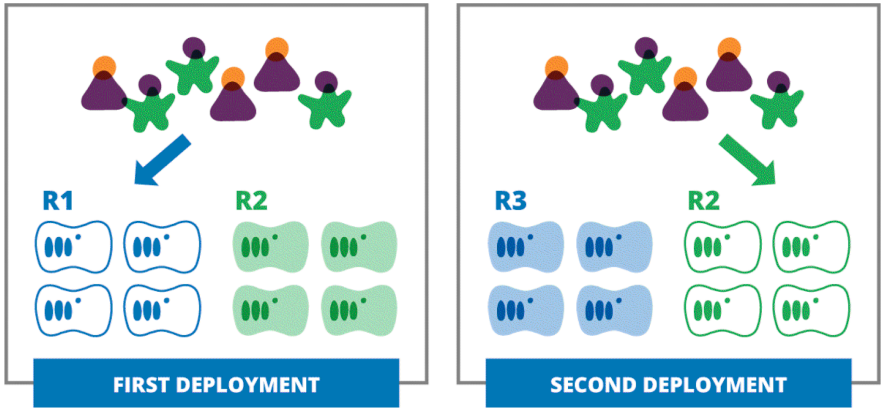


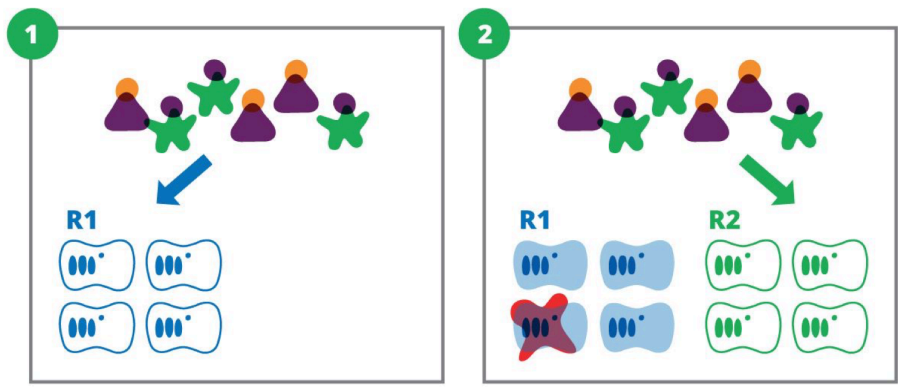
Végcél

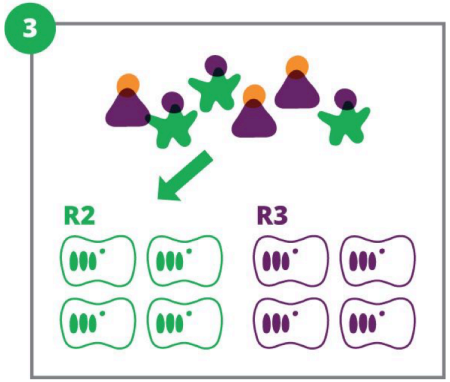

Push to VCS after testing locally
Test in a hosted sandbox environment before pushing

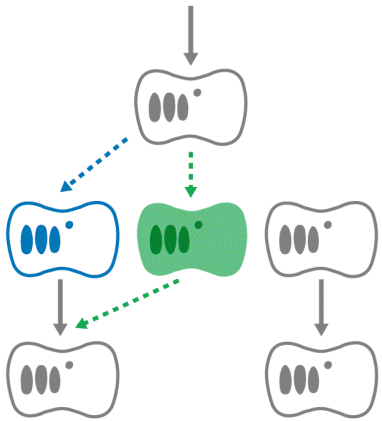
Folyamatosság

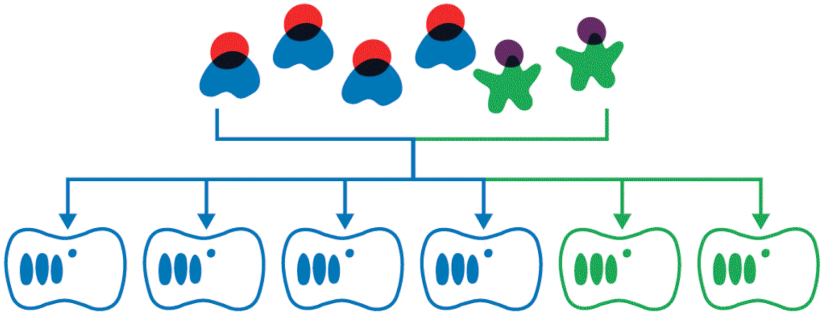
- Zéró hibás időszak
- Kék-Zöld csere
- Főnix csere
- Kanári kibocsájtás minta
- Sötét indítás





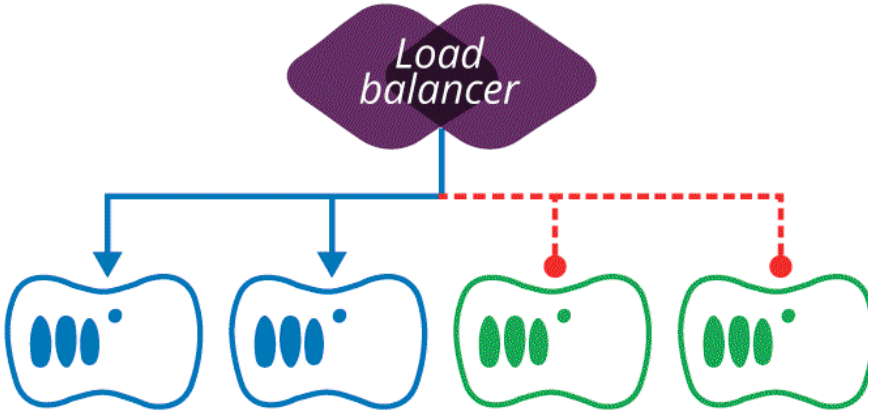


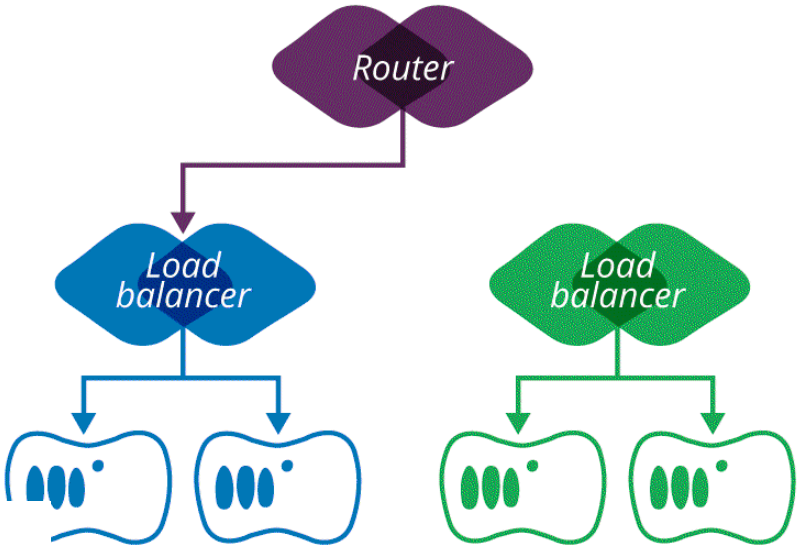


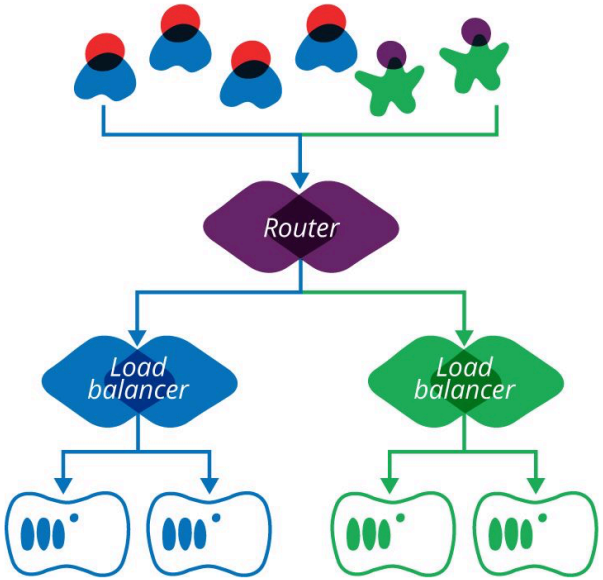


UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Háttér


Load balancer


Router
Load balancer
Load balancer

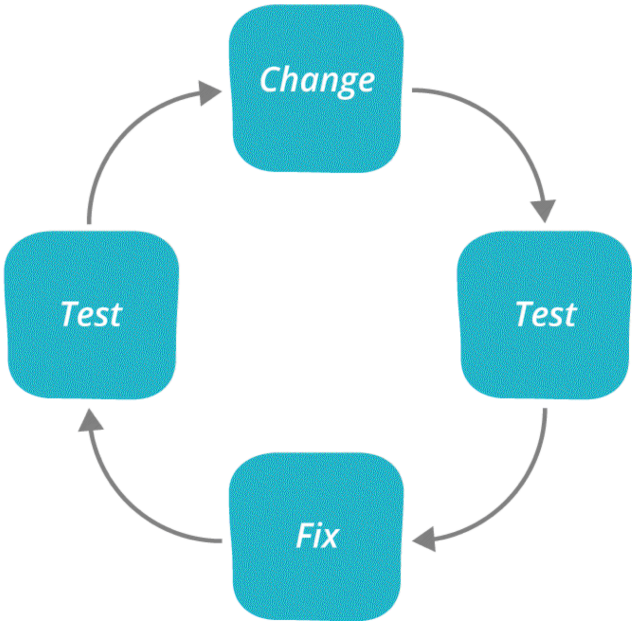

Router
Load balancer
Load balancer

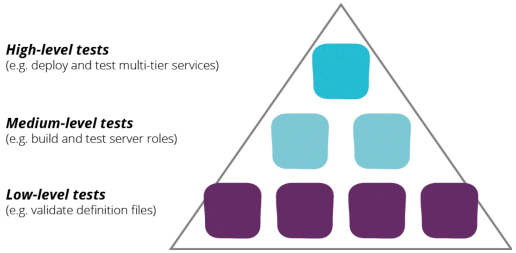
Adat folyamatosság

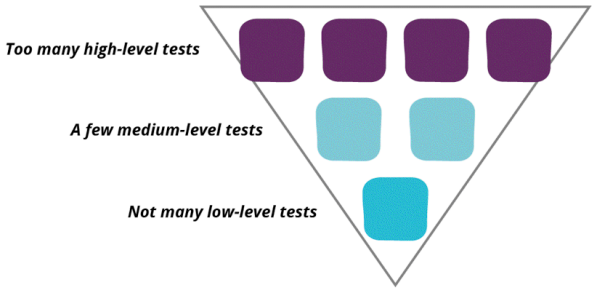
- Adat replikáció (klaszter)
- Adat regenerálás (konfig)
- Perzisztencia delegáció (RDS)
- Perzisztens mentés
  - Katasztrófa visszaállítás

Folyamatos kibocsájtás

- A folyamatos integráció az egy kódbázison történő párhuzamos fejlesztésre ad módszertant
- A folyamatos kibocsájtás kiterjeszti ezt az egész rendszerre

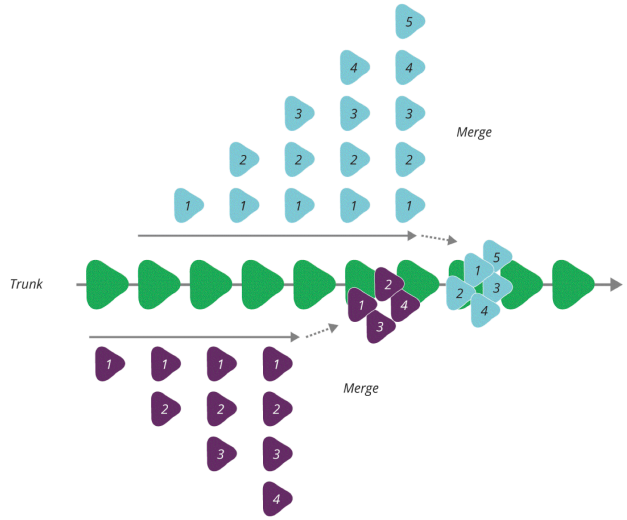


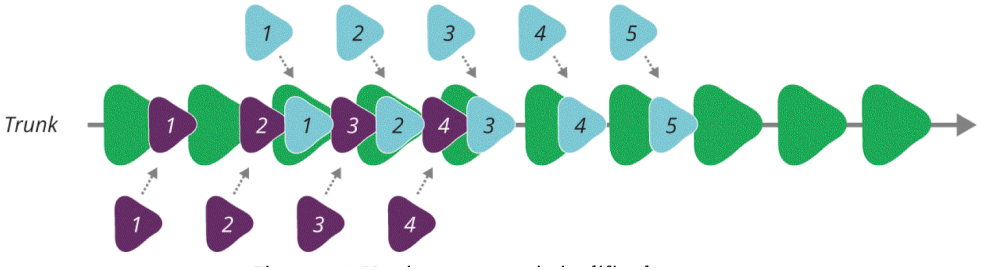




Folyamatos integráció

- Verziókezelés stratégiák
  - Funkció alapú
  - Kibocsájtás alapú
  - Folyamatos integráció

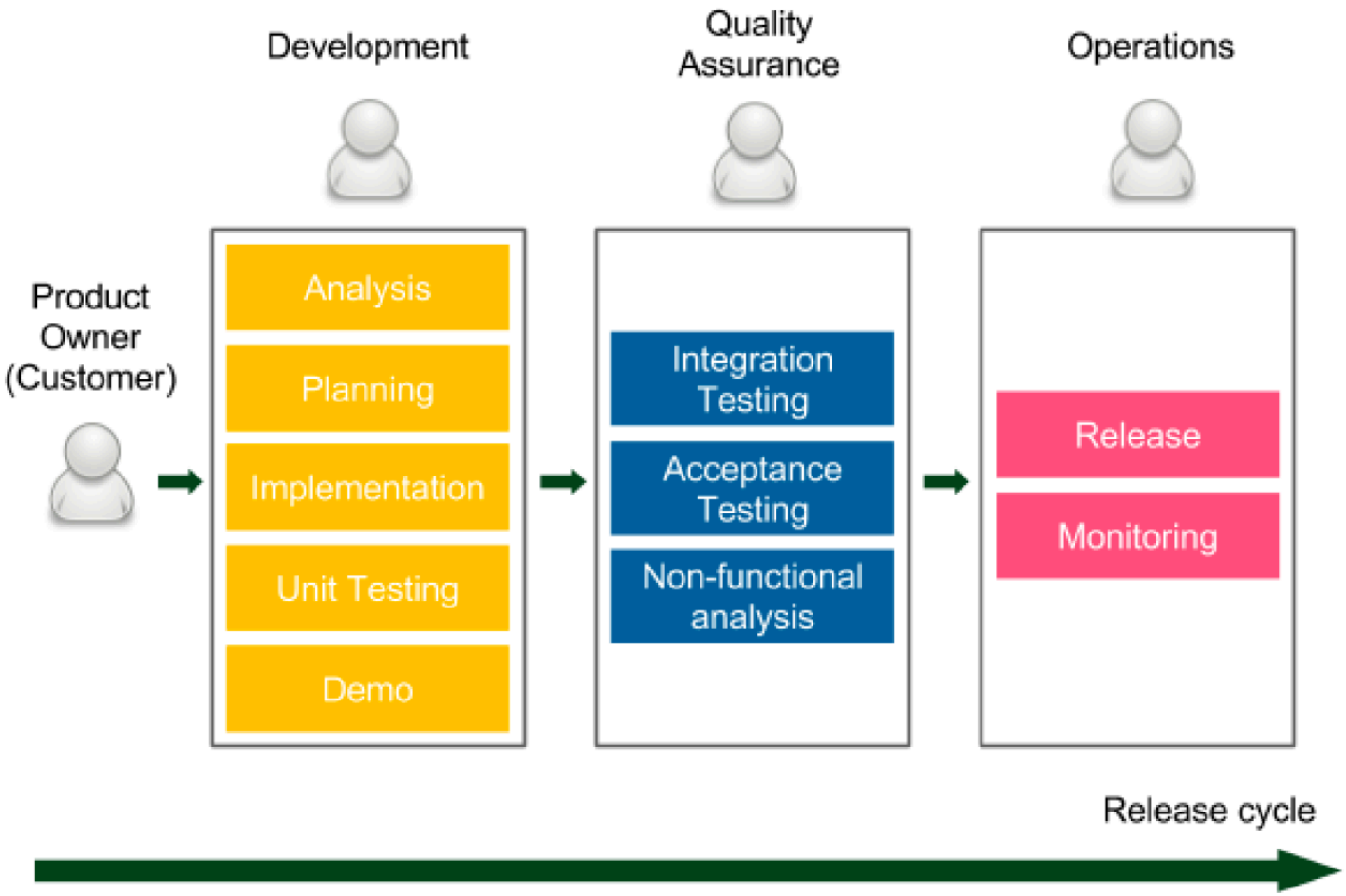




CI szintek

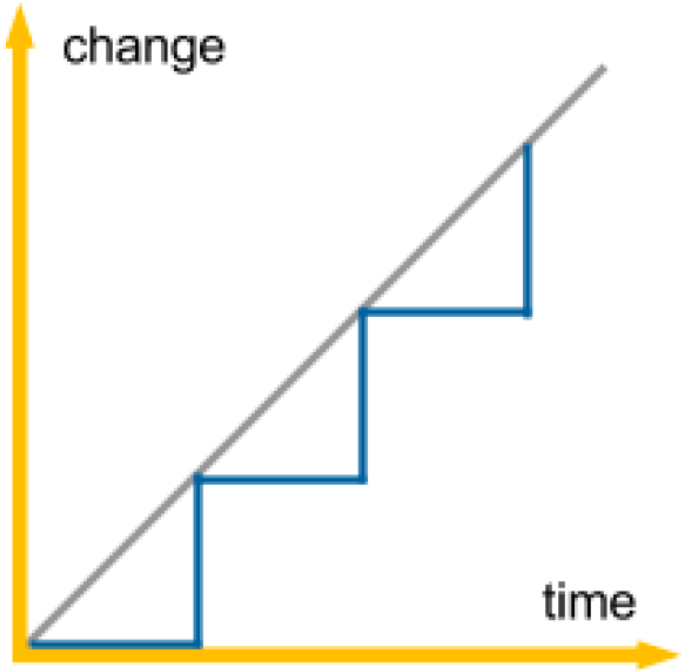
1. Fázis: Nincs build szerver
2. Fázis: Éjszakai build-ek
3. Fázis: Éjszakai tesztek és automatikus tesztek
4. Fázis: Metrikák használata
5. Fázis: Komolyabb tesztelés: TDD
6. Fázis: Automatikus elfogadás és automatikusabb kitelepítés
7. Fázis: CD – folyamatos telepítés

Klasszikus kibocsájtás

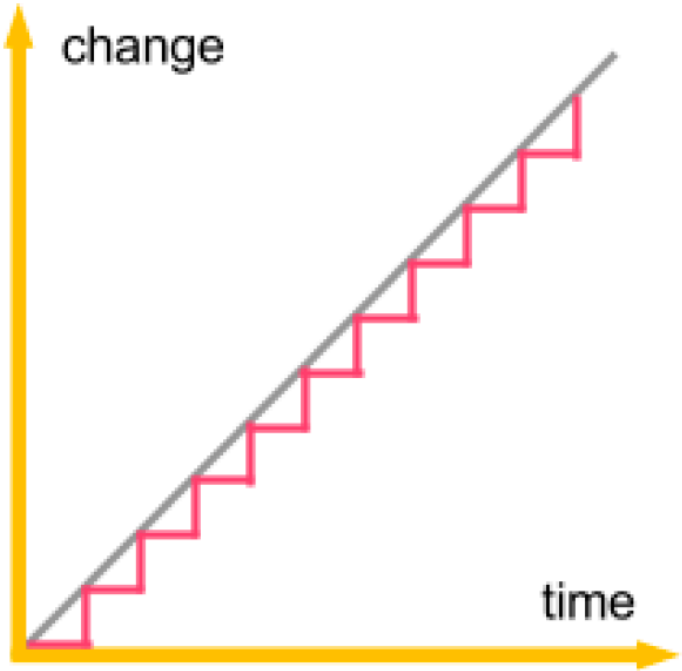


Klasszikuis vs CI/CD

YAHOO!



flickr



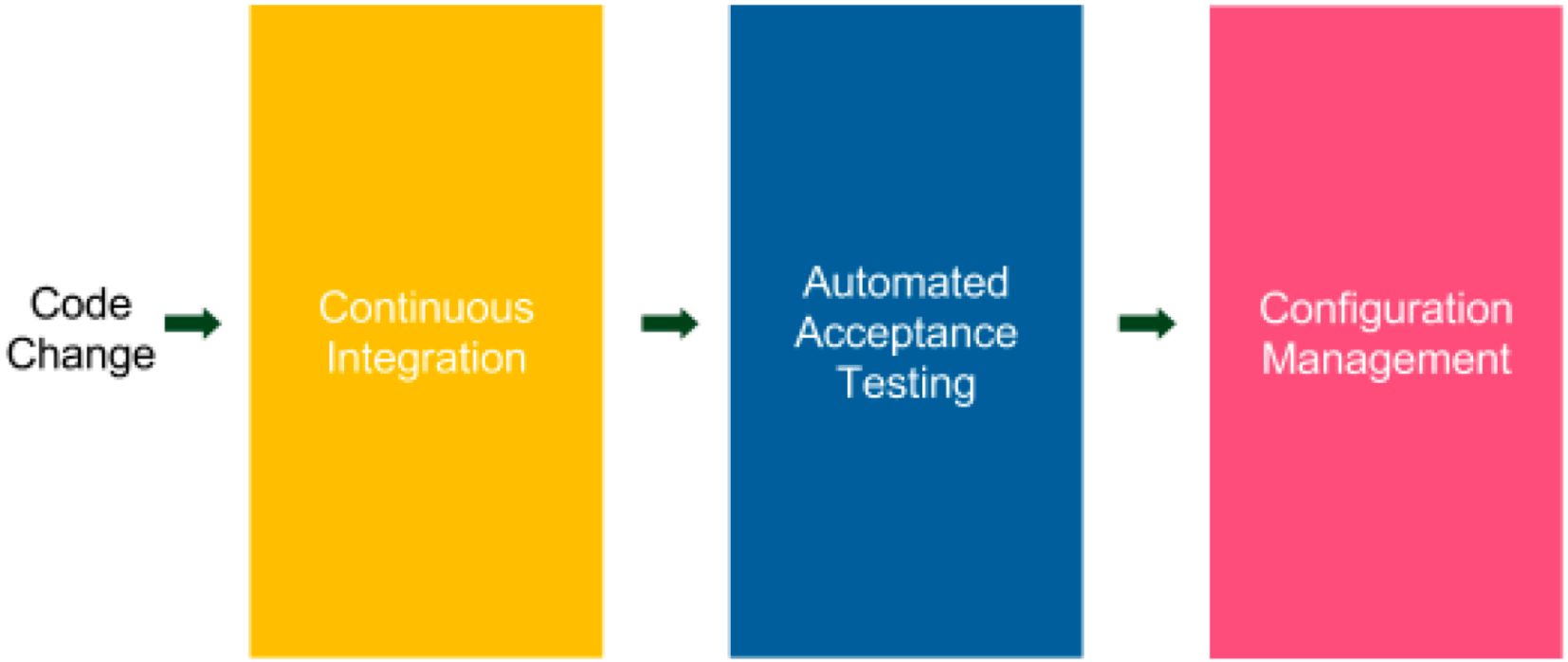
CD

- A változás kicsi
- A folyamat megismételhető

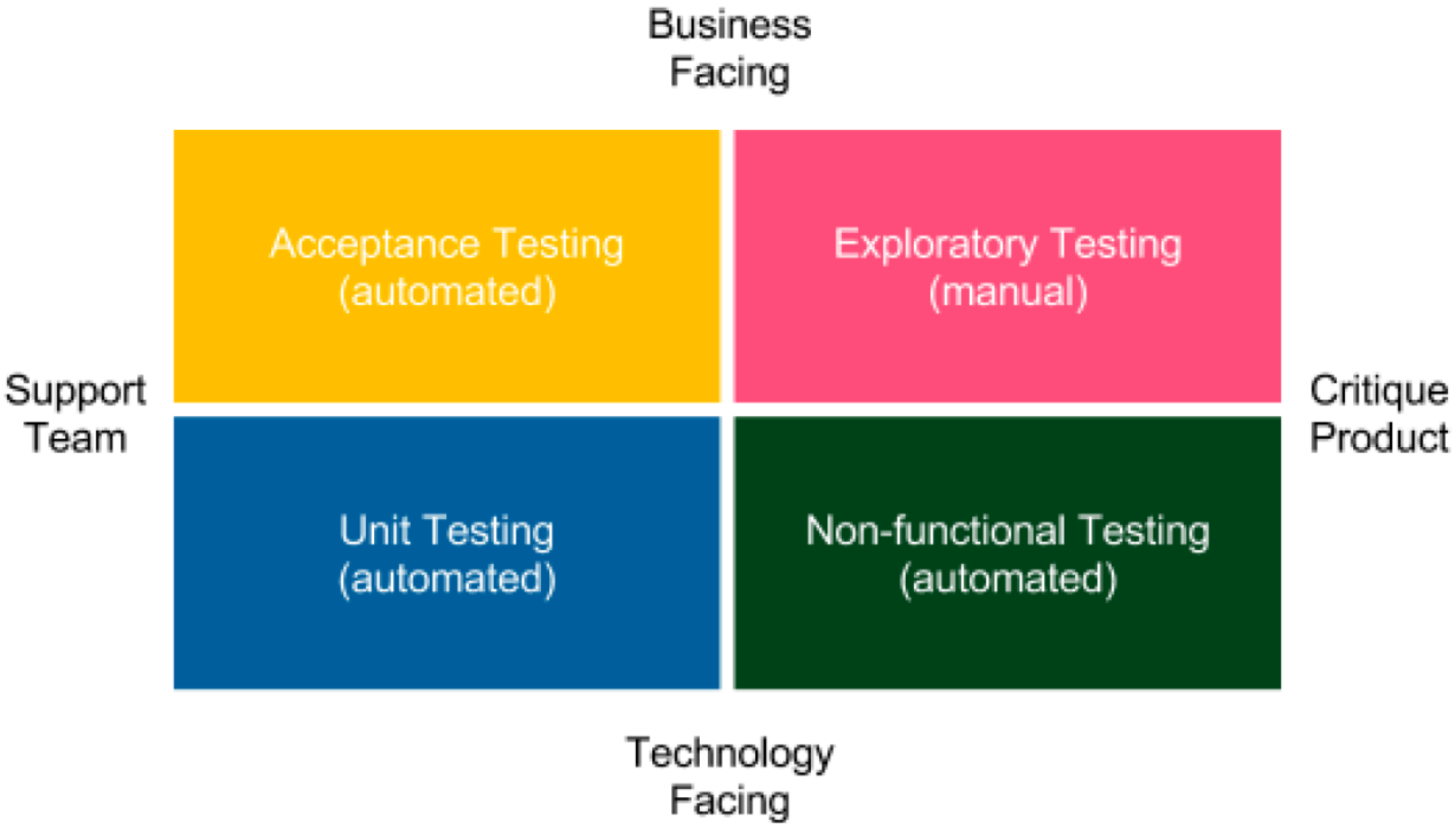
Érdekesség

- Amazon: 2011 a telepítések között 11.6 másodperc tellik el
- Facebook: 2013 a kódot naponta kétszer firssítik
- HubSpot 2013 300 telepítés naponta
- Atlassia: 2016 a vevők 65%-a CD-t alkalamas

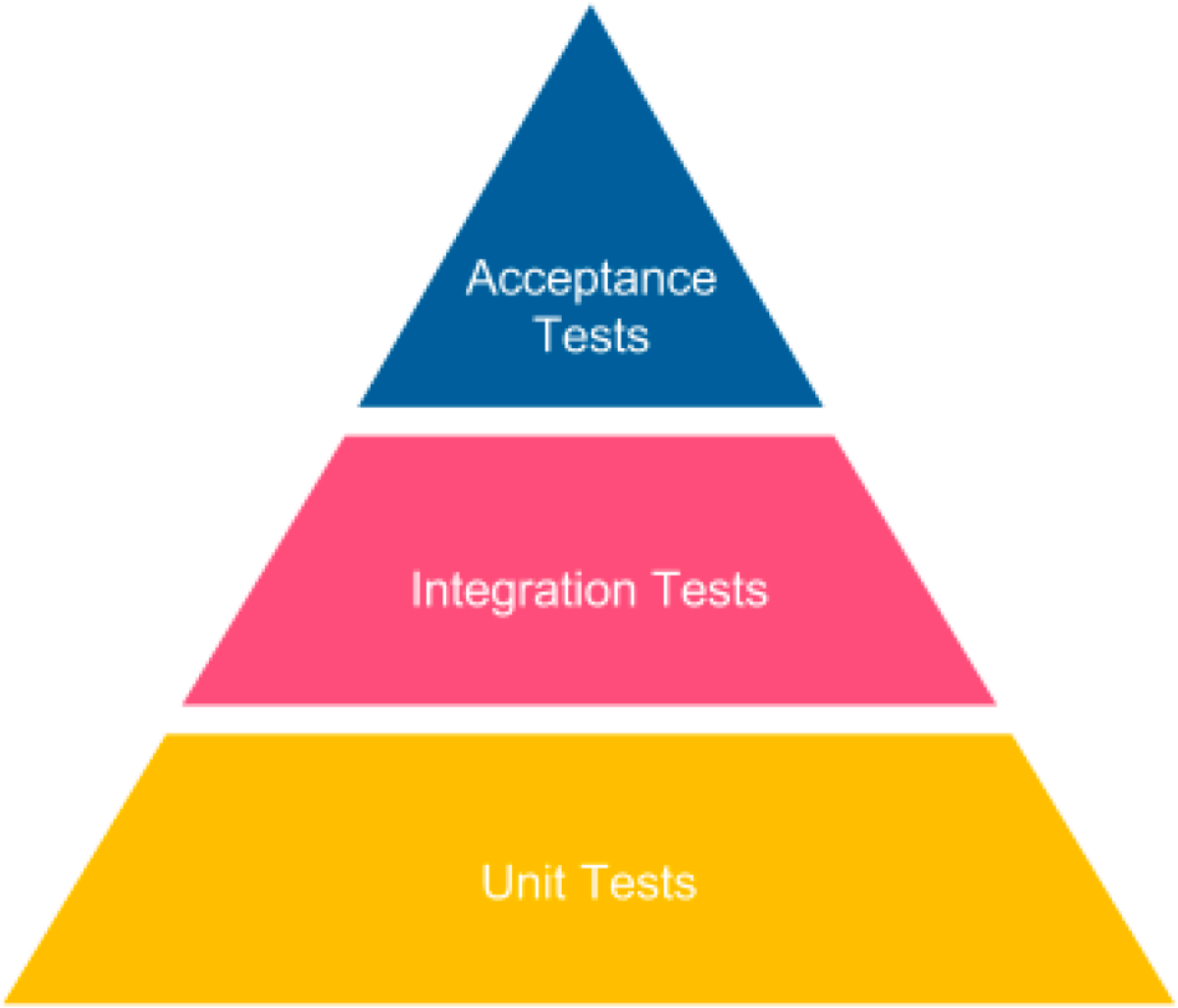
Automatikus telepítés



Agilis tesztelés mátrix



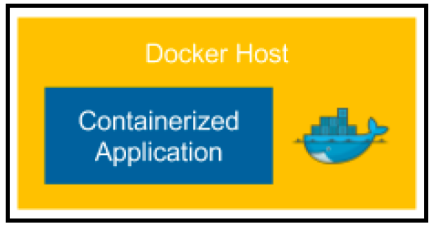
Tesztelés piramis



Technkiai követelmények

- Automatikus build, teszt, csomagolás, telepítés
- Gyors folyamatok (5-15 perc)
- Gyors hiba detektálás és korrekció (rollback)
- Zéró leállásidejű telepítés
- Trönk alapú fejlesztés

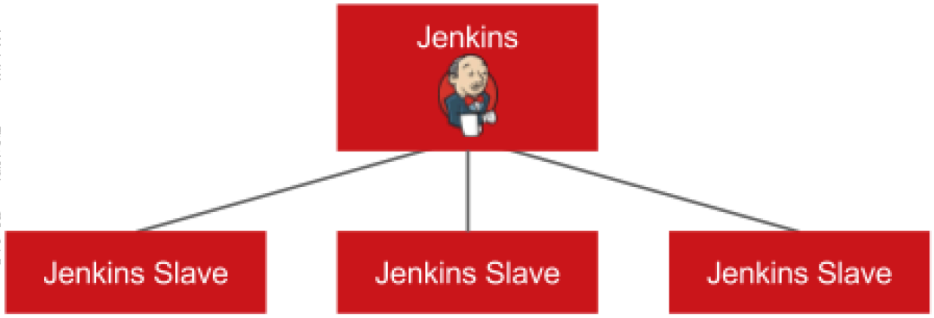
Eszközök



CHEF

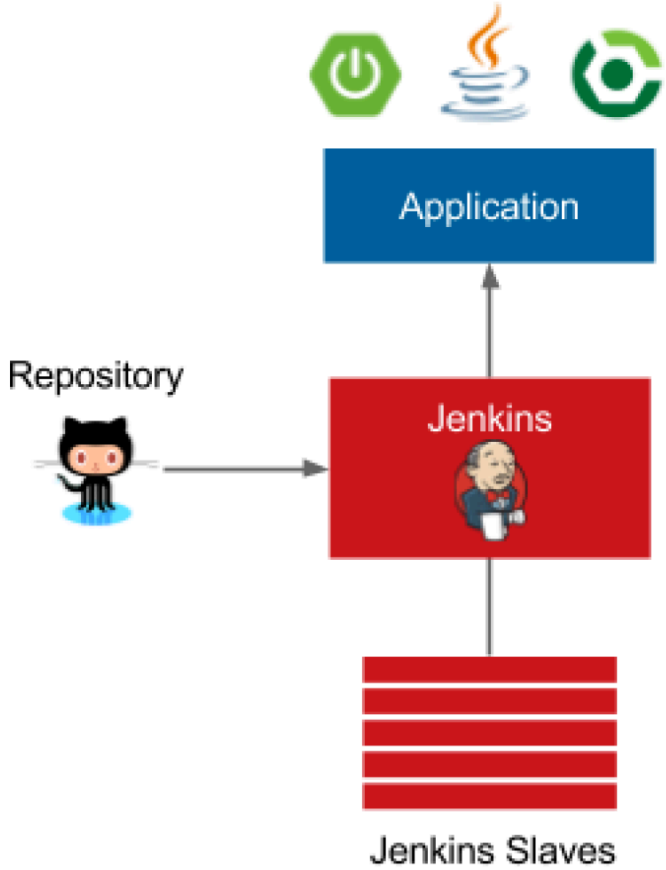
openstack

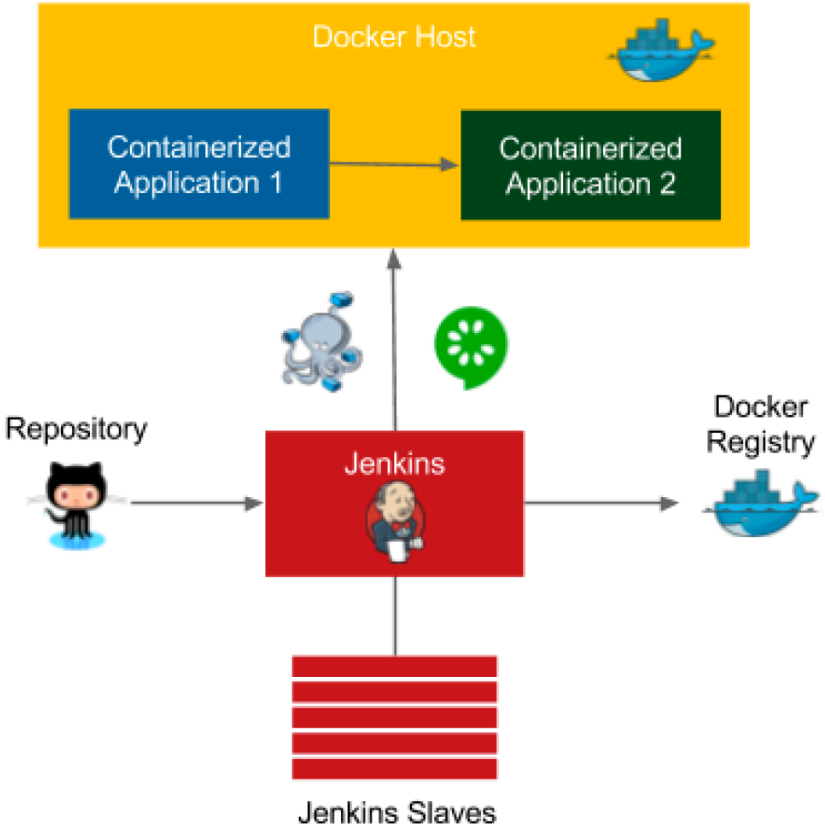
git



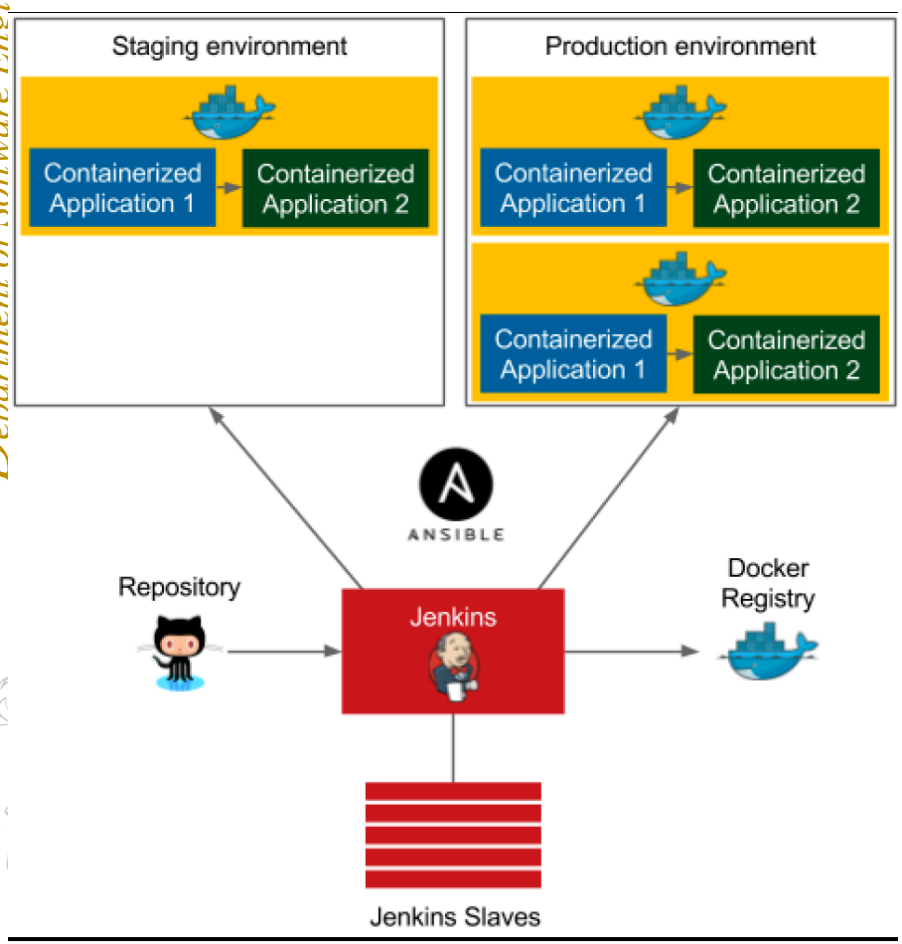
VAGRANT

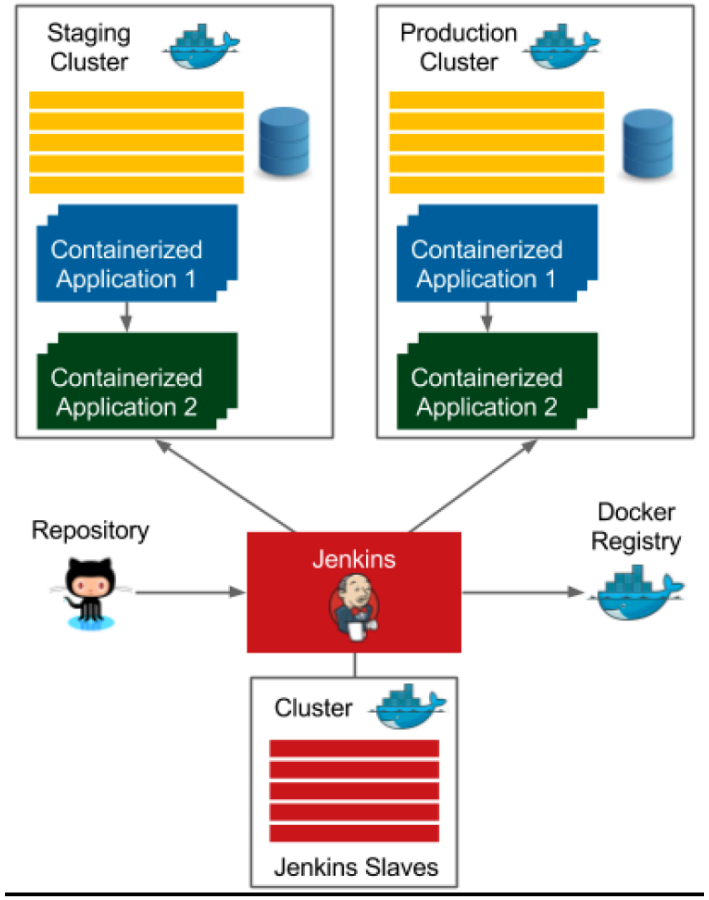
CI


Application
Repository
Jenkins
Jenkins Slaves


Docker Host
Containerized Application 1
Containerized Application 2
Repository
Jenkins
Docker Registry
Jenkins Slaves

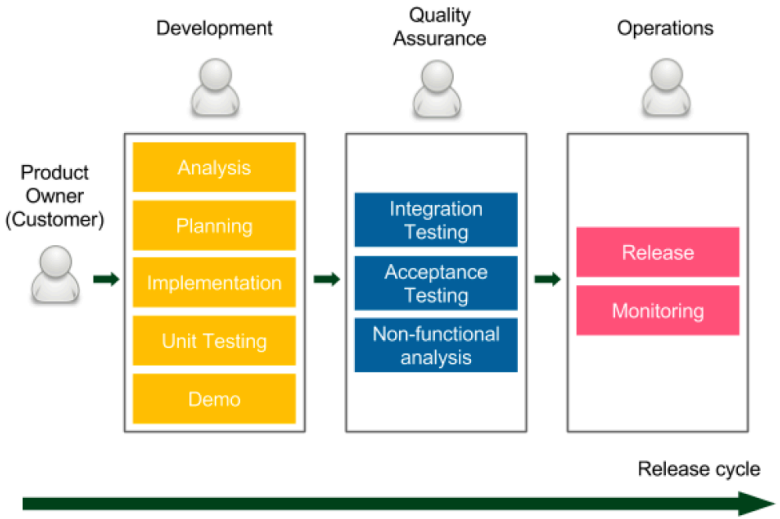
CD

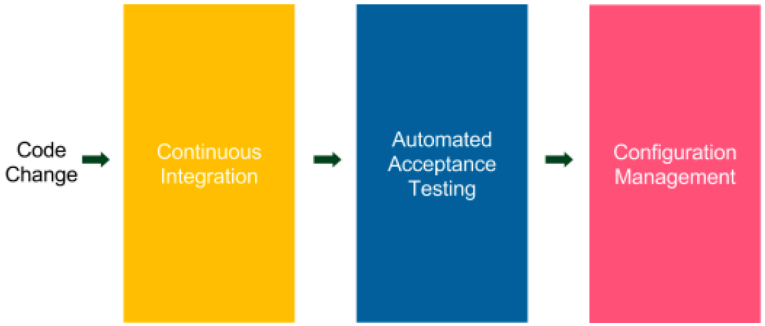

Staging environment
Containerized Application 1
Containerized Application 2
Production environment
Containerized Application 1
Containerized Application 2
Containerized Application 1
Containerized Application 2
ANSIBLE
Repository
Jenkins
Docker Registry
Jenkins Slaves


Staging Cluster
Containerized Application 1
Containerized Application 2
Production Cluster
Containerized Application 1
Containerized Application 2
Repository
Jenkins
Docker Registry
Cluster
Jenkins Slaves

CI/CD alapok

- Folyamatos kibocsájtás
  - A változások (kód, konfig,...) folyamatos, fenttartható kibocsájtása produkciós környezetbe.



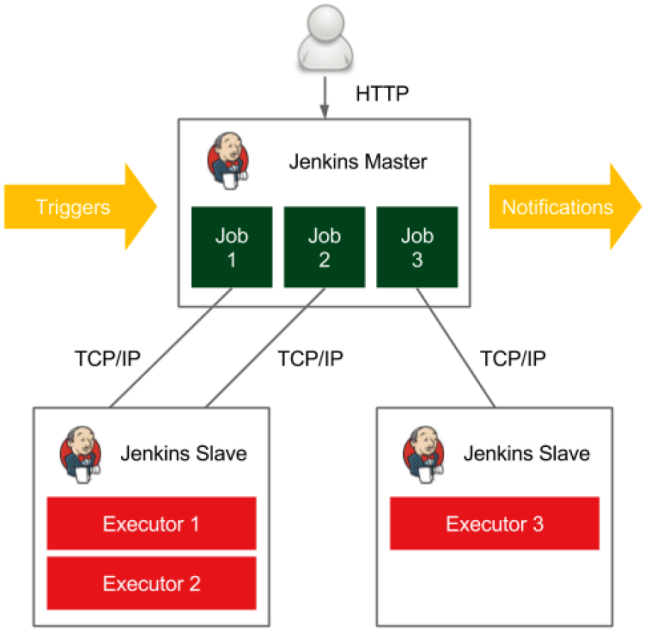


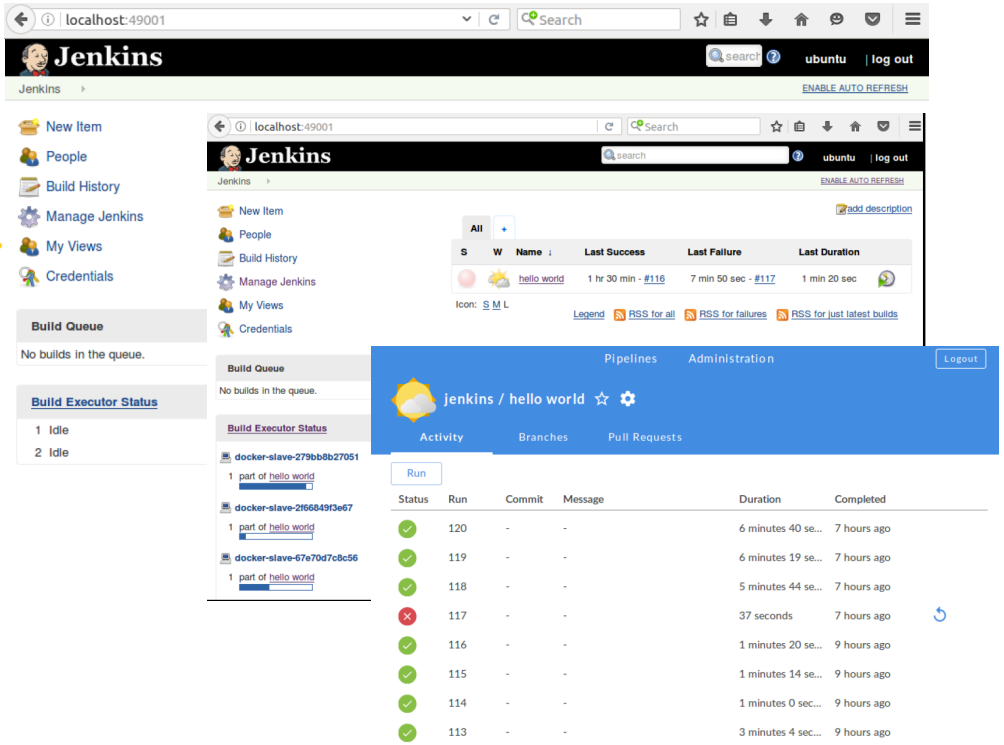
Klasszikus kibocsájtás

- Lassú kibocsájtás, hosszú folyamat (hetek)
- Lassú visszacsatolás
- Automatizálás hiánya
- Kockázatos hotfix-ek
- Stresszes
- Gyenge kommunikáció
- Megosztott felelőség

Jenkins

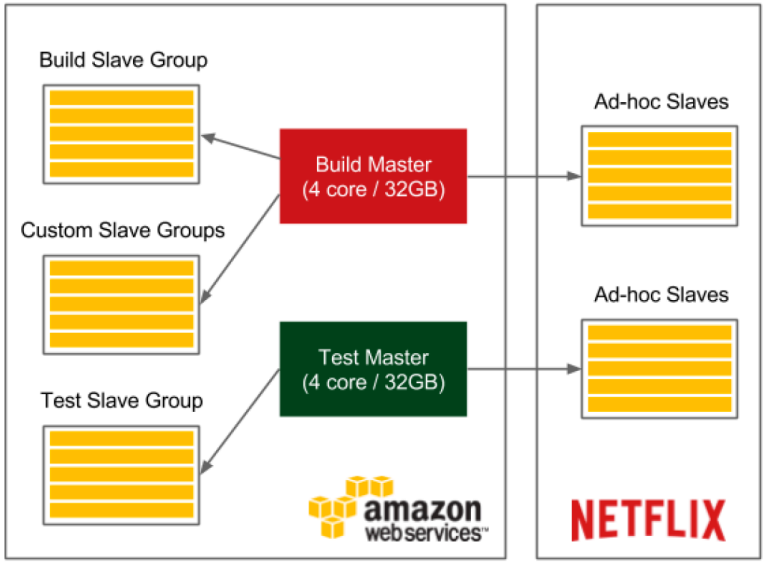
- Java alapú nyílt forrású szerver
- CI/CD megvalósításában segít

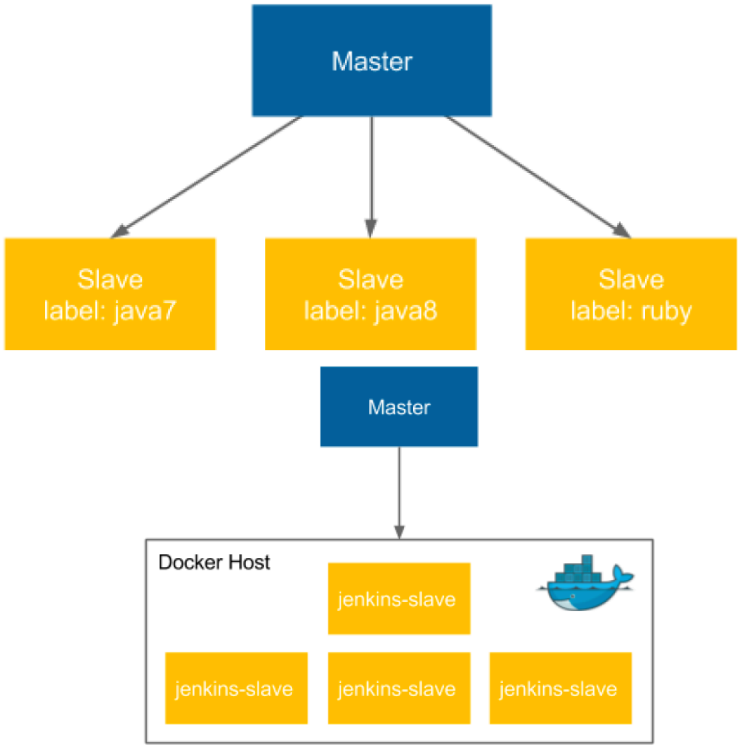




Működése

- Build triggerek beérkezése (GitHub commit)
- Értesítés kiküldése
- HTTP kérések kezelése
- Build környezet menedzselése



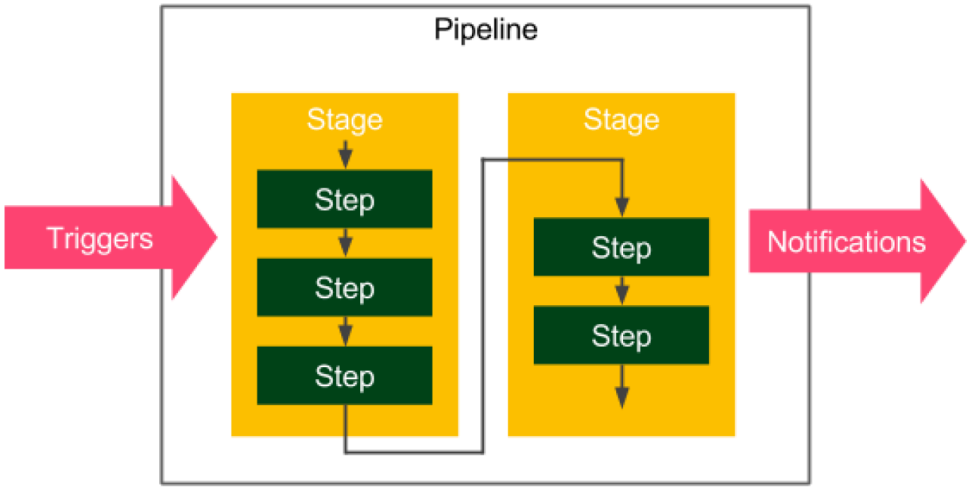


UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

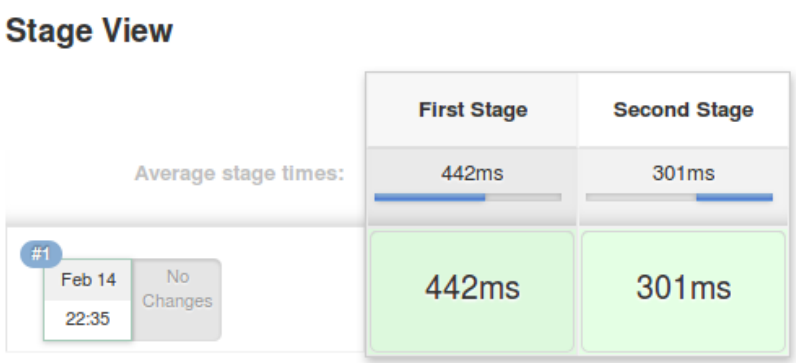
UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

CI sor

- Munka csoportosítása
  - Sor/Cső
  - Fokozat
  - Lépés




```
pipeline {
    agent any
    stages {
        stage('First Stage') {
            steps {
                echo 'Step 1. Hello World'
            }
        }
        stage('Second Stage') {
            steps {
                echo 'Step 2. Second time Hello'
                echo 'Step 3. Third time Hello'
            }
        }
    }
```



Példa

```
pipeline {
    agent any
    triggers { cron('* * * * *') }
    options { timeout(time: 5) }
    parameters {
        booleanParam(name: 'DEBUG_BUILD', defaultValue: true,
        description: 'Is it the debug build?')
    }
    stages {
        stage('Example') {
            environment { NAME = 'Rafal' }
            when { expression { return params.DEBUG_BUILD } }
            steps {
                echo "Hello from $NAME"
                script {
                    def browsers = ['chrome', 'firefox']
                    for (int i = 0; i < browsers.size(); ++i) {
                        echo "Testing the ${browsers[i]} browser."
                    }
                }
            }
        }
    }
    post { always { echo 'I will always say Hello again!' } }
}
```

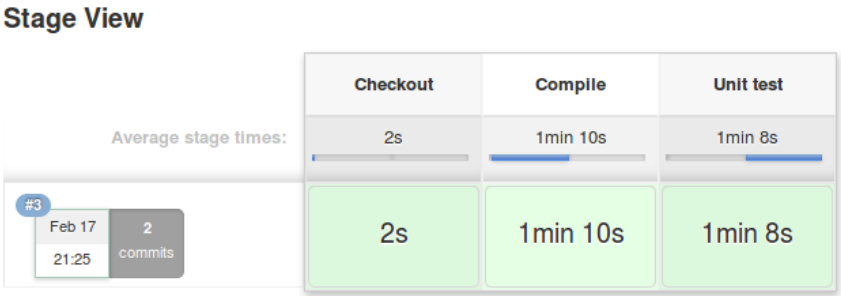
Szakaszok/Direktívák

- Stages
  - Lépések csoportosítása
- Steps
  - sh
  - custom
  - script
- Post
- Agent (label)
- Triggers (cron, pollSCM, ..)
- Options (timeout, retry)
- Enviroment
- Parameters
- Stage
- When

Commit pipeline

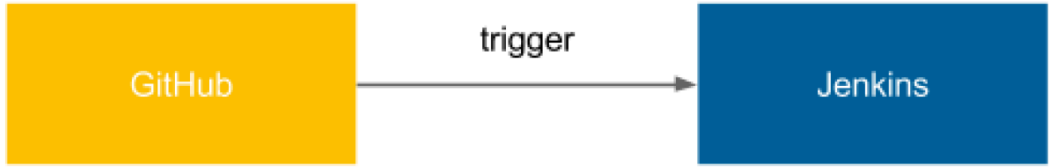
- checkout
- complie
- test

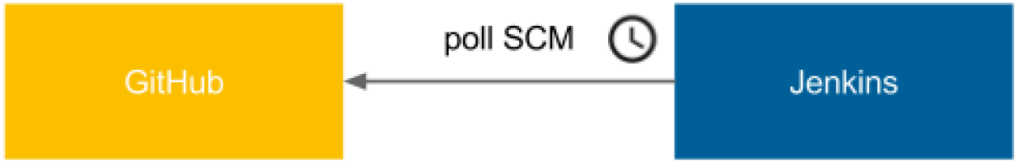
```
pipeline {
    agent any
    stages {
        stage("Checkout") {
            steps {
                git url: 'https://github.com/leszko/calculator.git'
            }
        }
    }
}
stage("Compile") {
    steps {
        sh "./gradlew compileJava"
    }
}
stage("Unit test") {
    steps {
        sh "./gradlew test"
    }
}
```

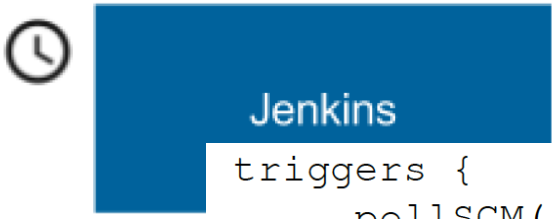


Értesítések, események

- Triggerek
  - scm
  - külső
  - időzített
- Értesítések
  - email
  - chat
  - team space





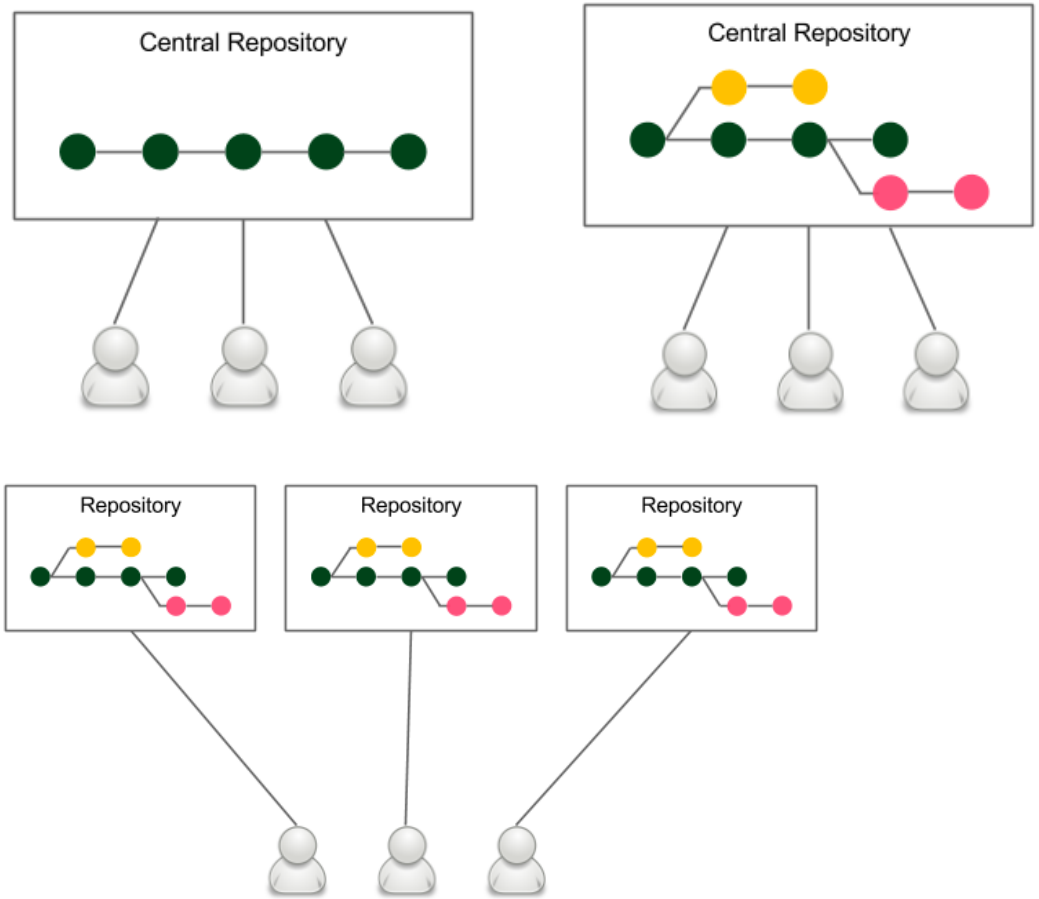



```
triggers {
    pollSCM('* * * * *')
}
```

```
post {
    always {
        mail to: 'team@company.com',
        subject: "Completed Pipeline: ${currentBuild.fullDisplayName}",
        body: "Your build completed, please check: ${env.BUILD_URL}"
    }
}
```

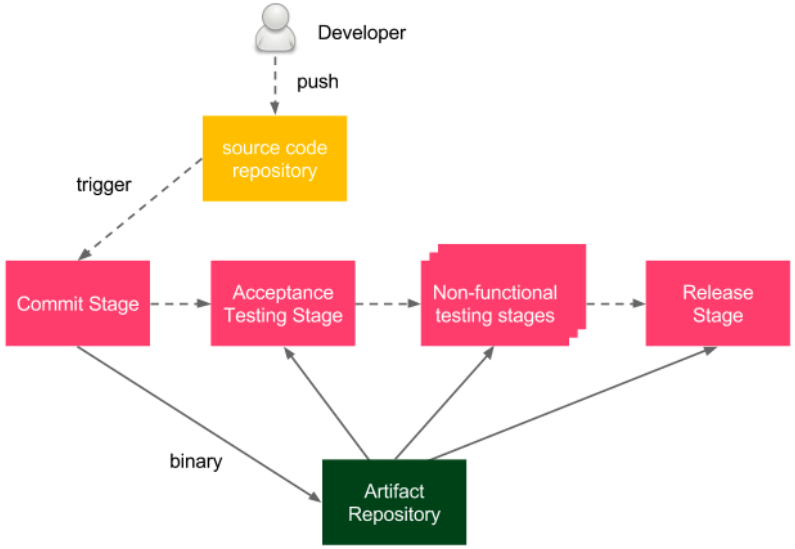
UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

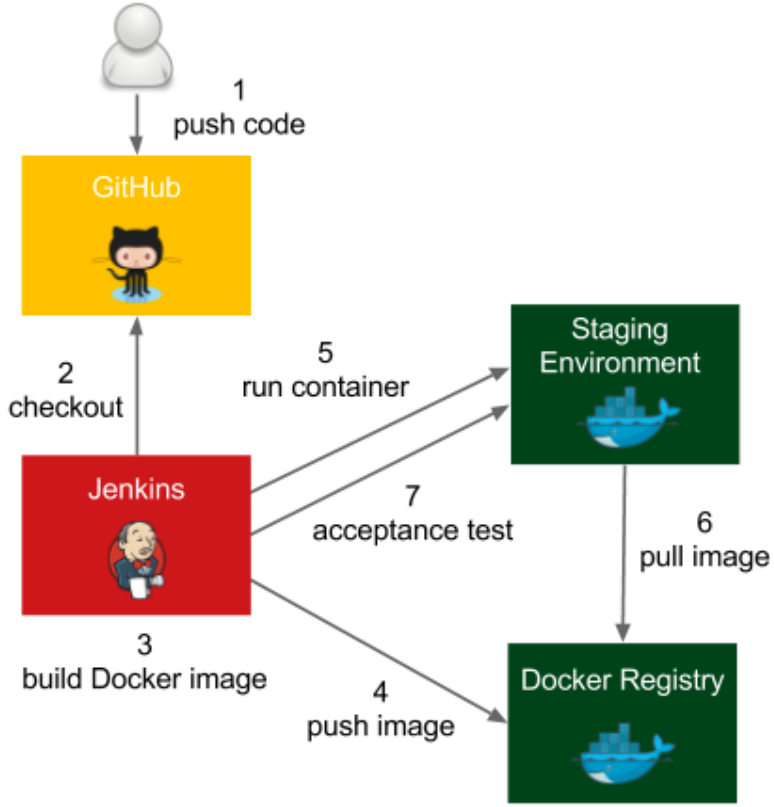
Fejlesztési stratégiák



Automatikus Elfogadás Tesztelés

- UAT
- Műalkotás tár (Articaft repo)

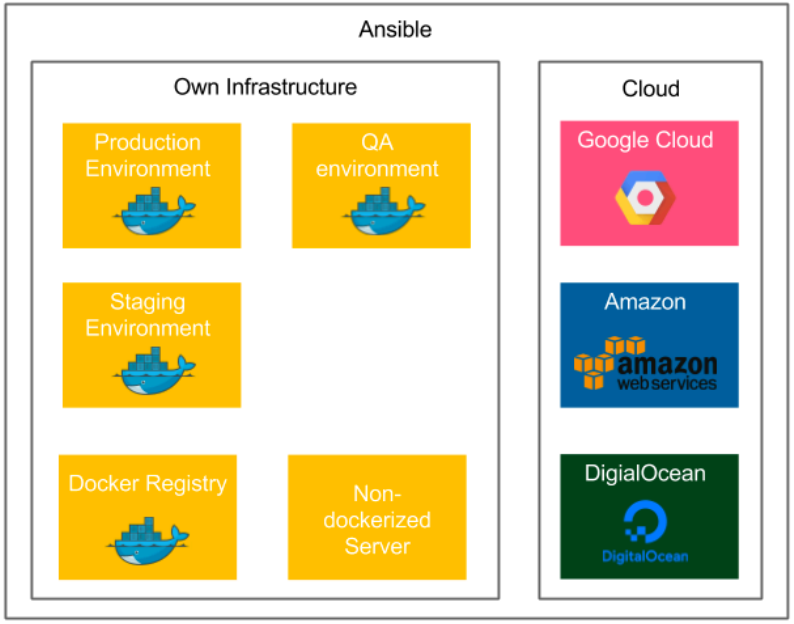




Konfiguráció menedzsment

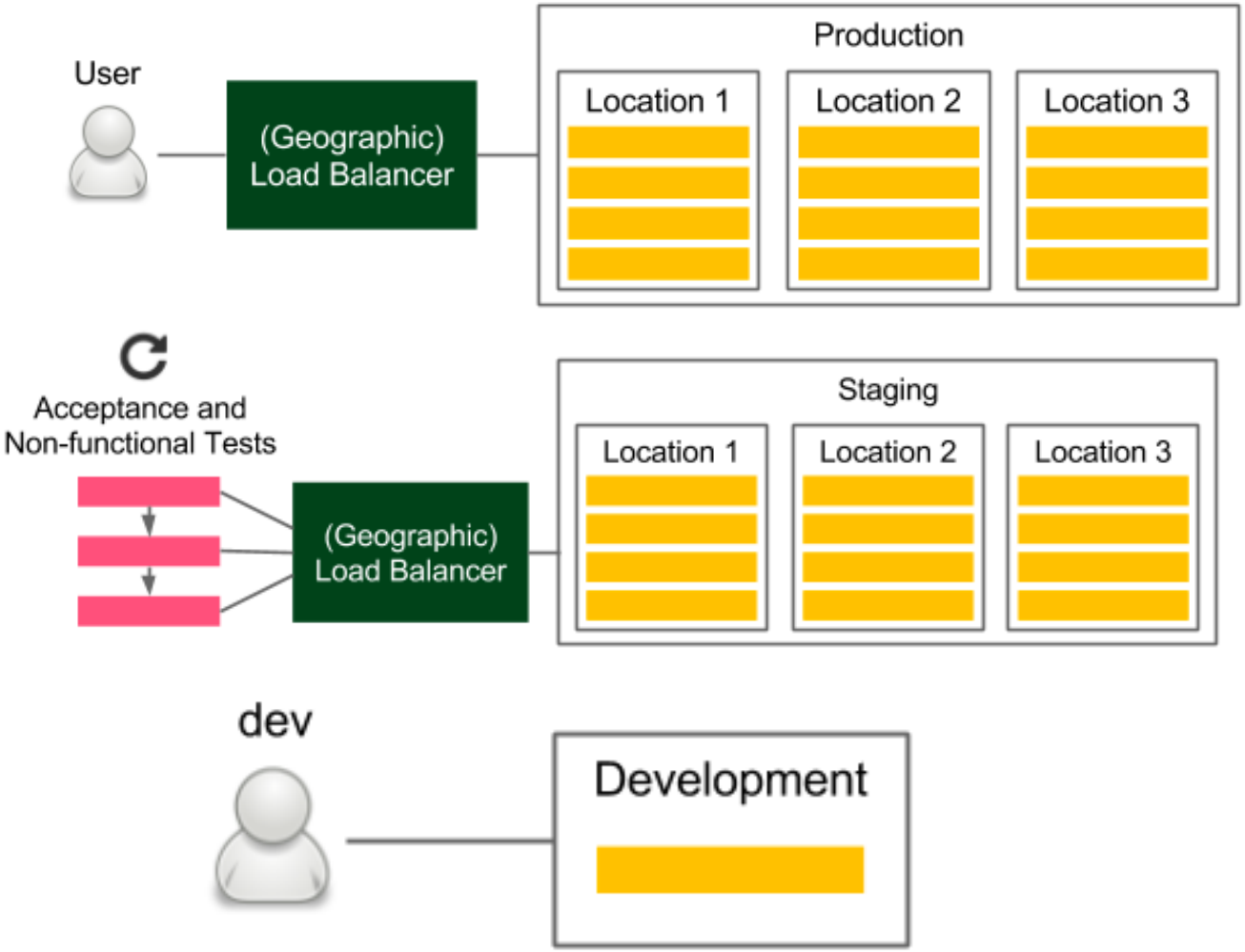
- Ansible
  - ügynökmentes (ssh)
  - YAML
- Plavbook

```
$ ansible all -m ping
web1 | SUCCESS => {
        "changed": false,
        "ping": "pong"
}
web2 | SUCCESS => {
        "changed": false,
        "ping": "pong"
}
```




```
---
- hosts: web1
  become: yes
  become_method: sudo
  tasks:
  - name: ensure apache is at the latest version
    apt: name=apache2 state=latest
  - name: ensure apache is running
    service: name=apache2 state=started enabled=yes
```

Conitnous Delivery



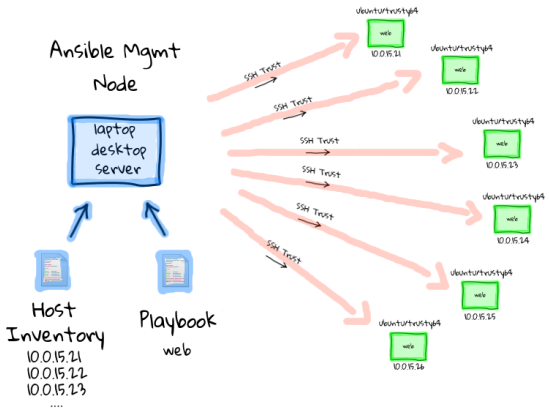
Ansible áttekintés

- Ad-hoc parancsok
- Forgatókönyvek
  - Környezeti változók
  - Változók
  - Elágazások
  - Delegáció, Helyi akciók, Szünetek
- Forgatókönyv szervezés
- Leltárak
- Szakácskönyvek
- Biztonság
- További lehetőségek

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Ansible Alapok

- ssh hozzáférs kell (:port)
  - Jelszómentes (u@gep.com)
- Nem kell a szerveren ügynök (memória, ... takarékos)
- Push modell
- Reguláris shell parancsok, de idempontens képességget megvalósító forgatókönyvekbe



UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Leltár fájl

- A szerverek listáját tartalmazza
  - Csoportosítható [csoport]
- /etc/ansible/hosts
- Külön környezetekre
- Változók

```
[www]
# You can define host-specific variables inline with the host.
www1.example.com ansible_ssh_user=johndoe
www2.example.com

[db]
db1.example.com
db2.example.com

# You can add a '[group:vars]' heading to create variables that will\
 apply
# to an entire inventory group.
[db:vars]
ansible_ssh_port=5222
database_performance_mode=true
```

```
servercheck/
  inventories/
    inventory-prod
    inventory-cert
    inventory-dev
  playbook.yml
```

```
$ ansible-playbook playbook.yml -i
inventories/inventory-dev
```

Ad-Hoc parancsok

- ansible csoport parancs – [felhasználónév]
- Alapértelmezésben a prancsokat párhuzamosan futtatja le (-f kapcsoló)
- Valósidejű hibaelhárítás, diagnózis esetén
  - Erőforrások állapota (disk, ...)
  - Log fájlok elérése
  - Csoportok, felhasználók menedzselése
  - DNS, Host fájlok kezelése
  - Fájlok másolása
  - Gépek újraindítása
  - CRON Job-ok kezelése

```
$ ansible example -m ping -u [username]

$ ansible example -a "tree -m" -u [username]
```

Komplexebb host fájl

```
# Lines beginning with a # are comments, and are only included for
# illustration. These comments are overkill for most inventory files.

# Application servers
[app]
192.168.60.4
192.168.60.5

# Database server
[db]
192.168.60.6

# Group 'multi' with all servers
[multi:children]
app
db

# Variables that will be applied to all servers
[multi:vars]
ansible_ssh_user=vagrant
ansible_ssh_private_key_file=~/.vagrant.d/insecure_private_key
```

Példa parancsok

```
$ ansible multi -a "hostname"
```

```
# First run results:
192.168.60.5 | success | rc=0 >>
| rc=0 >>
orc-app2.dev

192.168.60.6 | success | rc=0 >>
| rc=0 >>
orc-db.dev

192.168.60.4 | success | rc=0 >>
| rc=0 >>
orc-app1.dev
```

```
# Second run results:
192.168.60.6 | success
orc-db.dev

192.168.60.5 | success
orc-app2.dev

192.168.60.4 | success
orc-app1.dev
```

Példa parancsok

```
$ ansible multi -a "df -h"
192.168.60.6 | success | rc=0 >>
Filesystem                Size  Used Avail Use% Mounted on
/dev/mapper/centos-root    19G 1014M   18G   6% /
devtmpfs                  111M     0  111M   0% /dev
tmpfs                     120M     0  120M   0% /dev/shm
tmpfs                     120M  4.3M  115M   4% /run
tmpfs                     120M     0  120M   0%
/sys/fs/cgroup
/dev/sda1                 497M  124M  374M  25% /boot
none                      233G  217G   17G  94% /vagrant

192.168.60.5 | success | rc=0 >>
Filesystem                Size  Used Avail Use% Mounted on
/dev/mapper/centos-root    19G 1014M   18G   6% /
devtmpfs                  111M     0  111M   0% /dev
tmpfs                     120M     0  120M   0% /dev/shm
tmpfs                     120M  4.3M  115M   4% /run
tmpfs                     120M     0  120M   0%
/sys/fs/cgroup
/dev/sda1                 497M  124M  374M  25% /boot
none                      233G  217G   17G  94% /vagrant
```

Példa parancsok

```
$ ansible app -b -m yum -a "name=MySQL-python
state=present"
$ ansible app -b -m yum -a "name=python-setuptools
state=present"
$ ansible app -b -m easy_install -a "name=django
state=present"
```

```
$ ansible db -b -m yum -a "name=mariadb-server
state=present"
$ ansible db -b -m service -a "name=mariadb state=started \
enabled=yes"
$ ansible db -b -a "iptables -F"
$ ansible db -b -a "iptables -A INPUT -s 192.168.60.0/24 -p
tcp \
-m tcp --dport 3306 -j ACCEPT"
```

```
$ ansible app -b -a "service ntpd restart" --limit
"192.168.60.4"
```

```
$ ansible multi -m copy -a "src=/etc/hosts dest=/tmp/hosts"
```

Forgatókönyvek

- Konfigruációs fájlok
- Egymásba ágyazhatóak
- shell alapú (könnyű meglévő szkripteket bevonni)
- -check
- YAML szintaxis

```
1 ---
2 - hosts: all
3   become: yes
4   tasks:
5   - name: Ensure NTP (for time synchronization) is
installed.
6     yum: name=ntp state=present
7   - name: Ensure NTP is running.
8     service: name=ntpd state=started enabled=yes
```

```
if ! rpm -qa | grep -qw ntp; then
    yum install ntp
fi
```

```
# Start ntpd if it's not already running.
if ps aux | grep -v grep | grep "[n]tpd" > /dev/null
then
    echo "ntpd is running." > /dev/null
else
    systemctl start ntpd.service > /dev/null
    echo "Started ntpd."
fi
# Make sure ntpd is enabled on system startup.
systemctl enable ntpd.service
```

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Példa

```
---
- hosts: all

  tasks:
    - name: Install Apache.
      command: yum install --quiet -y httpd
httpd-devel
    - name: Copy configuration files.
      command: >
        cp httpd.conf
/etc/httpd/conf/httpd.conf
    - command: >
        cp httpd-vhosts.conf
/etc/httpd/conf/httpd-vhosts.conf
    - name: Start Apache and configure it to
run at boot.
      command: service httpd start
    - command: chkconfig httpd on
```

Idempotens

```
---
- hosts: all
  become: yes

  tasks:
    - name: Install Apache.
      yum: name={{ item }} state=present
      with_items:
        - httpd
        - httpd-devel
    - name: Copy configuration files.
      copy:
        src: "{{ item.src }}"
        dest: "{{ item.dest }}"
        owner: root
        group: root
        mode: 0644
      with_items:
        - src: "httpd.conf"
          dest: "/etc/httpd/conf/httpd.conf"
        - src: "httpd-vhosts.conf"
          dest: "/etc/httpd/conf/httpd-vhosts.conf"
    - name: Make sure Apache is started now and at boot.
      service: name=httpd state=started enabled=yes
```

Forgatókönyvek utasítások

- Belső függőségek
  - handlers

```
tasks:
  - name: Enable Apache rewrite module.
    apache2_module: name=rewrite state=present
    notify: restart apache

handlers:
  - name: restart apache
    service: name=apache2 state=restarted
```

```
- name: Rebuild application configuration.
  command: /opt/app/rebuild.sh
  notify:
    - restart apache
    - restart memcached

handlers:
  - name: restart apache
    service: name=apache2 state=restarted
    notify: restart memcached

  - name: restart memcached
    service: name=memcached state=restarted
```

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Forgatókönyvek utasítások

- változók

```
- name: Add an environment variable to the
remote user's shell.
  lineinfile: "dest=~/.bash_profile
regexp=^ENV_VAR= \
  line=ENV_VAR=value"
```

```
vars:
  var_proxy:
    http_proxy: http://example-proxy:80/
    https_proxy: https://example-proxy:443/
    [etc...]

tasks:
- name: Download a file, using example-proxy as
a proxy.
  get_url:
url=http://www.example.com/file.tar.gz
dest=~/Downloads/
  environment: var_proxy
```

```
ansible-playbook example.yml --extra-vars
"foo=bar"
```

```
1 ---
2 - hosts: example
3   vars_files:
4     - "apache_default.yml"
5     - "apache_{{ ansible_os_family }}.yml"
6   tasks:
7     - service: name={{ apache }}
  state=running
```

Forgatókönyvek utasítások

- Jinja2 kifejezések

```
1 - name: Do something only for version 4 of
the software.
2   [task here]
3   when: software_version.split('.')[0] == '4'
```

```
- yum: name=mysql-server state=present
  when: (is_db_server is defined) and
is_db_server
```

Forgatókönyvek utasítások

- Delegáció, várakozás

```
- name: Remove server from load balancer.
  command: remove-from-lb {{ inventory_hostname }}
  delegate_to: 127.0.0.1
```

```
- name: Wait for webserver to start.
  local_action:
    module: wait_for
    host: "{{ inventory_hostname }}"
    port: "{{ webserver_port }}"
    delay: 10
    timeout: 300
    state: started
```

Forgatókönyvek utasítások

- Címkék

```
1 ---
2 # You can apply tags to an entire play.
3 - hosts: webservers
4   tags: deploy
5
6   roles:
7     # Tags applied to a role will be applied
to tasks in the role.
8     - { role: tomcat, tags: ['tomcat',
'app'] }
9
10  tasks:
11    - name: Notify on completion.
12      local_action:
13        module: osx_say
14        msg: "{{inventory_hostname}} is
```

```
1 $ ansible-playbook tags.yml --tags
"tomcat,say"
```

```
1 $ ansible-playbook tags.yml --skip-tags
"notifications"
```

Szerepkörök, beágyazás

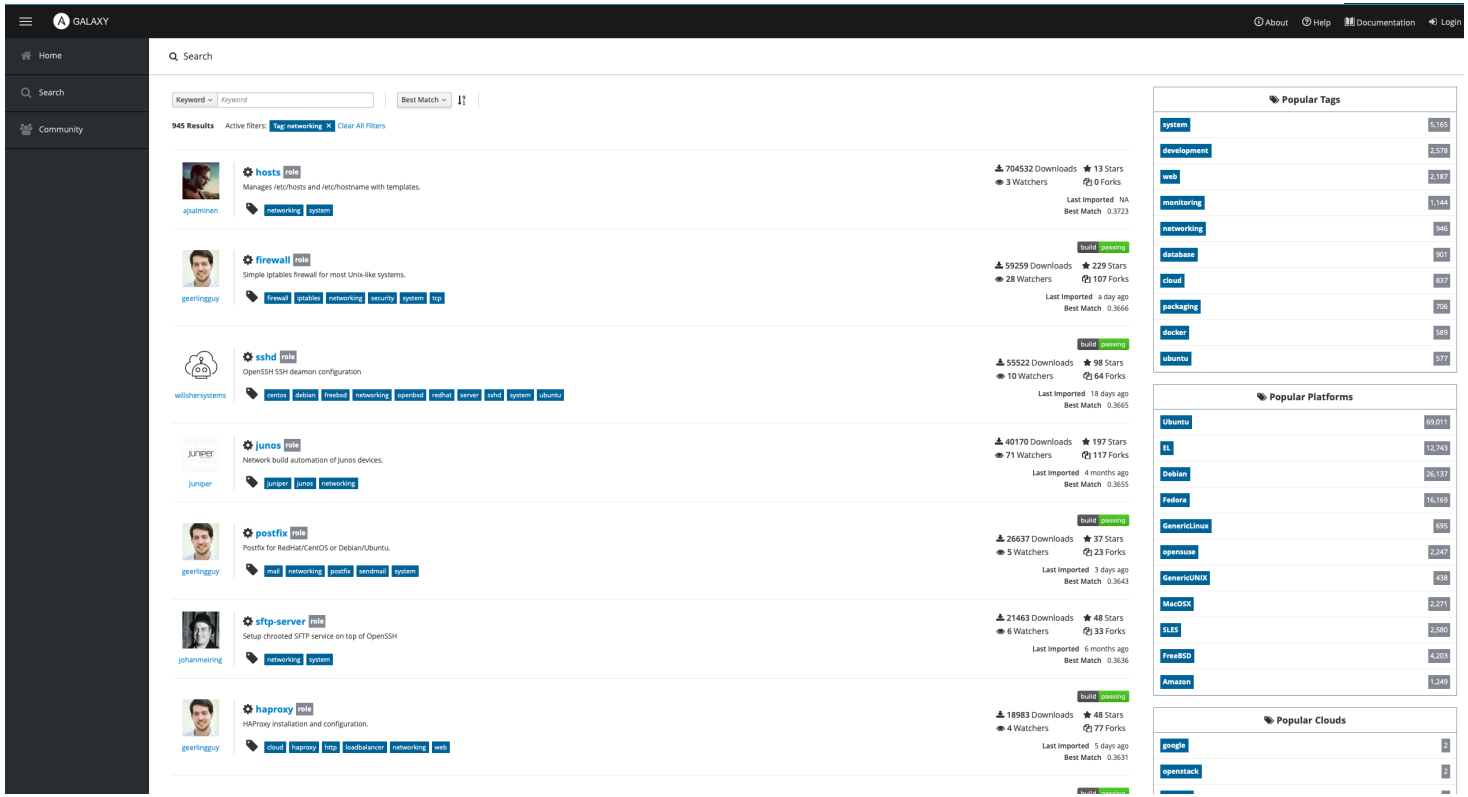
```
tasks:
  - include: included-playbook.yml

- hosts: all
  remote_user: root

  tasks:
    [...]

- include: web.yml
- include: db.yml
```

```
role_name/
  meta/
  tasks/
```

```
1 ---
2 - hosts: appservers
3   roles:
4     - yum-repo-setup
5     - firewall
6     - nodejs
7     - app-deploy
```

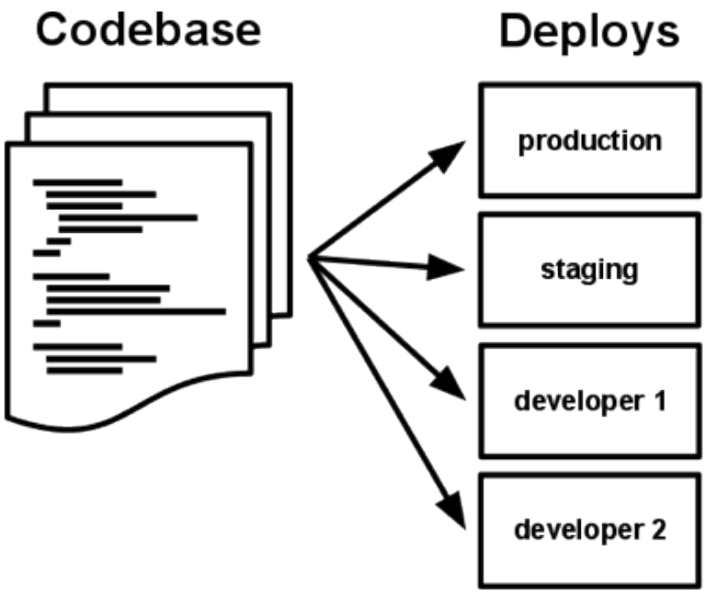
Ansible galaxy

```
$ ansible-galaxy install geerlingguy.apache
geerlingguy.mysql geerli\
ngguy.php
```



12 faktoros alkalmazás

- Kódbázis
  - Egy repó – egy alkamazás
  - Nem
    - Több repó – egy alkalmazás (elosztott rendszer)
    - Több alkalmazás – egy repó (függőségek kezelése)



12 faktoros alkalmazás

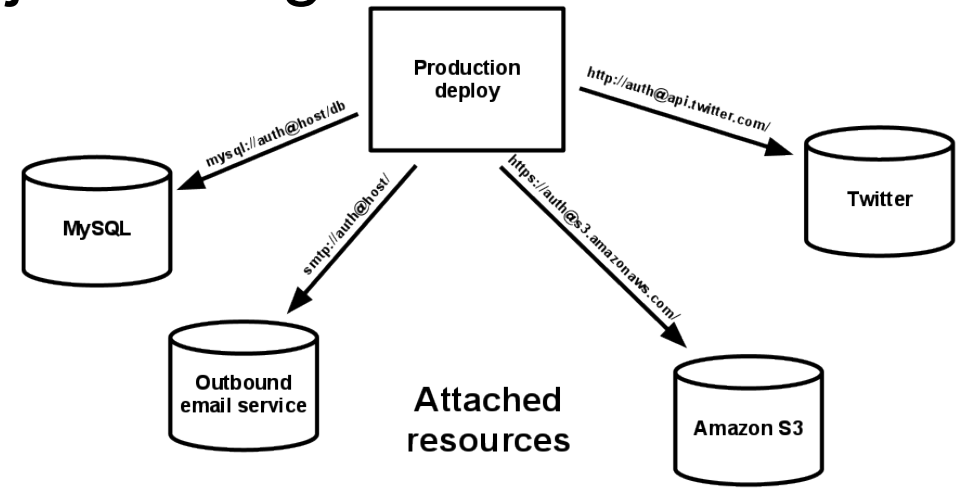
- Függőségek
  - Nem számíthatunk rendszerszintű meglévő csomagokra (pl.: curl, ...)
  - Minden függőséget deklarálni kell
  - A függőségeket el kell különíteni (implicit függőségek)
  - Production === staging

12 faktoros alkalmazás

- Konfiguráció
  - Ez különbözhet a telepítések között
    - Erőforrások elérése (Cassandra, SQL, ...)
    - Azonosító információk
    - Telepítés előtti változók (hostname,...)
  - Akkor jó a bármikor nyílt forrással publikálható
  - Környezeti változók alkalmazása
  - Nem csoportosítjuk ezeket dev, ... Környezetek alapján

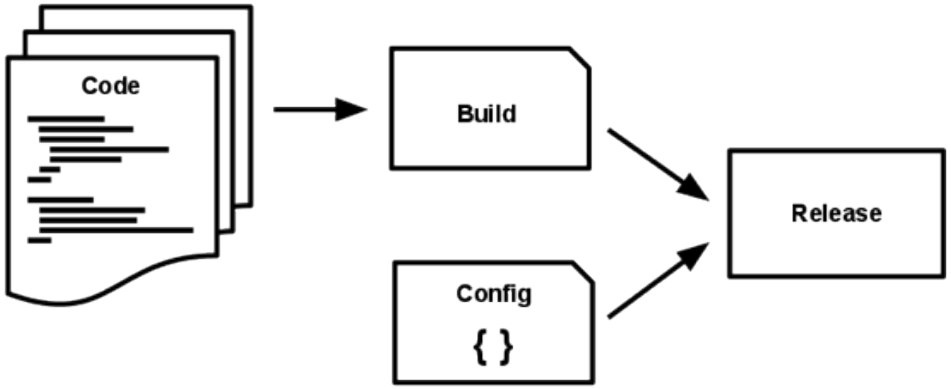
12 faktoros alkalmazás

- Támogató szolgáltatások
  - Kezeljük ezeket erőforrásokként (cserélhető, lecsatolható, ...)
  - Nincs különbség a saját és mások által nyújtott szolgáltatások között



12 faktoros alkalmazás

- Build, release, run
  - Build
    - Kód repó átalakítása futtaható kóddá
    - A függőségeke itt tölti le
    - Bináris az eredménye
  - Release
    - A konfigurációval együtt telepíti a futtató környezetbe
  - Run
    - Az alkalmazás adott verziójét futtatja
    - Lehető legegyszerűbbnek kell lennie (újraindítás, ...)
- Minden release-nek egyedi ID-vel kell rendelkeznie
- Minden változás új release-t eredményez



12 faktoros alkalmazás

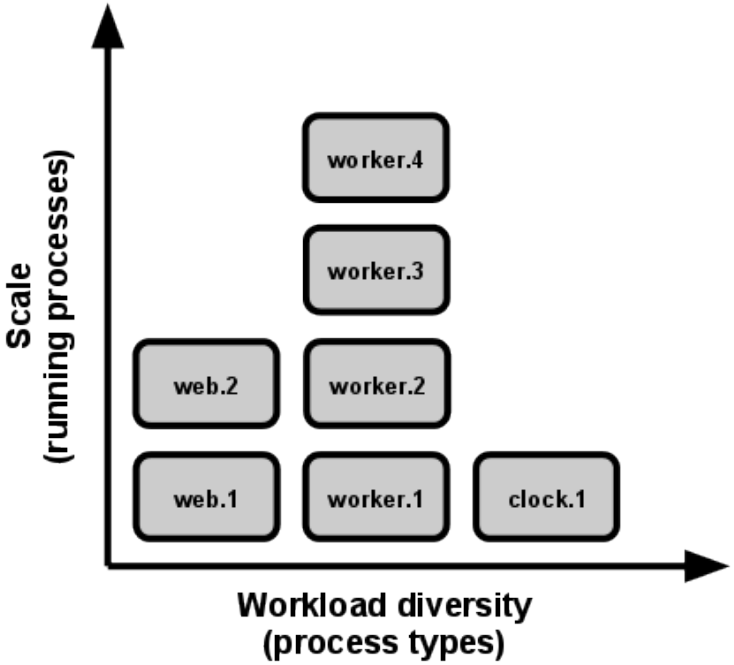
- Processzusok
  - Az alkalmazás egy vagy több processzus formájában fut
  - Ezek nem osztanak meg közvetlenül semmit sem (share-nothing)
  - A közös adatok a támogató rendszereken keresztül kerülnek át
  - A helyi perzisztens tárak csak átmenetileg használhatóak
  - Sticky session-ra nem számíthatunk

12 faktoros alkalmazás

- Port kötés
  - Teljesen független a környezettől
  - Nem számít webszerver előtétre
  - A http-mint IP:port párost szolgáltatja
  - Egy alkalmazás lehet egy másik támogató alkalmazása

12 faktoros alkalmazás

- Párhuzamosság
  - A processzusok az első osztályú elemei
  - Nem alapulhat démonokra vagy PID fájlok írására
  - Az operációs rendszer processz kezelésére kell építenie (pl.: systemd)



12 faktoros alkalmazás

- Eldobhatóság
  - A processzusok eldobhatóak
  - Bármikor értesítés nélkül leállíthatóak
  - Minimalizálni kell az indulási időt
  - Szépen kell leállítanunk (SIGTERM)
  - Robosztusnak kell lennie a hirtelen leállásokkal szemben

12 faktoros alkalmazás

- Dev/prod egyenlő
  - Idő
  - Személy
  - Eszköz
- Folyamatos telepítés
- Azonos támogató szolgáltatások

12 faktoros alkalmazás

- Naplózás
  - Eseményfolyamként kell kezelni
  - A helyi operációs rendszer standard megoldását kell használni (STDOUT)
  - Nem foglalkozunk ezzel begyűjtésével

12 faktoros alkalmazás

- Adminisztrációs folyamatok
  - One-off folyamatok
    - Azonos környezet
    - Adott release / konfig/kódbázis
    - A kódbázissal közösen kezelendő
    - REPL környzetekek