Platform-as-a-Service és Telekom felhő

Simon Csaba

Budapesti MŤszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Távközlési és Médiainformatikai Tanszék

INTRO – CLOUDS REVISITED

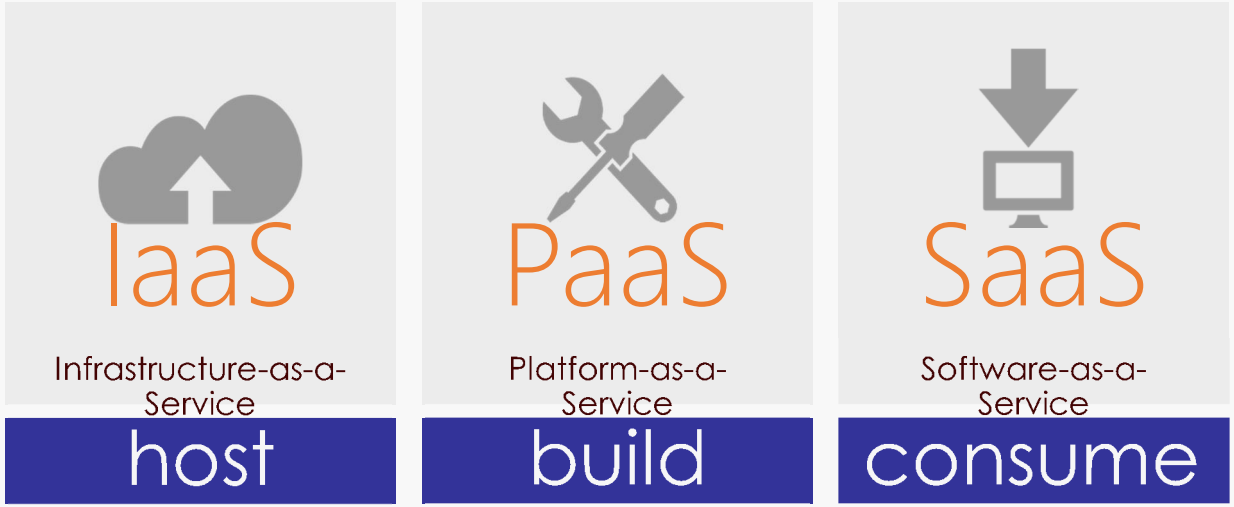
A felhő alapú rendszerek értelme és szerepe az IT világban

Felhő szerepe – üzleti oldalról

ELVÁRT ELŐNY/HASZON:

Ehhez az infrastruktúra oldaláról szükséges:

Felhő terminológia felhasználási mód szerint



A felhő típusok meghatározása

Telepítés/üzemeltetés módja

- Private Cloud
- Public Cloud
- Hybrid Cloud

Szolgáltatás módja

- Infrastructure as a Service (IaaS)
- Platform as a Service PaaS
- Software as a Service SaaS

MI A PAAS?

Platform as a Service – the services

Szolgáltatás: mit ad nekünk a PaaS?

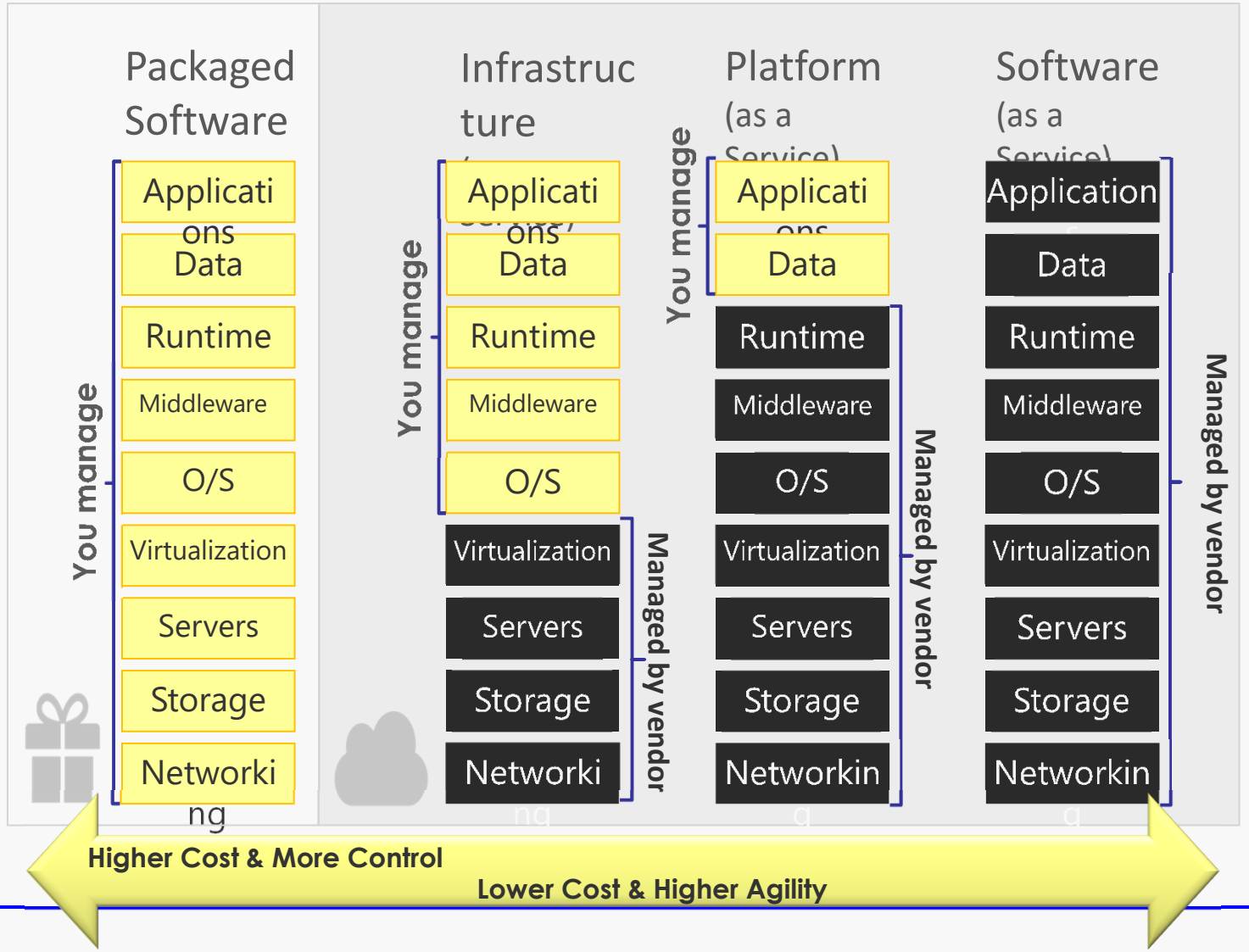
- Telepítés - app deployment
- Skálázhatóság - scaling (horizontal, vertical, auto)
- Terhelésmegosztás - load balancing
- Felügyelet - health monitoring, auto recovery
- Naplózás - logging service
- Eszköztár - external/internal services, marketplace

A PaaS előnyei – a Platform szolgáltatásai

- Deployment
- Load Balancing (LB)
- High Availability (HA)
- Log aggregation
- Scaling
- Image mgmt
  - » Lib, kernel ver.
  - » Biztonsági frissítések

(Tenantos problem: just develop your own app)
- ElőnyA felhasználó a saját alkalmazásának kidolgozására koncentrálhat

Compare the *aaS-es



Az *aaS modellek összehasinlítása

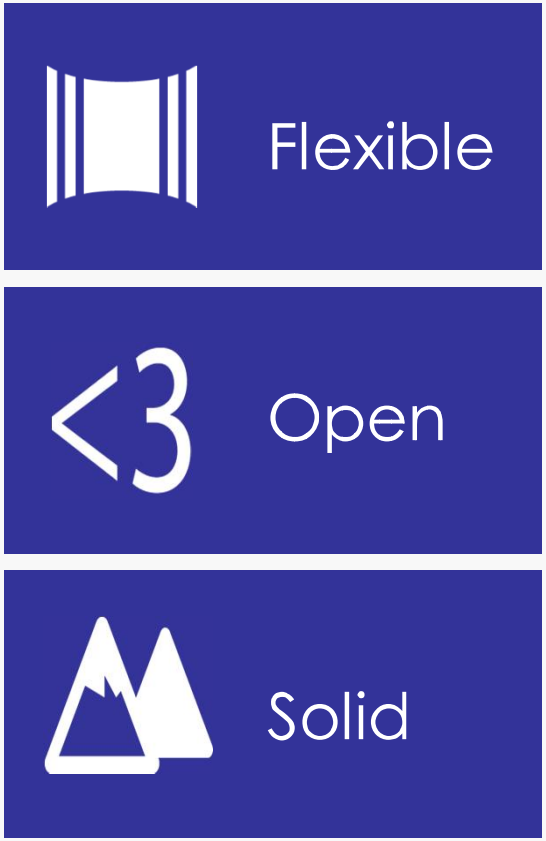
- IaaS: a végfelhasználó konfigurálja az operációs rendszert is, az alkalmazásokat és a környezetet is (pl. biztonsági beállítások)
- PaaS: csak a fejlesztésre, tesztelésre és a telepítésre (deployment) kell figyelni, valamint a karbantartásra
- SaaS: a szoftver testre szabását kell megoldani, felhasználási intenzitás alapján kell fizetni (pl. felhasználók száma, adatforgalom mértéke vagy tranzakciók száma)

A PaaS dióhéjban

Teljes körŤ és komplex szolgáltatás halmaz

Gyors alkalmazás kezelés
- build
- telepítés
- menedzsment

Globális (több-helyszínen mŤködŘ) hálózaton



PaaS rendszerek nagyon változatosak

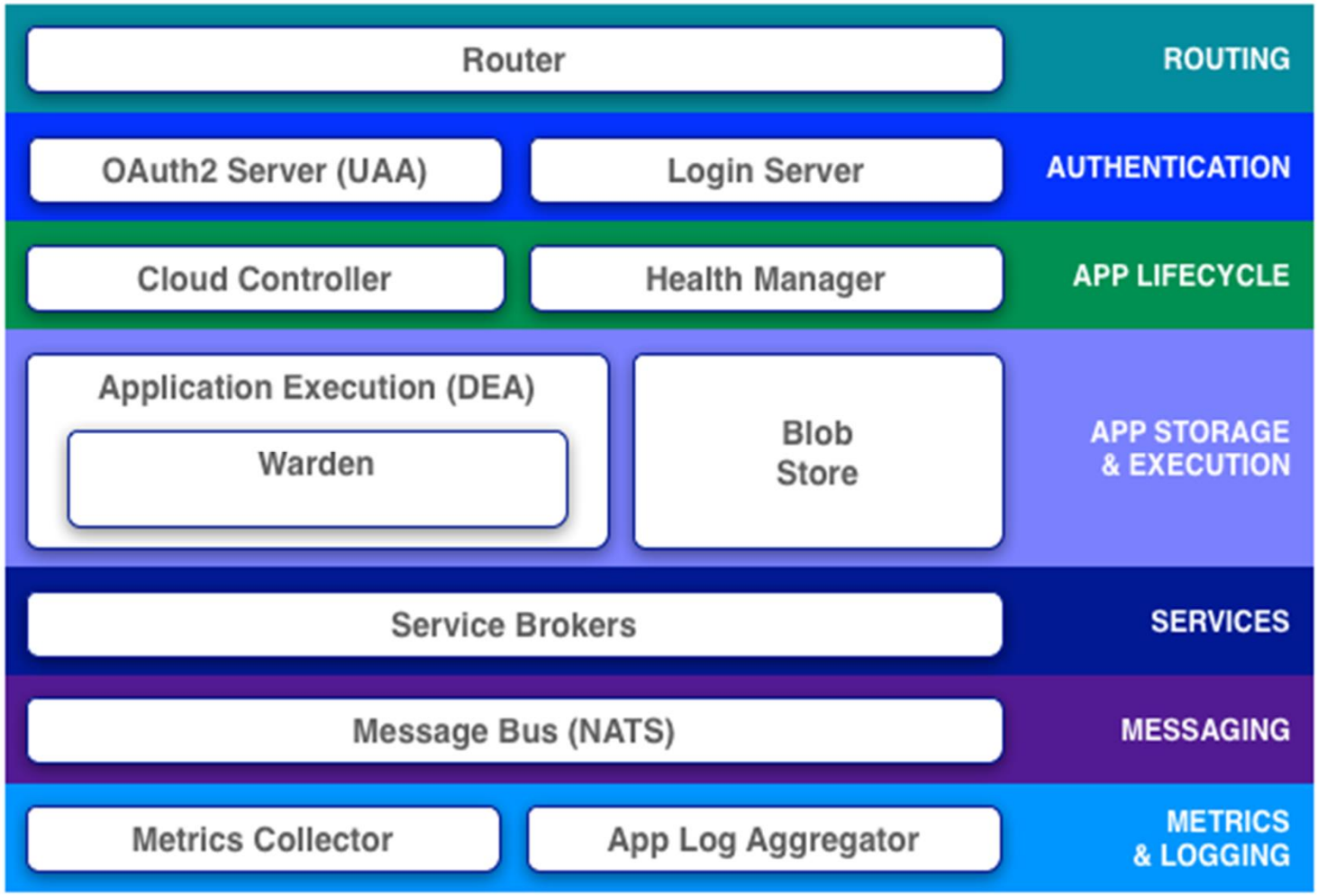
- Services that help develop and test apps
  - infrastructure is maintained by the provider
- Reduced infrastructure complexity
  - more effective overall application development
- Runtime environments are usually lock-in free
  - but might create lock-ins to provider specific infrastructure
- Usually simple network topology and access control
  - build your services as they would be open to the Internet
- The features and services provided vary a lot
  - from simple customizable runtime (CloudFoundry) to full marketplace of services (Heroku)

PaaS, mint egy eszköztár (Toolchain as a Service)

- Manage your project
  - Trello, Jira OnDemand, Sprint.ly, PivotalTracker, ...
- Create your code
  - Cloud9, Koding, Nitrous, ...
- Host your code
  - GitHub, Bitbucket, ...
- Build your code
  - Codeship, Travis CI, CloudBees, Drone, ...
- Test your code
  - BrowserStack, Sauce Labs, Xamarin Test Cloud, Blitz, ...
- Distribute your code
  - npm, Bintray, Maven Central, PyPI, Docker Hub, ...

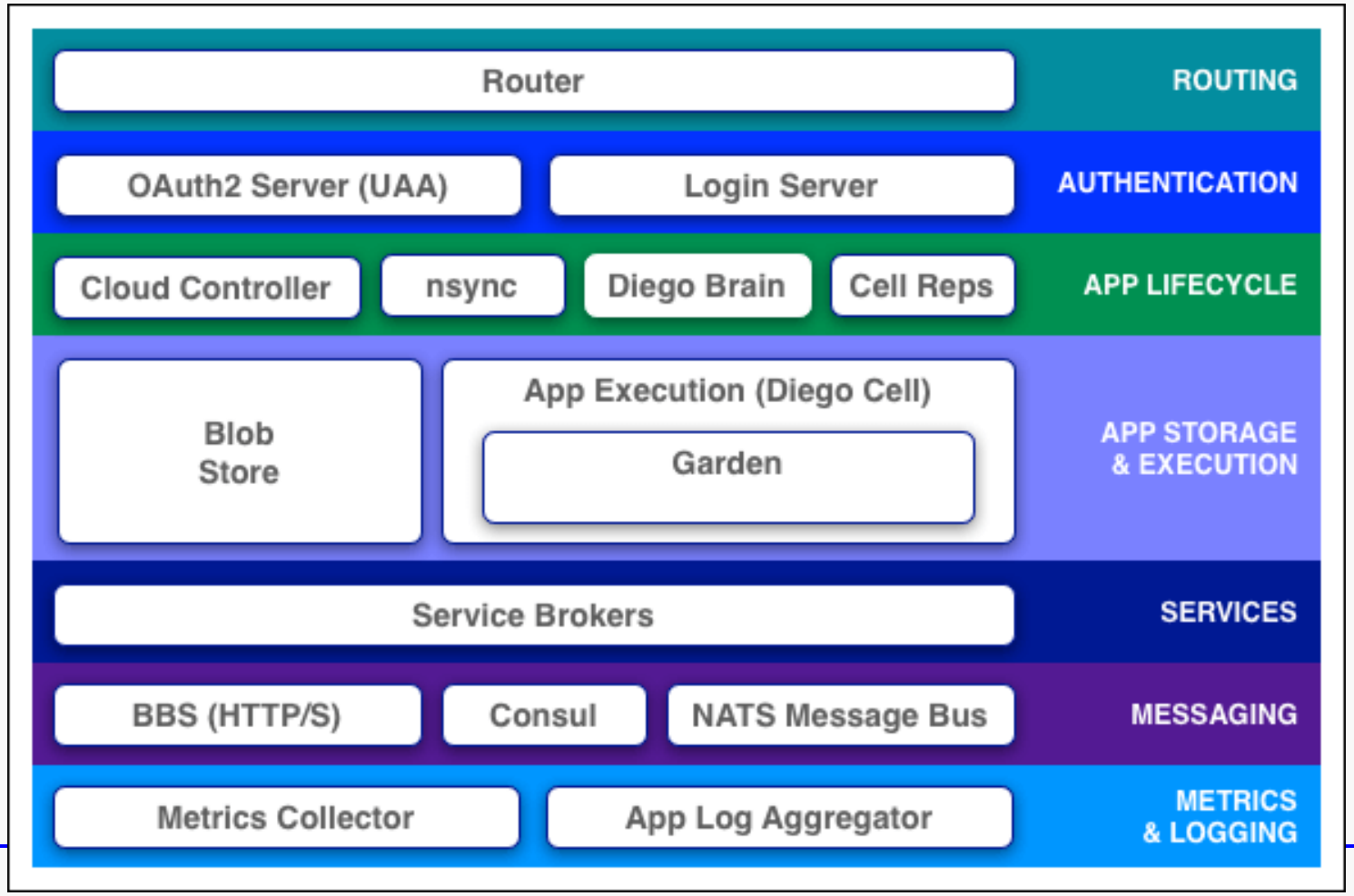
PAAS ARCHITEKTÚRA

Cloud Foundry - Architecture

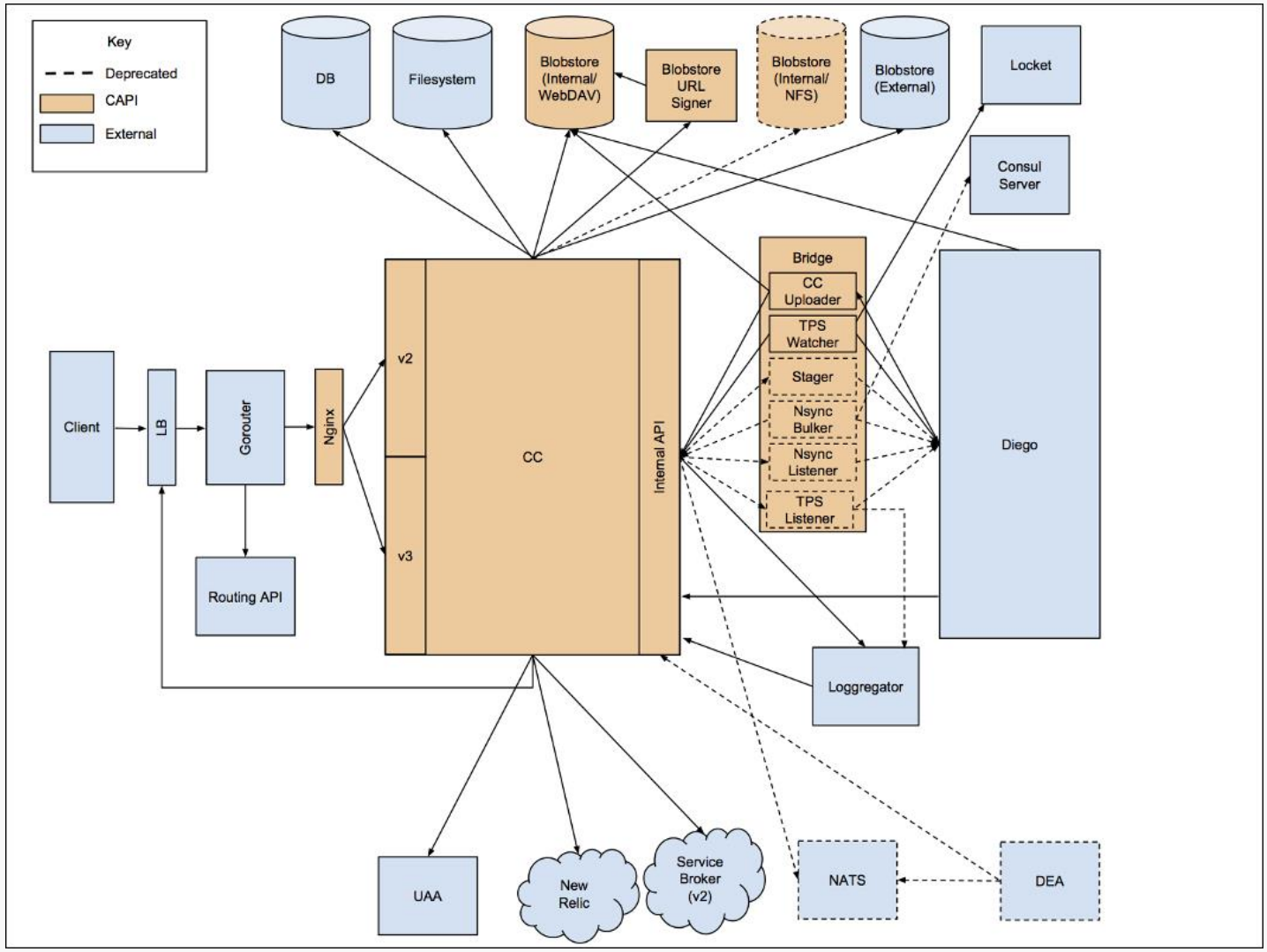


Cloud Foundry - Architecture

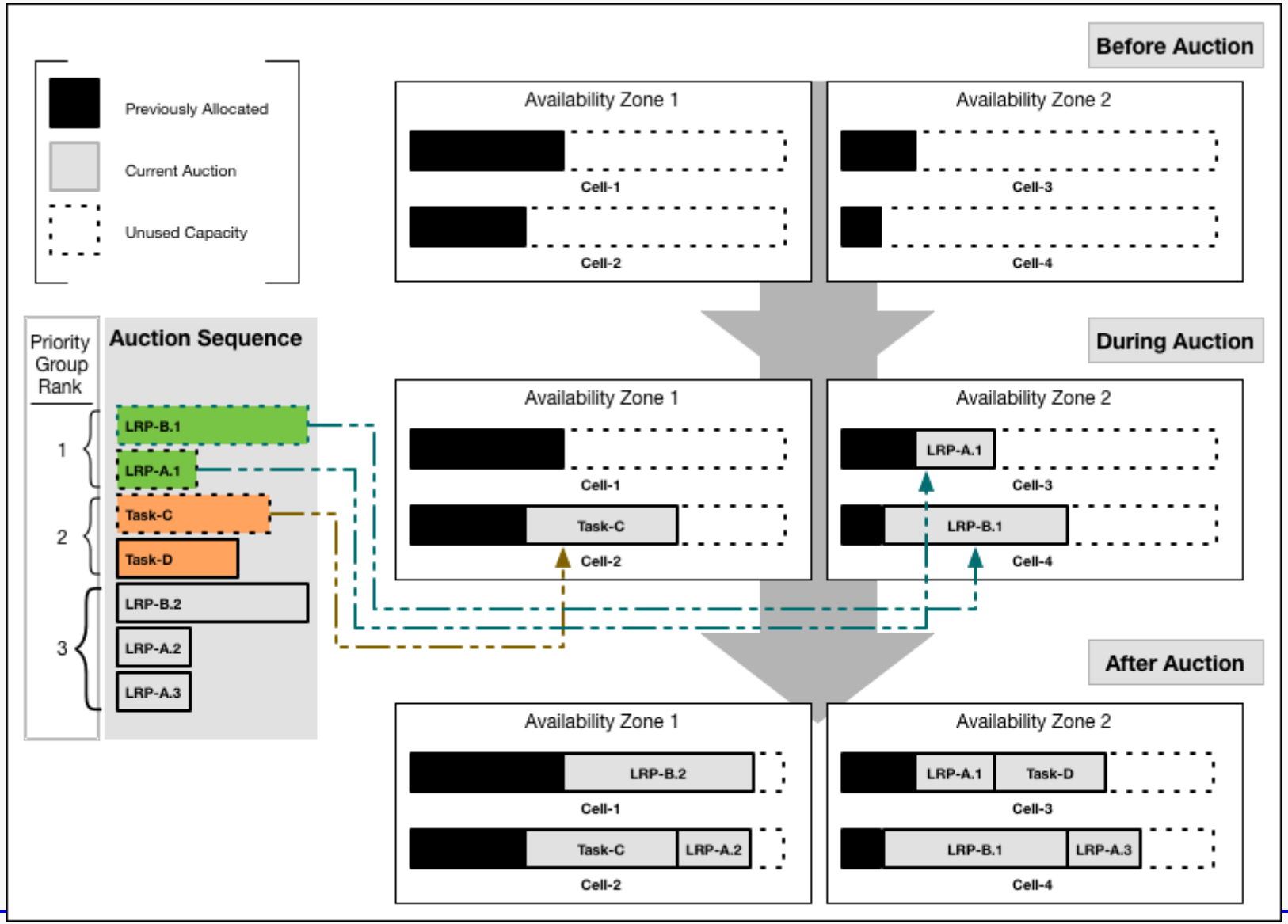
Diego ï a CF konténer menedzsment rendszere
Feladat végzése (task deployment)
Garden konténerek a Diego Cell VM-ben



CF Cloud Controller



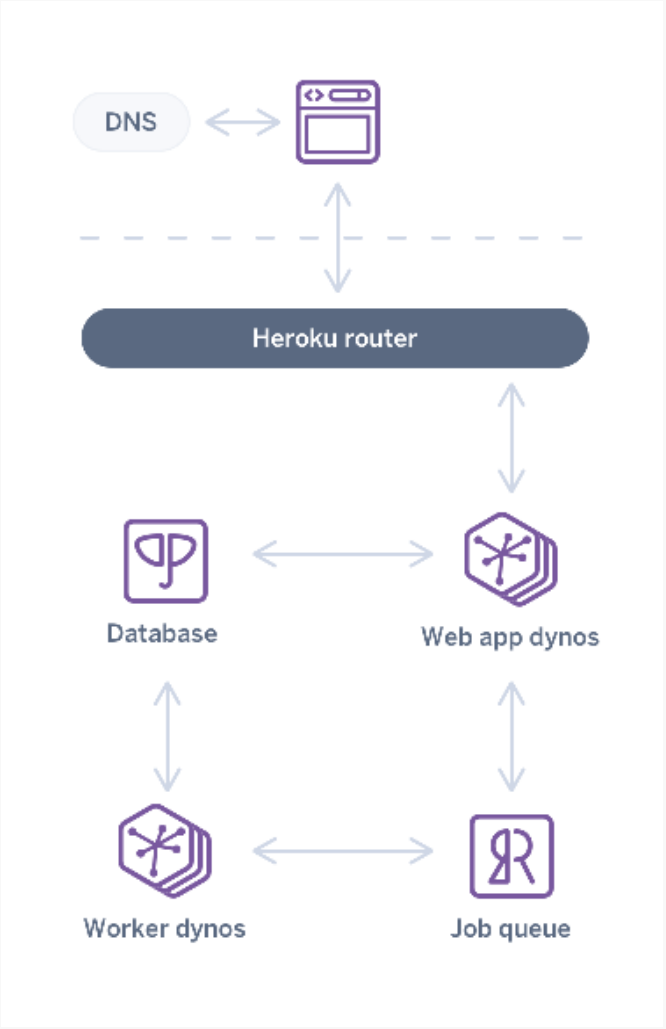
Diego Auction – CF Load Balancer



Heroku – egyike az első PaaS szolgáltatóknak

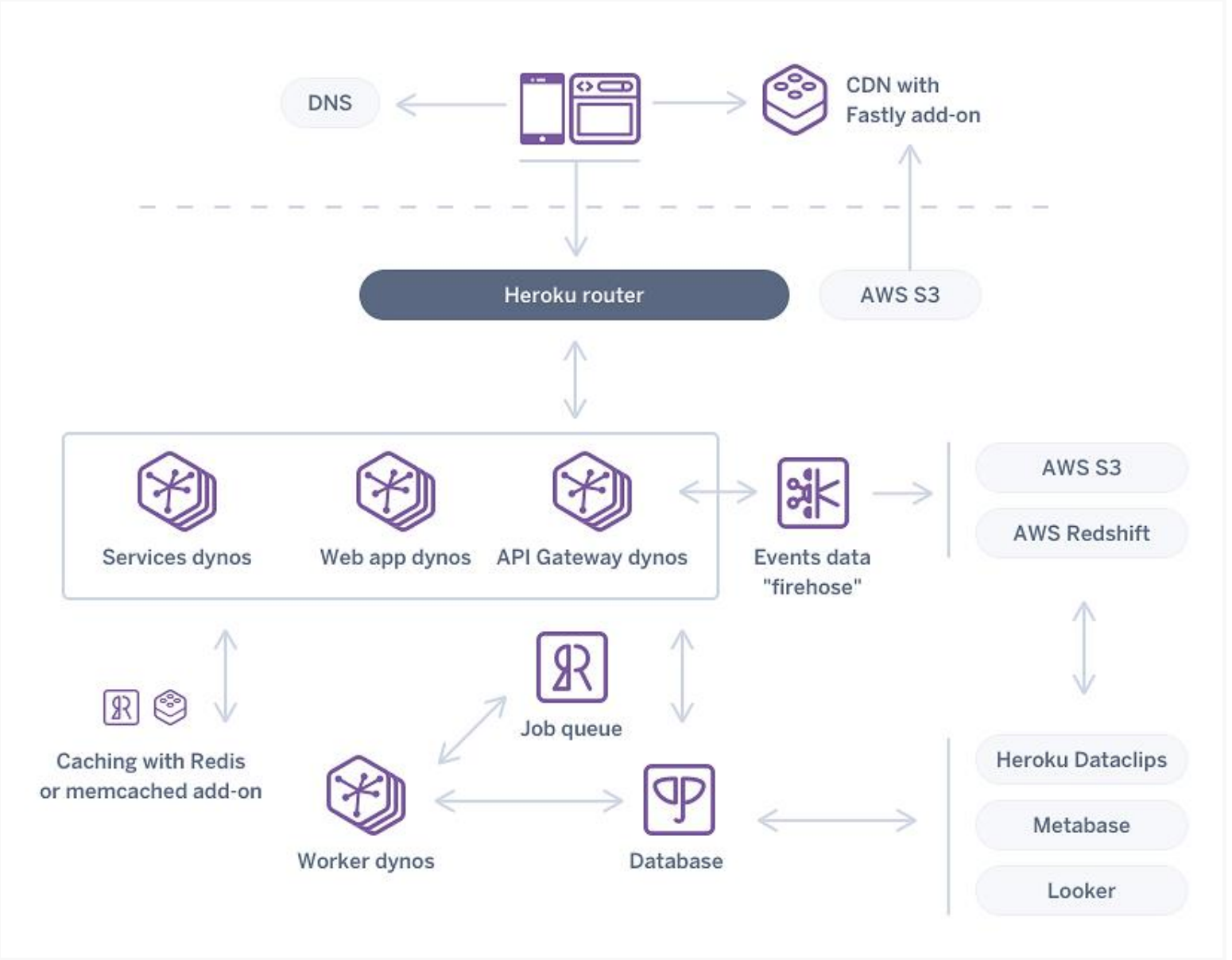
HEROKU

- A web-app csak egy gateway
- Sorba rendezett kérések
- Back-end végzi a feladatot



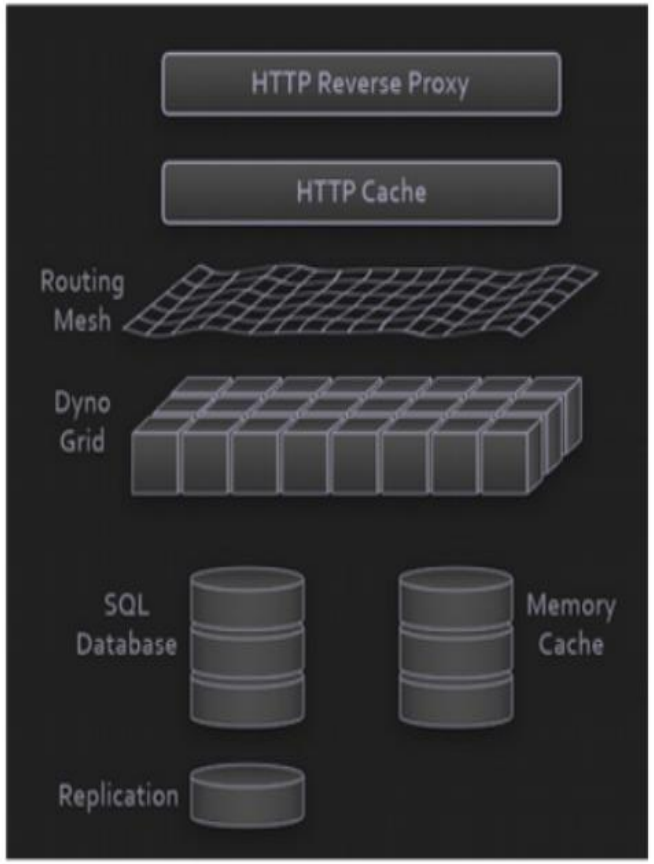
- Változatos eszközök (okostelefon)
- Cache-elt válaszok, előfeldolgozás
- Interfész más felhő alapú szolgáltatásokkal
- APIs = az új user interface
  - Elsősorban fejlesztők felé
  - Nem csak tech szolgáltatók részéről
  - Új, innovatív kis csapatok „ráépülhetnek”
    - Pl. BKK Futár API alapján új szolgáltatás
    - „creative extension”, „leverage”

Modern Heroku alapú rendszer



Heroku architecture

- Reverse Proxy by nginx
  - terminates SSL
  - forwards to cache layer
- HTTP Cache by Varnish
  - returns cached pages immediately
  - forwards to routing mesh
- Routing Mesh written in Erlang
  - routes to an existing dyno
  - spawns a dyno if none available
- Dyno Grid ('railgun' servers)
  - AWS hosted EC2 instances
  - multiple dynos per server



Heroku termonology

- **Application** source code and description of any dependencies
- **Procfile** list of process types – named commands to be executed
- **Deployment** sending application to Heroku using git or dropbox
- **Buildpack** compilation process that creates a slug from application
- **Slug** bundle of application, language runtime and compilation output
- **Dyno** isolated, virtualized Linux container for application runtime
- **Release** append-only ledger of slugs, config vars and add-ons
- **Config var** configuration data hosted independently of source code
- **Add-on** easily attachable third party cloud services
- **Logplex** collates logs from all running dynos and other components

Application and Procfile

`Procfile`

- Your app's web server
- Multiple types of worker processes
- A singleton process, such as a clock
- Tasks to run before a new release is deployed

If you are using heroku.yml as your build manifest, a Procfile is not required.

`heroku.yml`

- `setup` - Specifies the add-ons and config vars to create during app provisioning
- `build` - Specifies the `Dockerfile` to build
- `release` - Specifies the release phase tasks to execute
- `run` - Specifies process types and the commands to run for each

Buildpacks

- Buildpack compilation process that creates a slug from application

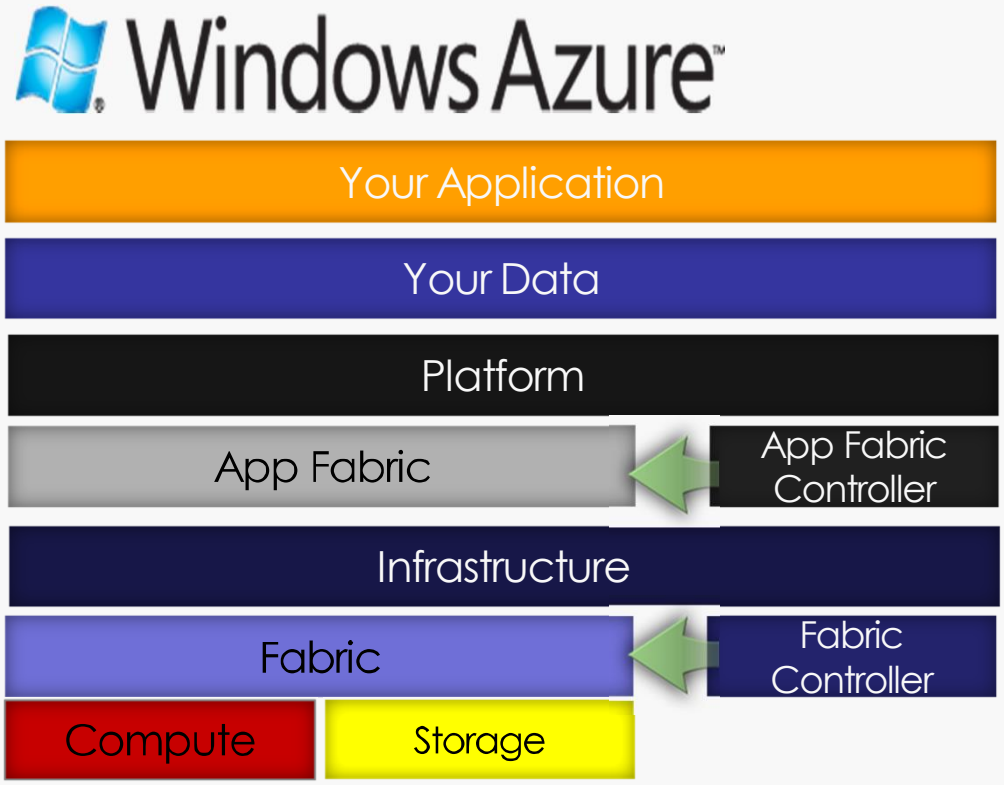
Programming language of the application

Buildpack	Shorthand
Ruby	heroku/ruby
Node.js	heroku/nodejs
Clojure	heroku/clojure
Python	heroku/python
Java	heroku/java
Gradle	heroku/gradle
Grails 3.x	heroku/gradle
Scala	heroku/scala
Play 2.x	heroku/scala
PHP	heroku/php
Go	heroku/go

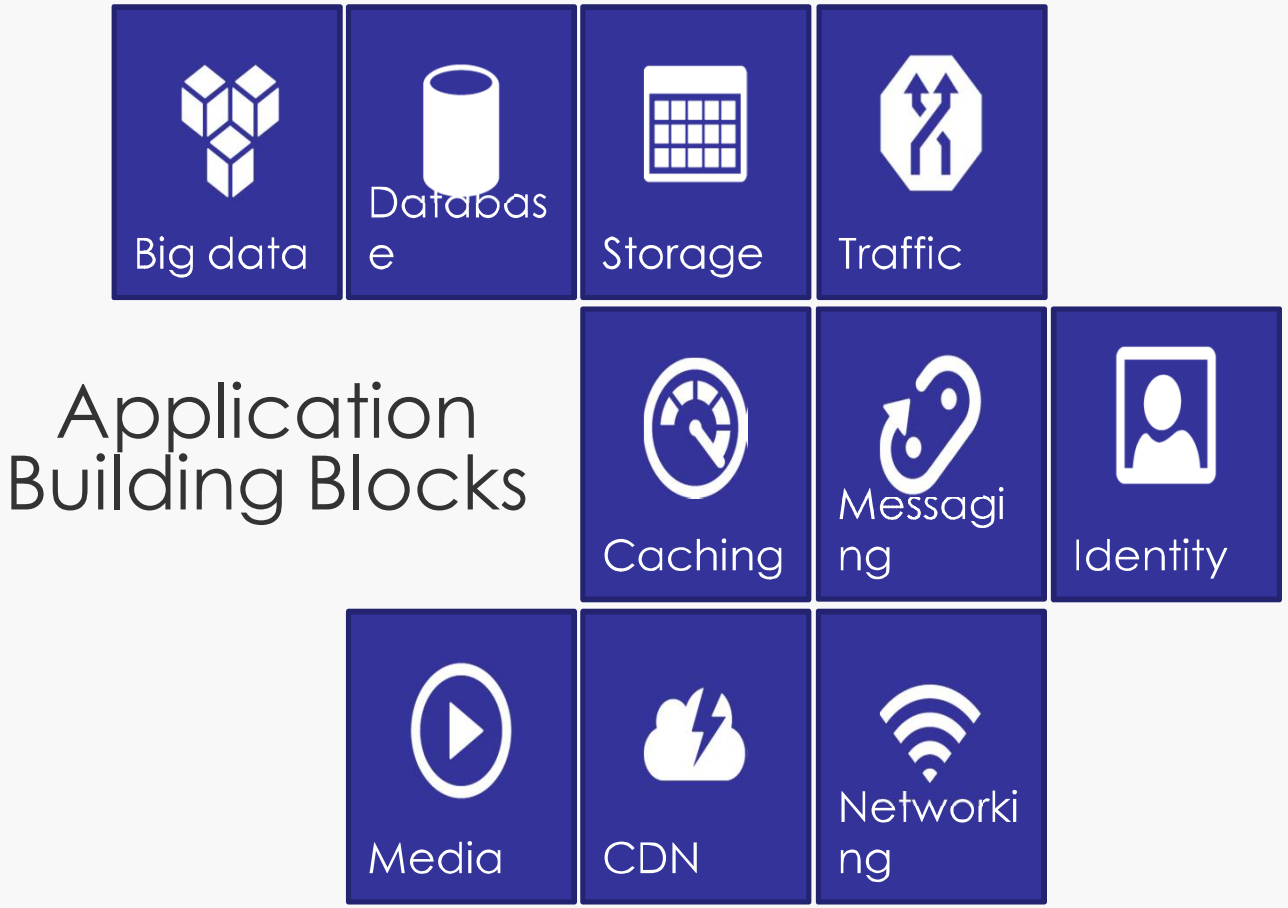
Dyno Type ~ OpenStack flavor

Dyno Type	Memory (RAM)	CPU Share	Compute	Dedicated	Sleeps
free	512 MB	1x	1x-4x	no	yes
hobby	512 MB	1x	1x-4x	no	no
standard-1x	512 MB	1x	1x-4x	no	no
standard-2x	1024 MB	2x	4x-8x	no	no
performance-m	2.5 GB	100%	11x	yes	no
performance-l	14 GB	100%	46x	yes	no

Winazure


Windows Azure
Your Application
Your Data
Platform
App Fabric
App Fabric Controller
Infrastructure
Fabric
Fabric Controller
Compute
Storage

Azure Building Block Services



Google Platform as a Service (PaaS)

Google App Engine (**GAE**) is a Platform as a Service (PaaS) cloud computing platform for developing and hosting web applications in Google-managed data centers.

Google App Engine lets you run web applications on Google's infrastructure.

Easy to build.
Easy to maintain.
Easy to scale as the traffic and storage needs grow.

Free ???

Yes, free for upto 1 GB of storage and enough CPU and bandwidth to support 5 million page views a month. 10 Applications per Google account.

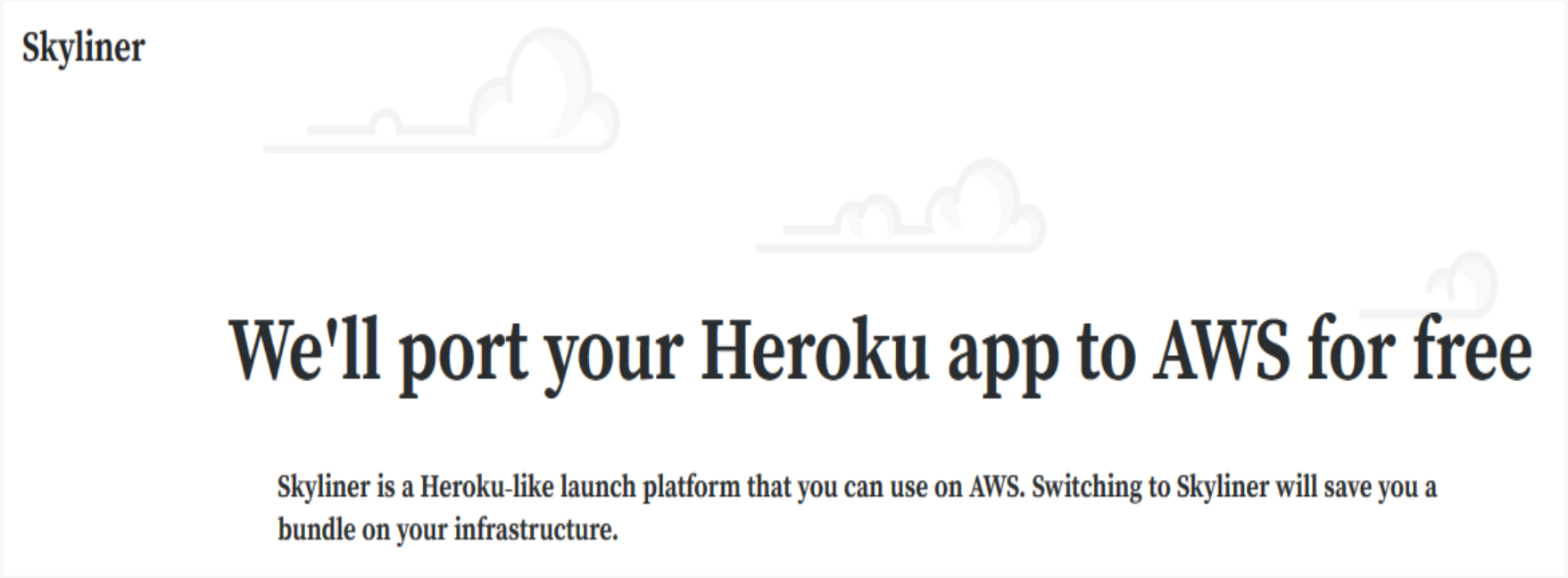
AWS Elastic Beanstalk

- AWS Elastic Beanstalk http://aws.amazon.com/elasticbeanstalk/ is a PaaS (Platform as a Service) service from Amazon Web Services that allows users to create applications and push them to a set of AWS services, including Amazon EC2, Amazon S3, Amazon Simple Notification Service (SNS), Amazon CloudWatch, auto scaling, and elastic load balancers

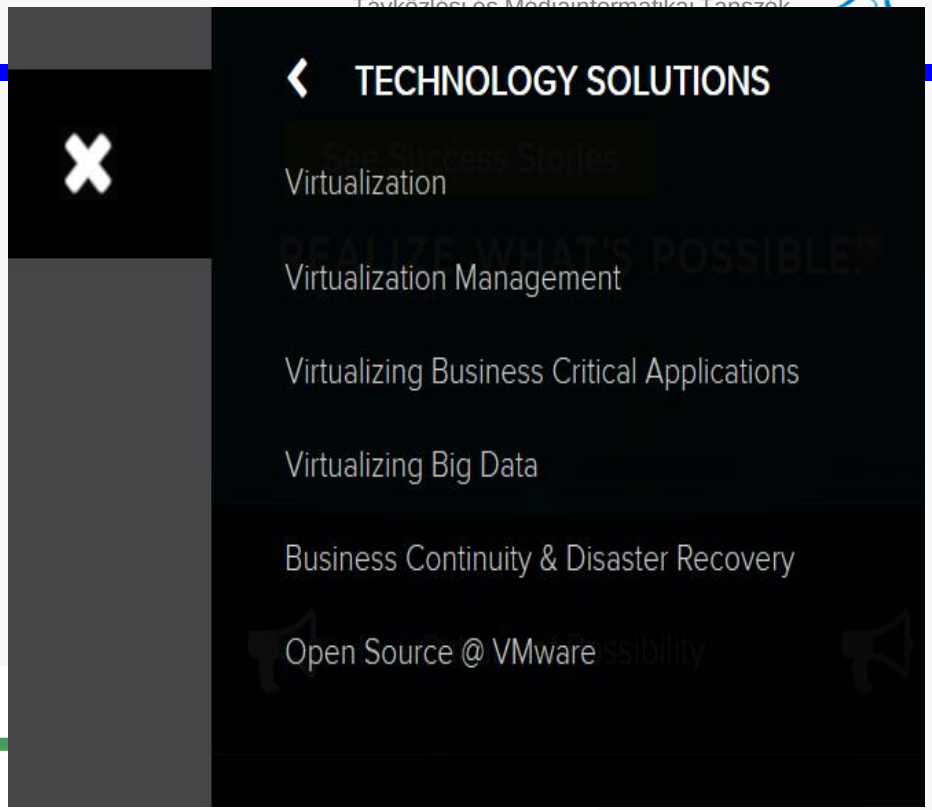
Supports Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers such as Apache, Nginx, Passenger, and IIS.

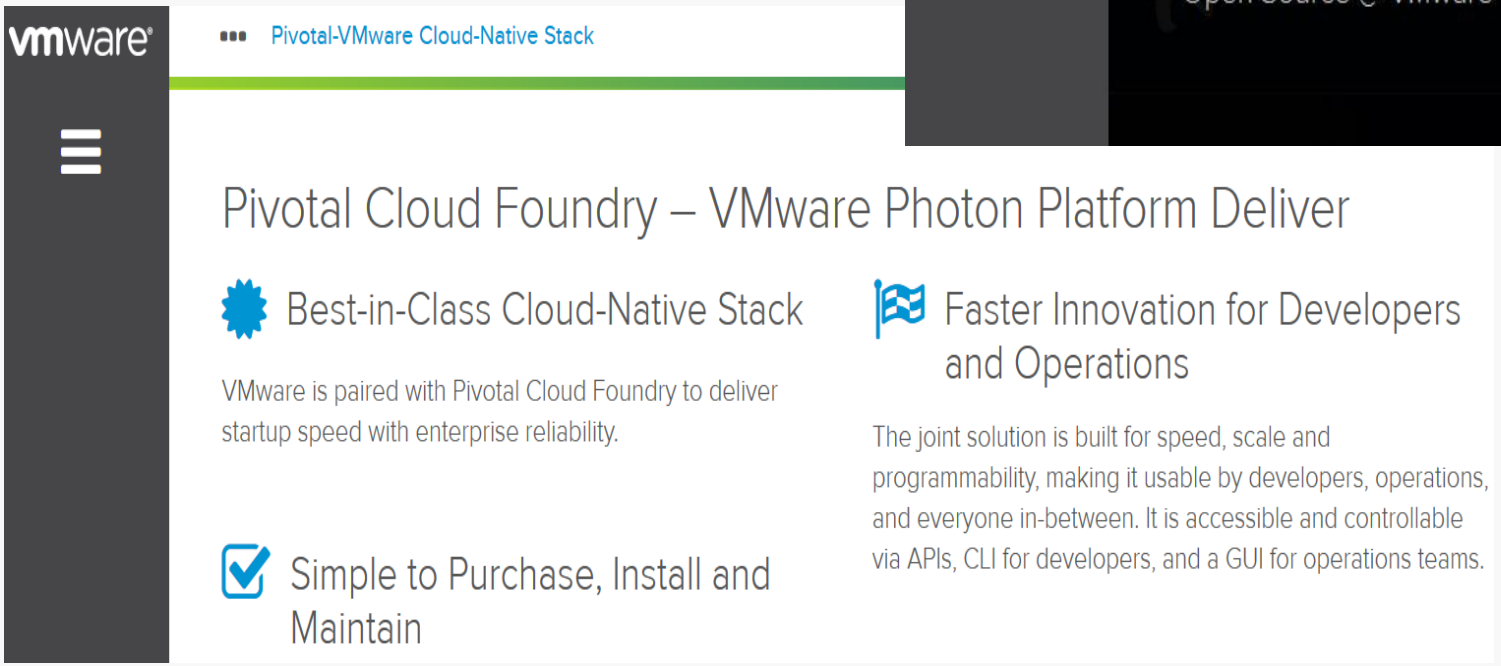
Éles versenyben „elkel” a segítség

- Kiszervezni a felhasználókat egy PaaS-ból
  - Migrálni egy másik (IaaS) szolgáltató hálózatába
  - „Véletlenül” az ajánlattevő biztosítja a PaaS-t ebben az új felállásban
- https://www.skyliner.io/offer/heroku



Virtualizáció úttörője





TELCO GRADE PAAS

Problems with the Paas

No telecom-ready PaaS

PaaS is coming from the IT (web) world

- Általában HTTP load balancer van csak,
  - egyéb protokollok (SIP, diameter, TCP session, stb.) jellemzően nem támogatottak.
- Belső állapotinformációk kiszervezése külső DB/cache
  - teljesítmény problémák
- Semmilyen QoS / válaszidő garancia nincs
- PaaS teljes mértékben elrejti a virtuális gépeket és hálózatokat
  - nem lehet közös VM-re tenni egymással sokat kommunikáló alkalmazásokat,
  - nem lehetséges hálózati optimalizációt (pl. Intel DPDK) kihasználni
- Nincs szabványos PaaS
  - a PaaS alkalmazásokat minden egyes operátor hálózatra fel kell(ene) készíteni

Requirements for a PaaS

- Actually Saas, PaaS, microservices…
- http://12factor.net/

Req.s 1/2

I. Codebase

One codebase tracked in revision control, many deploys

II. Dependencies

Explicitly declare and isolate dependencies

III. Config

Store config in the environment

IV. Backing Services

Treat backing services as attached resources

V. Build, release, run

Strictly separate build and run stages

VI. Processes

Execute the app as one or more stateless processes

Req.s 2/2

VII. Port binding

Export services via port binding

VIII. Concurrency

Scale out via the process model

IX. Disposability

Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity

Keep development, staging, and production as similar as possible

XI. Logs

Treat logs as event streams

XII. Admin processes

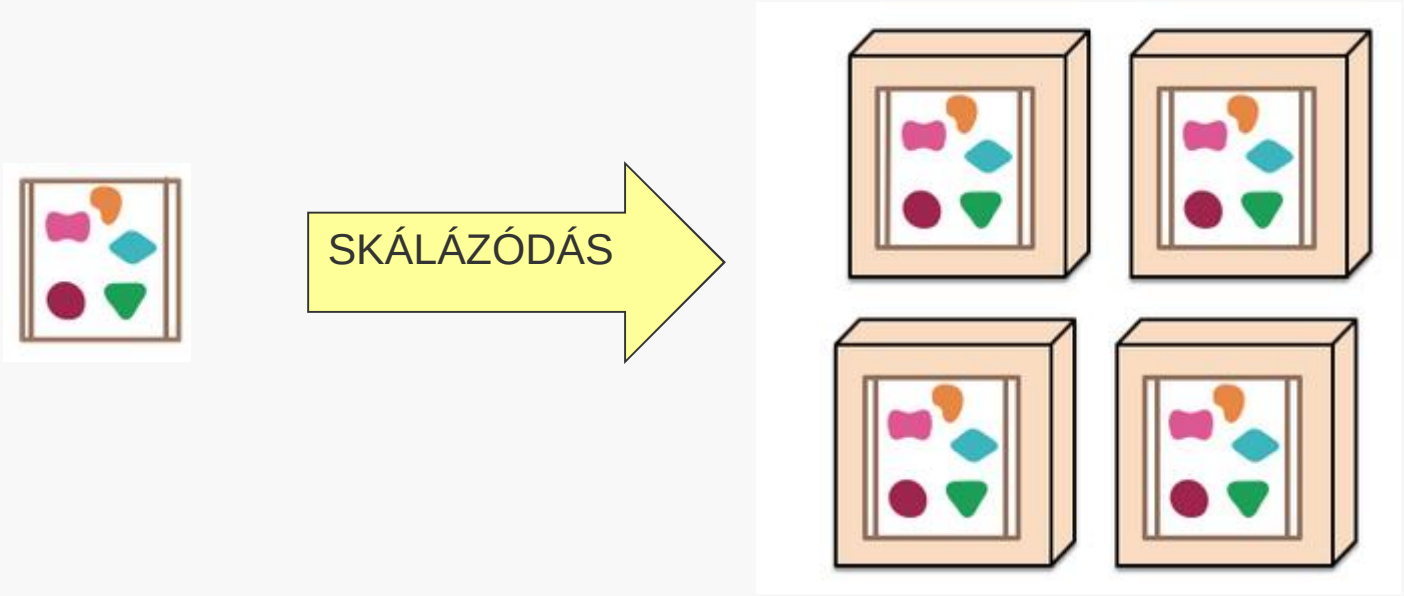
Run admin/management tasks as one-off processes

Microservices architecture

- http://martinfowler.com/articles/microservices.html

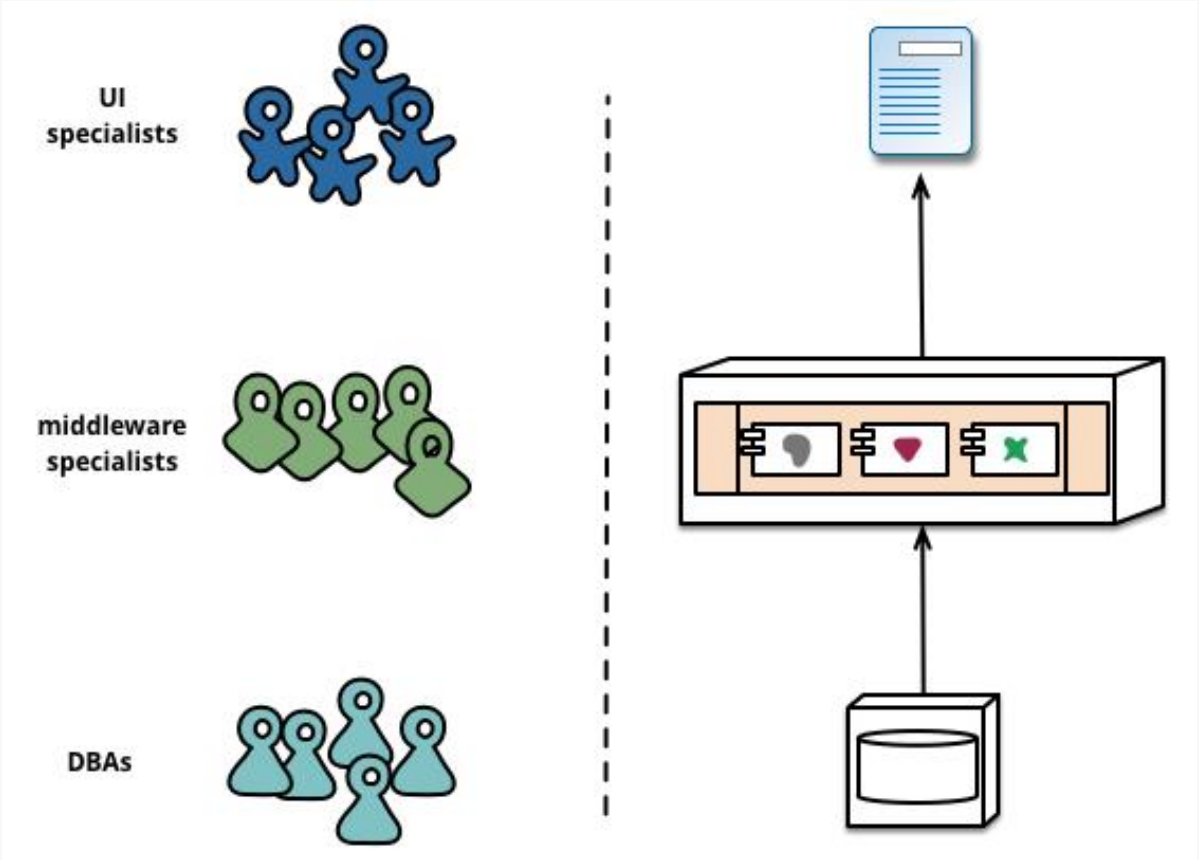
Monolitikus alkalmazások

- Minden funkcionalitás egy nagy program része
- Skálázódás során mindent többszörözni kell



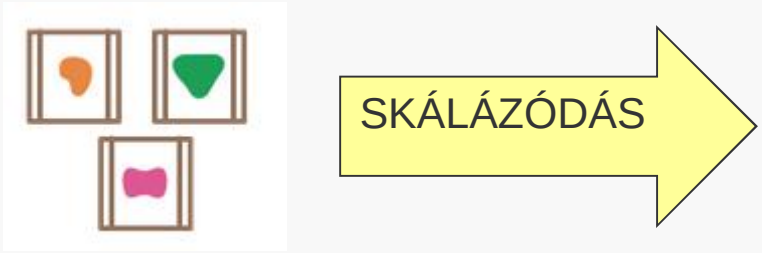
Monolitikus alkalmazások „átka”

» Egy nagy alkalmazás, „szakértők” elkülönülten saját részüket írják meg, majd összefűzik egy alkalmazássá



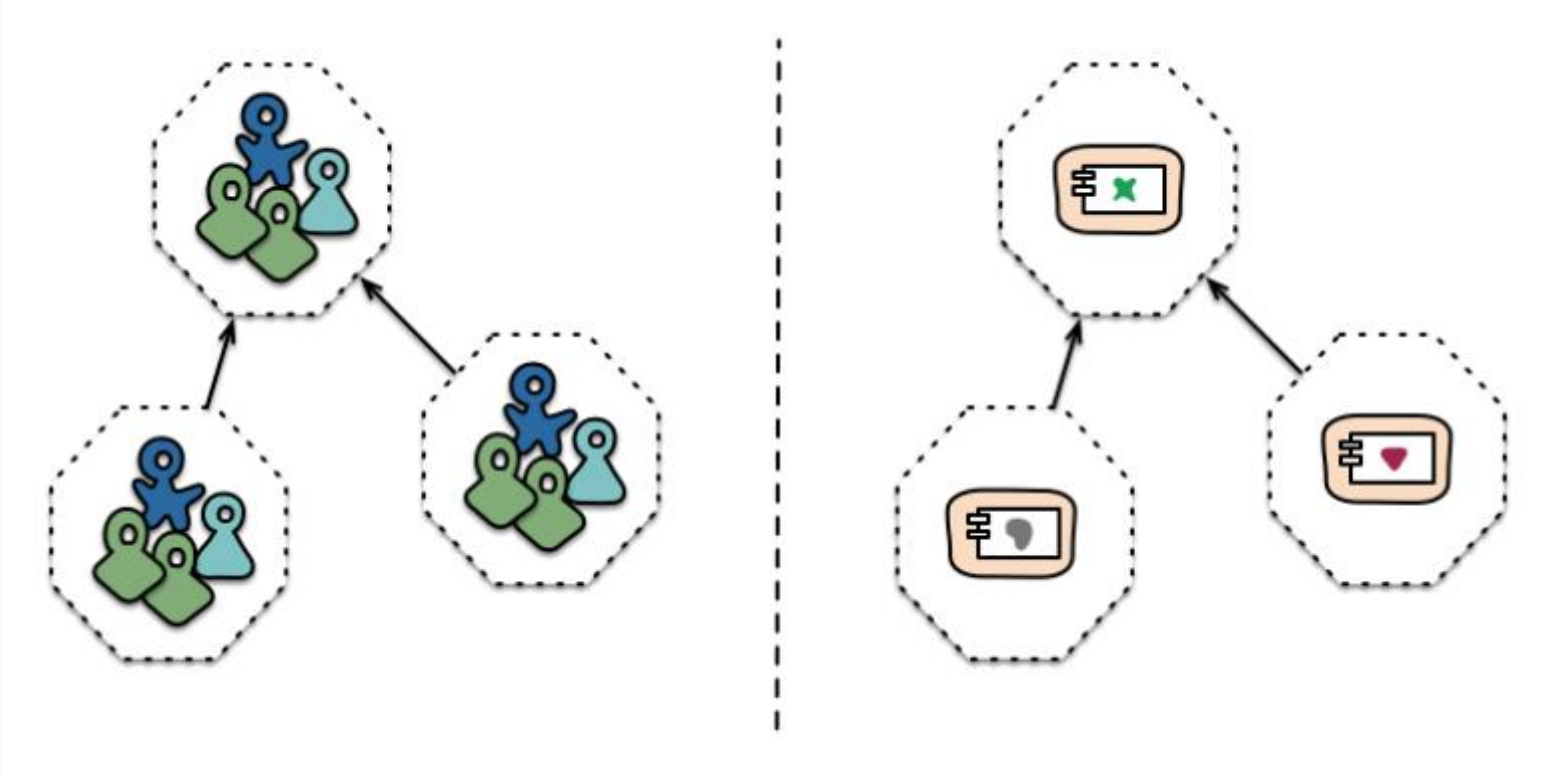
Mikroszolgáltatások

- Funkcionalitásonként egy-egy külön program
  - Szabványos interfész (API) a programok közt
  - Telepítési (együttműködési) logika
- Skálázódás során csak a terhelt alkalmazást kell többszörözni



Mikroszolgáltatás alapú fejlesztések

- „Crossfunctional teams”
- Könnyebb szétosztani, kiszervezni de nem elég egy fedél (cég, projekt) alá gyűjteni a fejlesztőket



Mikroszolgáltatás alapú architektúra

- Az új architekturális paradigma sok mindenre kihat
- Pl. Monolitikus vs. Mikroszolgáltatás alapú tárolás