IOT KERETRENDSZEREK ÉS IPARI ALKALMAZÁSAIK

SmartComLab

IoT keretrendszerek felépítése I. : M2M, M2C protokollok, feldolgozási módszerek

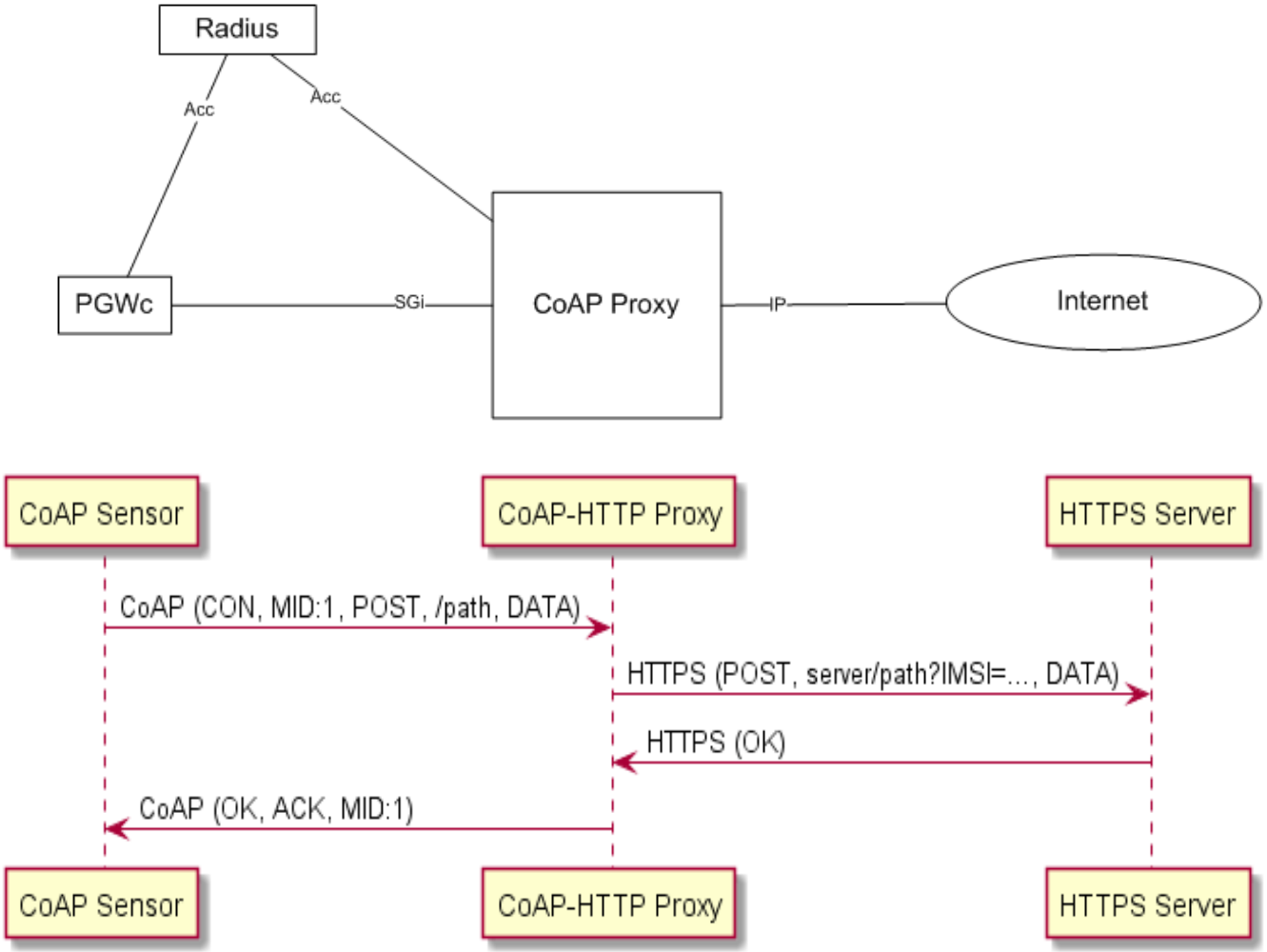
Edge bróker

- Azon platformbeli modul, amely tartja a kapcsolatot az eszközökkel valamilyen (ált. perzisztens) módon.
- Általában egy magasan skálázható message queue megoldás, több ezer eszközzel tartja a kapcsolatot.
  - Load balancing, broker clustering, topic queue-ing, resiliency, inter-broker routing,
- Nem tárol adatot, csak továbbít a többi modul felé.
- Főként M2C, de gyakran C2M kommunikációt is enged.
- Számlázás: forgalom alapján
- Példák
  - Azure: IoT Hub
  - Nyílt forráskód: RabbitMQ, Mosquitto

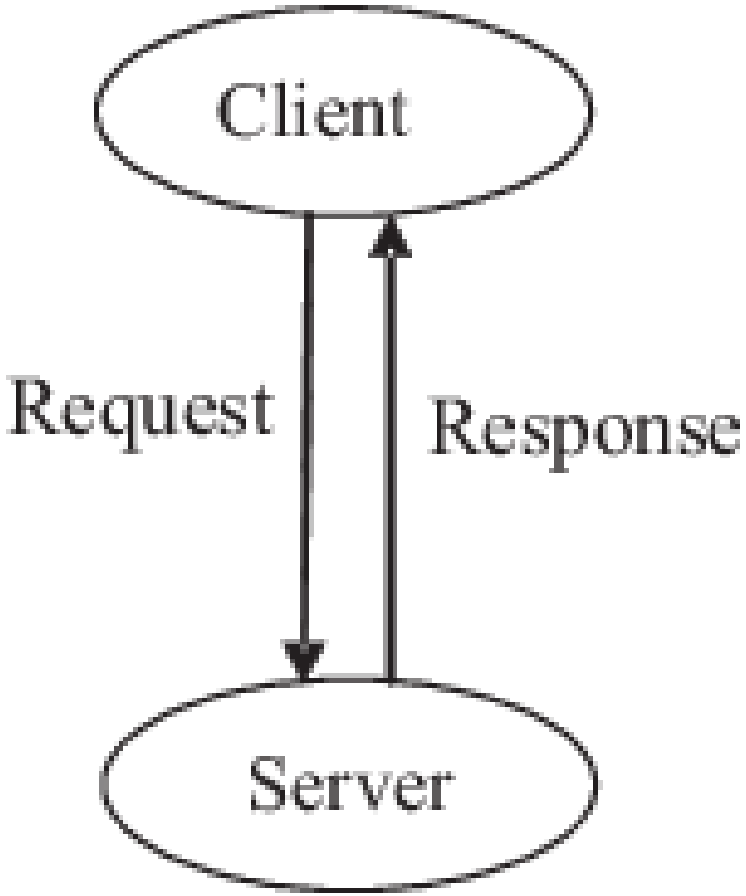
Edge – Cloud kommunikáció

- Protokollok:
  - Kliens szerver: CoAP, HTTP, Websocket
  - Publish-subscribe: MQTT, AMQP
- Fő tulajdonságai:
  - Állapotmentes kommunikáció (nincs nagyon protokoll szintű session kiépítés)
  - NAT traversal: az eszközök NAT, tűzfal mögött lehetnek
  - Valamilyen módon perzisztens kapcsolat: a kommunikációs vonal (főleg TCP kapcsolat) fenn van tartva folyamatosan, mert a connection handshake-elése erőforrás igényes ha gyakran kellene csinálni
  - Kétirányú kapcsolat: M2C, C2M is megengedett (push notification például)

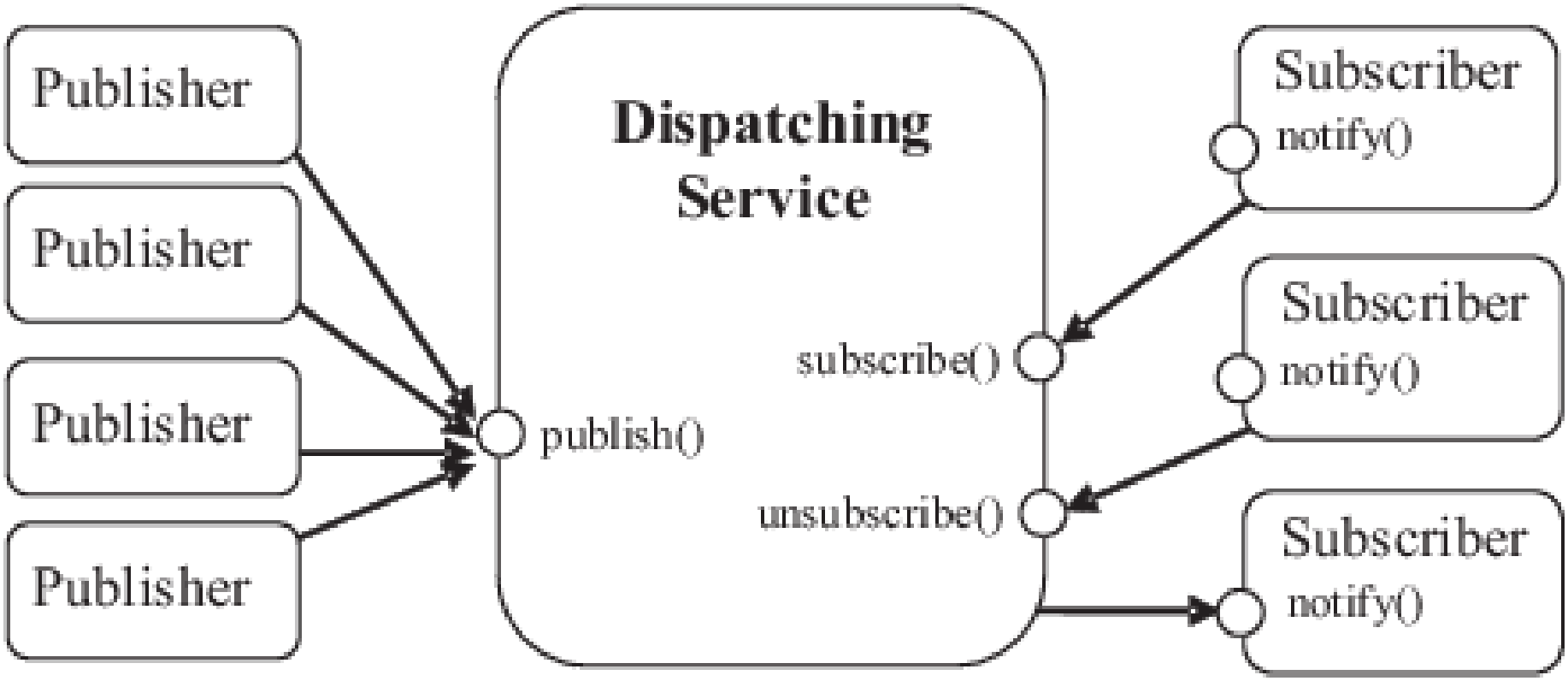
CoAP proxy – egy élő példa



Kliens-szerver vs PubSub



Kliens-szerver vs PubSub

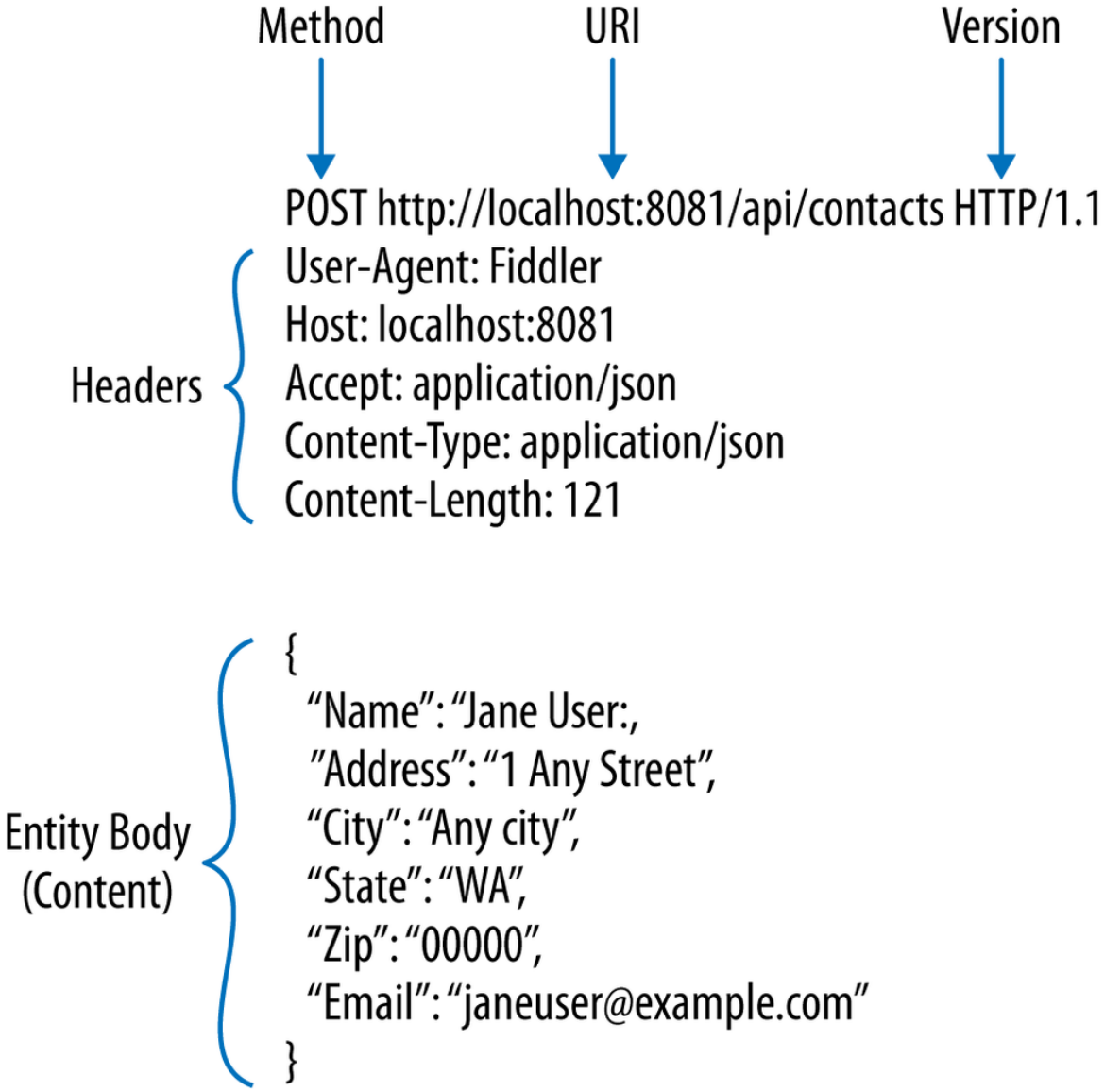


- További információért:
- VITMAV22 IOT RENDSZEREK KOMMUNIKÁCIÓS MEGOLDÁSAI
- http://www.tmit.bme.hu/vitmav22

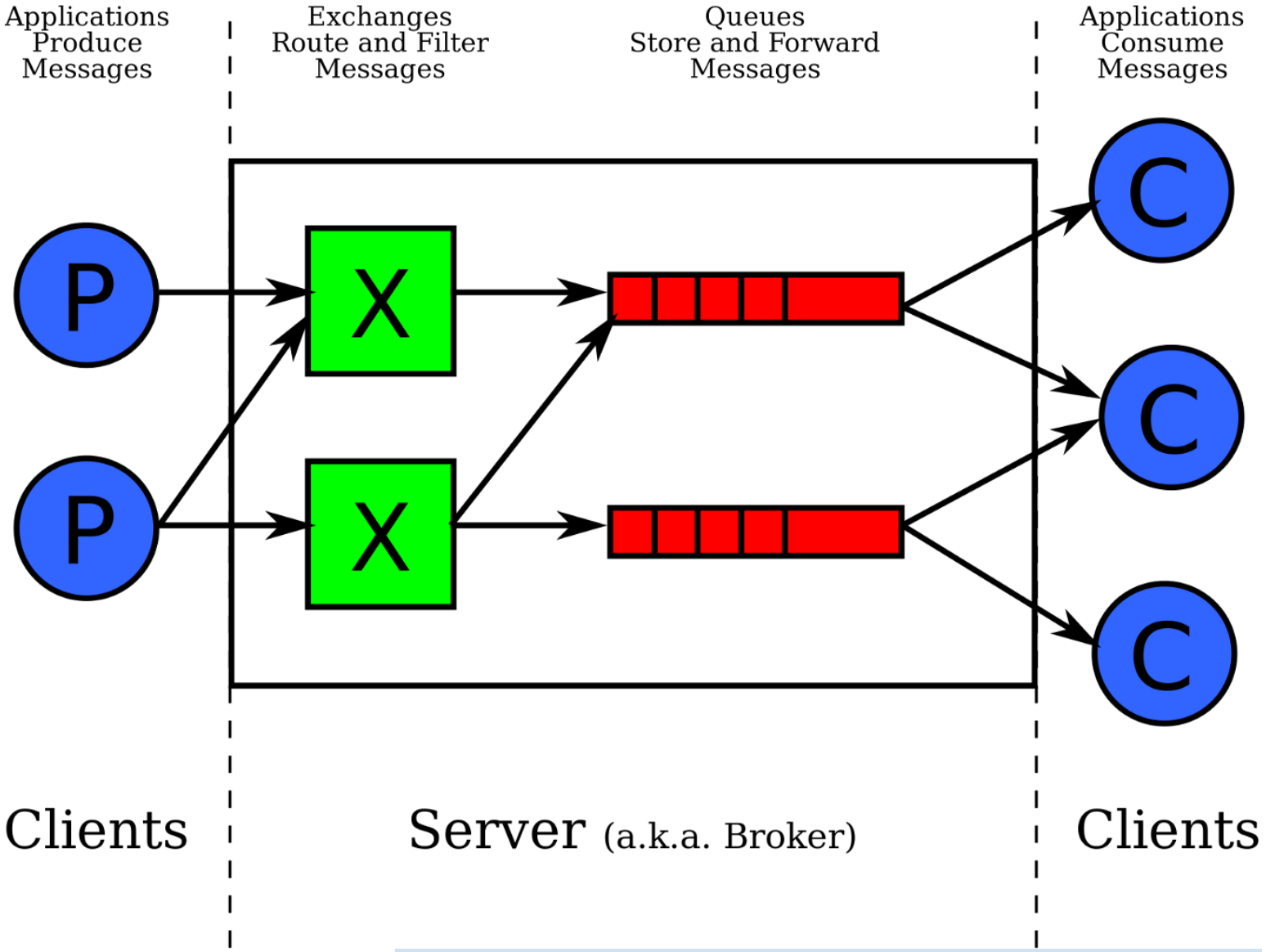
RESTful Web Services

- REST: Representational State Transfer
  - Gondolkodási mód kliens-szerver alapú kommunikációról, adatmodellről (erőforrás reprezentáció), szerver felépítésről
  - Több protokoll megvalósítja: HTTP, CoAP, …
- Tervezési elvek
  - Kliens-szerver architektúra: minél vékonyabb kliens
  - Állapotmentesség: a kérés mindig tartalmazzon minden szükséges információt a válaszadáshoz, az URI egyértelműen azonosítsa az erőforrást és az állapotát (a szerver maga lehet állapottartó!)
  - Gyorsítótárazhatóság, réteges felépítés
  - Igényelt kód: programrészeket tölthet le a kliens (Javascript szkriptek)
- Web Service interfészek
  - Erőforrás azonosítás URI-kkal
  - Erőforrások elérése és manipulálása is URI alapú
  - Önleíró üzenet
  - HATEOAS: hipermedia, mint az alkalmazásállapot motorja

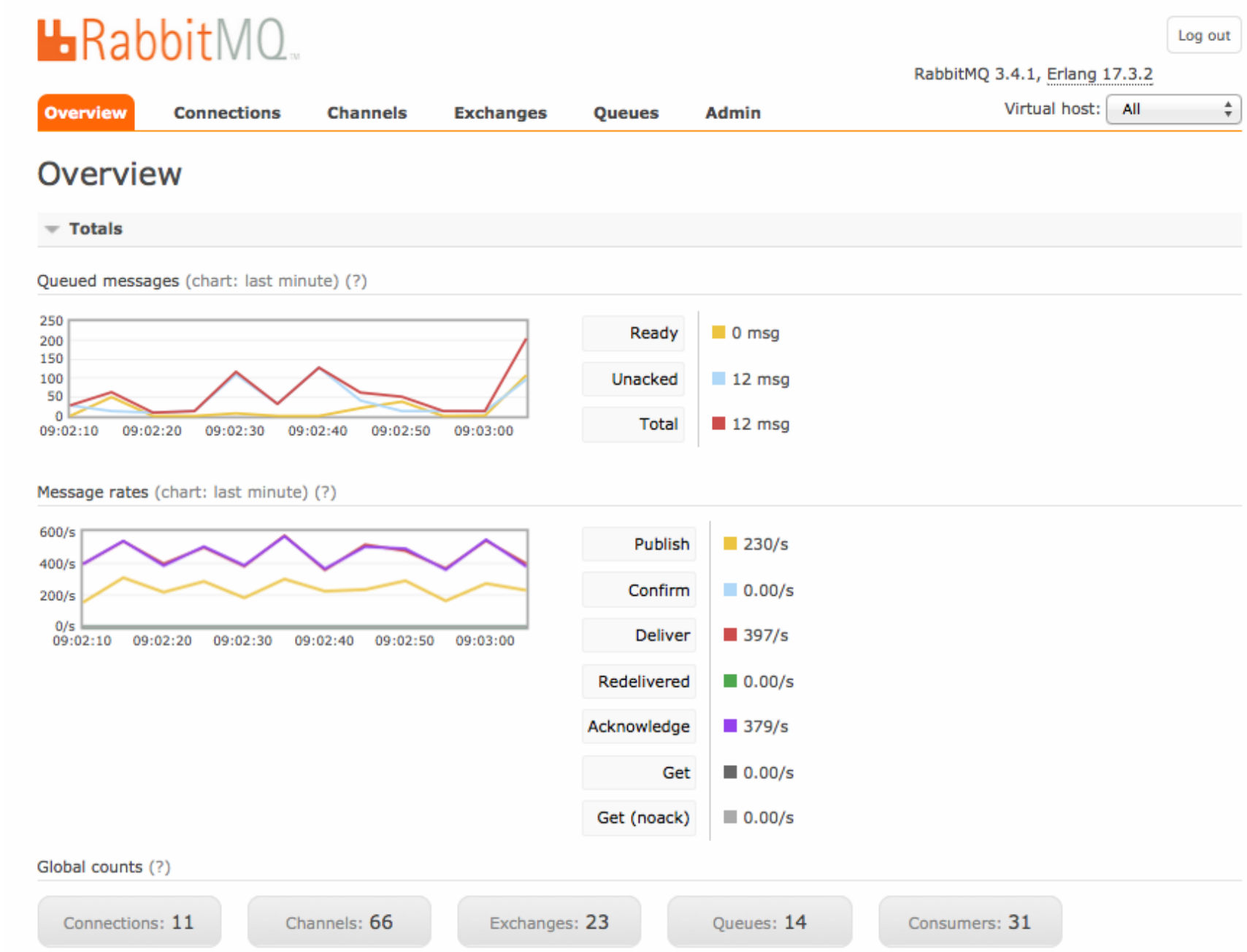
A HTTP protokoll üzenetformátuma



Message queue protokollok



Message queue



Message queue

RabbitMQ

User: guest

Overview | Connections | **Channels** | Exchanges | Queues | Users | Virtual Hosts

Channels

	Details					Message rates		
Channel	**User name**	**Transactional**	**Prefetch**	**Messages unacked**	**Status**	**publish**	**deliver / get**	**ack**
127.0.0.1:51717:1	guest	○	0	8	Active		5070/s	5072/s
127.0.0.1:51718:1	guest	○	0	12	Active		4046/s	4046/s
127.0.0.1:51719:1	guest	○	0	15	Active		3816/s	3822/s
127.0.0.1:51720:1	guest	○	0	7	Active		5877/s	5895/s
127.0.0.1:51721:1	guest	○	0	14	Active		3905/s	3907/s
127.0.0.1:51722:1	guest	○	0	0	Active	7092/s		

Last update: Thu Dec 02 2010 17:13:15 GMT+0000 (GMT)

Update every 5 seconds ▼

Speciális IIoT edge protokoll: OPC UA

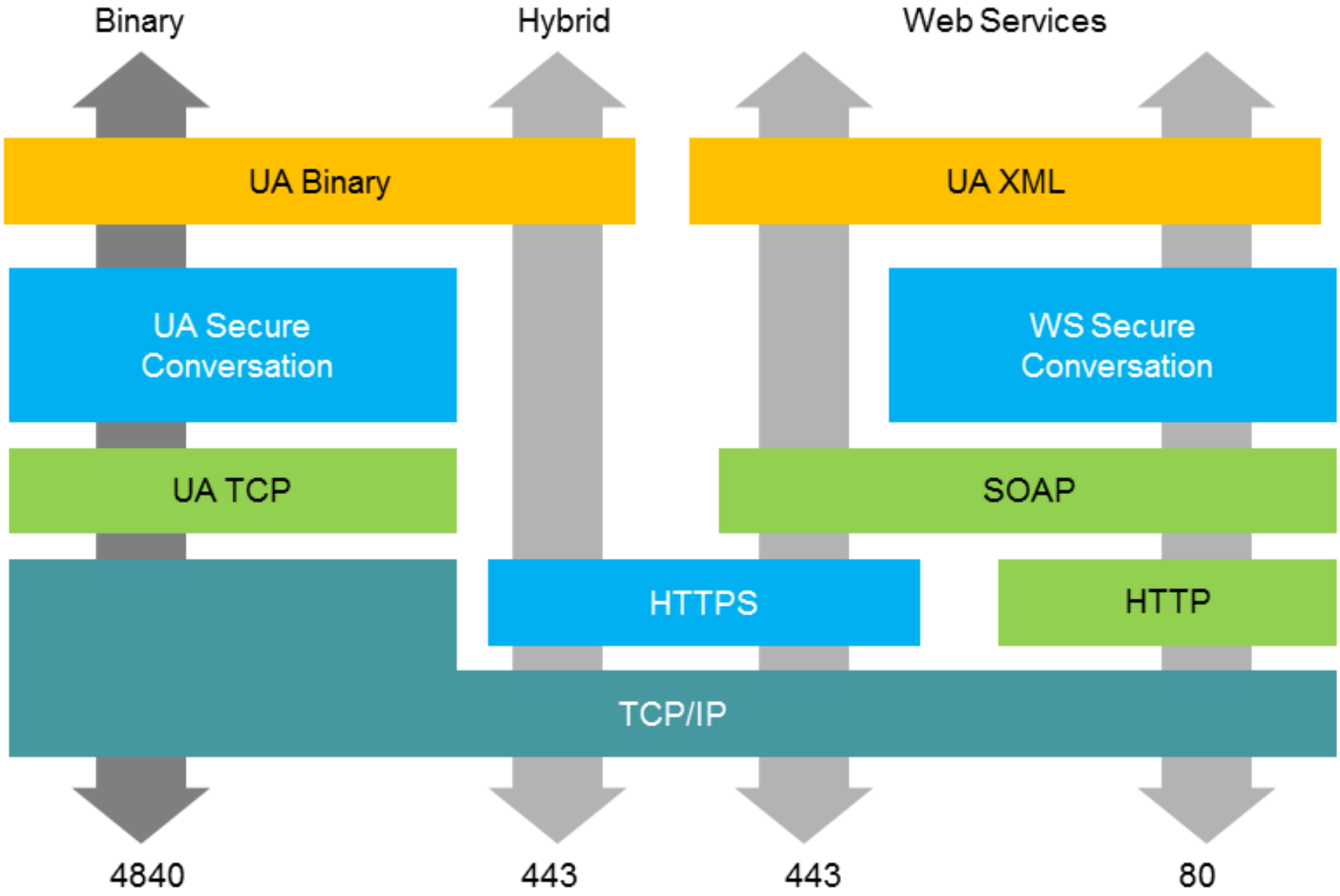
- OPC bináris protokoll és Web Service alapú
- Komplex standard: a kommunikáció minden aspektusát szabályozza (adatmodell a Microsoft COM-ból jön)
- Túl sok funkciót próbál támogatni: PubSub, Remote Procedure Call, WebService stack, SOA és discovery, …
- Server heavy architektúra
- Heterogén implementációk: nem működnek együtt

Speciális IIoT edge protokoll: OPC UA

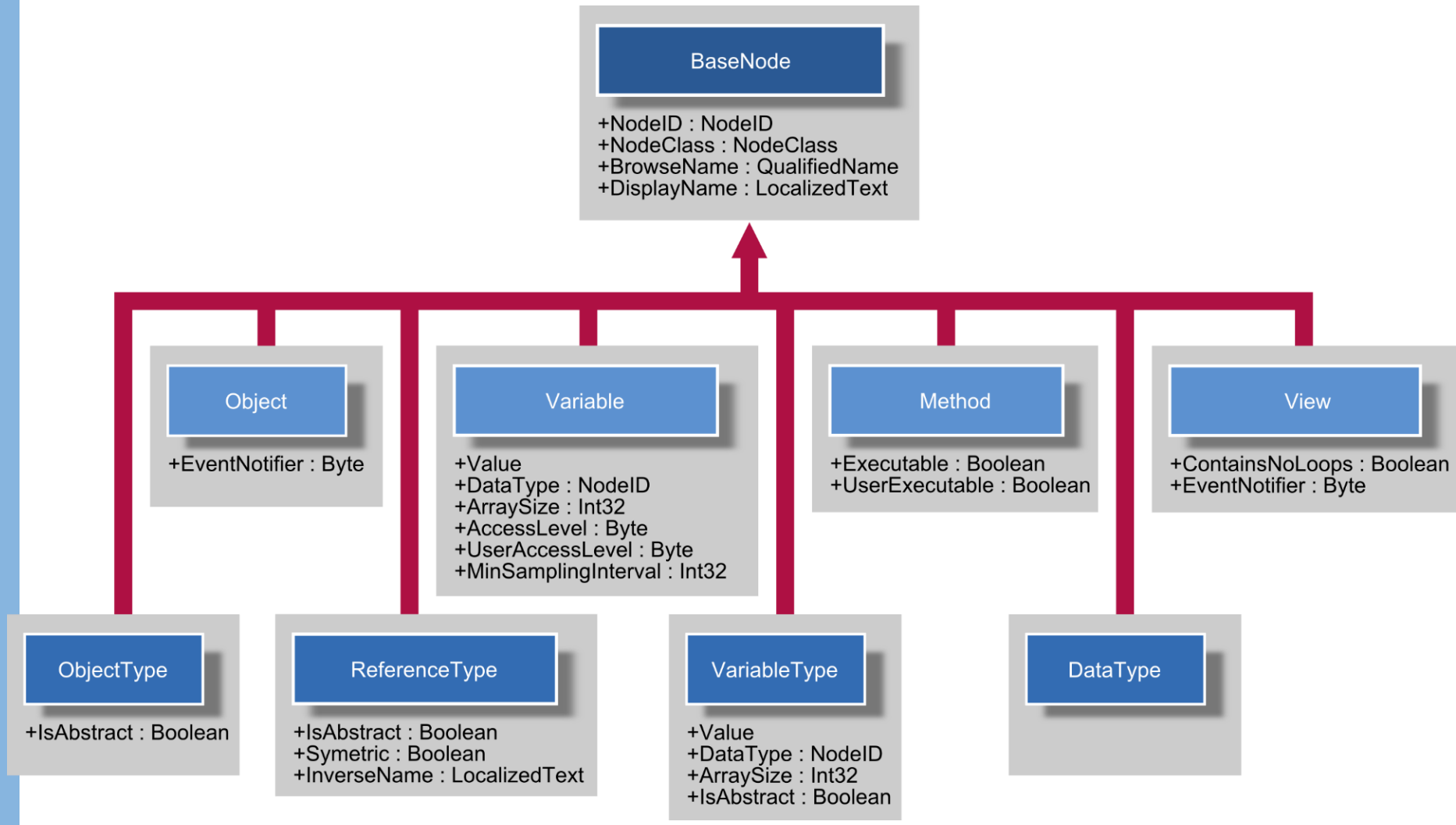
IEC 62541 Overview

ID	release date	title
IEC/TR 62541-1	02/2010	OPC Unified Architecture - Part 1: Overview and Concepts
IEC/TR 62541-2	02/2010	OPC Unified Architecture - Part 2: Security Model
IEC 62541-3	07/2010	OPC Unified Architecture - Part 3: Address Space Model
IEC 62541-4	10/2011	OPC Unified Architecture - Part 4: Services
IEC 62541-5	10/2011	OPC Unified Architecture - Part 5: Information Model
IEC 62541-6	10/2011	OPC Unified Architecture - Part 6: Mappings
IEC 62541-7	07/2012	OPC Unified Architecture - Part 7: Profiles
IEC 62541-8	10/2011	OPC Unified Architecture - Part 8: Data Access
IEC 62541-9	07/2012	OPC Unified Architecture - Part 9: Alarms and Conditions
IEC 62541-10	07/2012	OPC Unified Architecture - Part 10: Programs

OPC UA stack


Binary
Hybrid
Web Services
UA Binary
UA XML
UA Secure Conversation
WS Secure Conversation
UA TCP
SOAP
HTTPS
HTTP
TCP/IP
4840
443
443
80

OPC UA object model (MS COM)



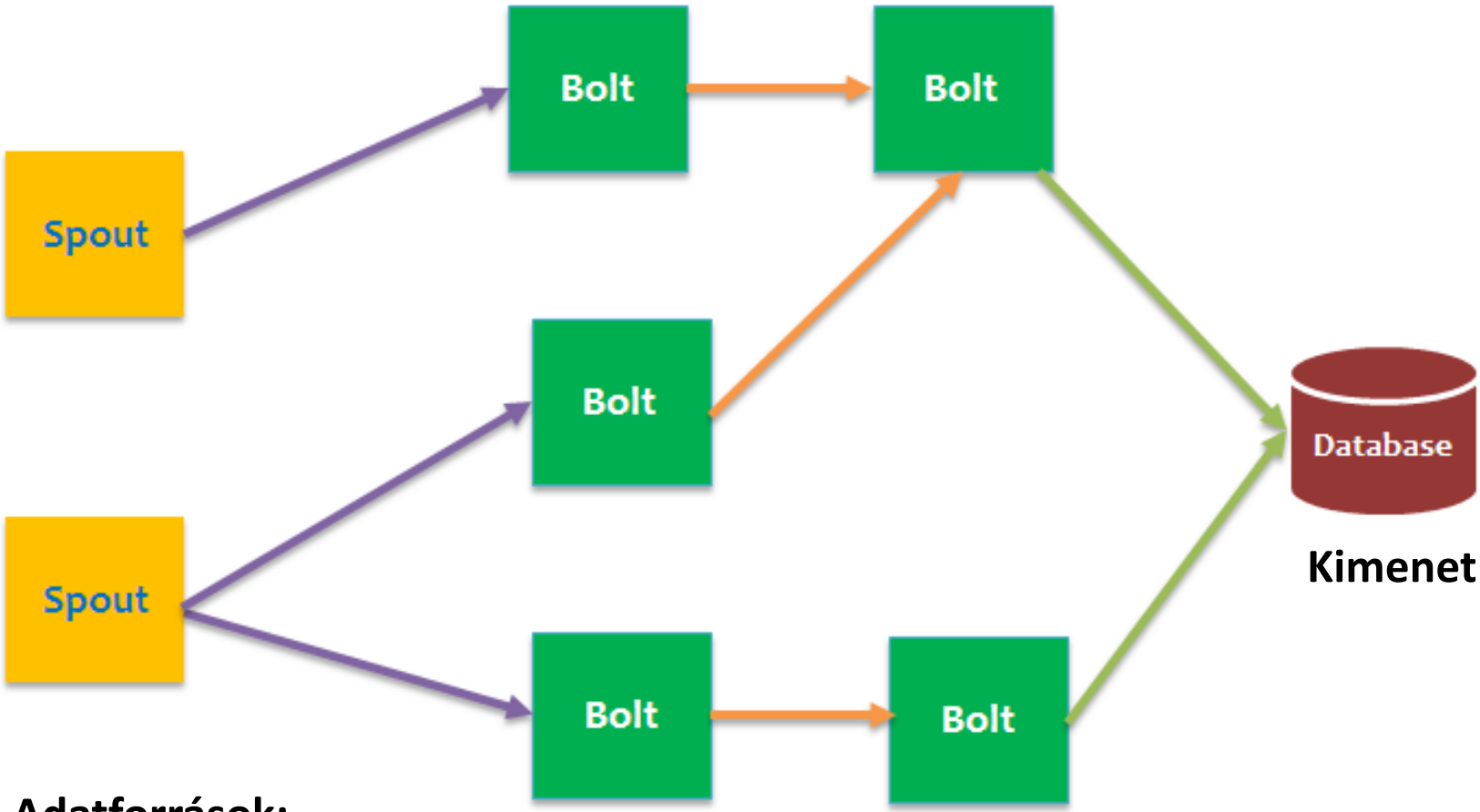
Stream Processzor

- Az edge brókertől átveszi a beérkező adatokat „on the fly”.
- A leírt szkript minden egyes bejövő touple-ra (adatelem) elvégződik.
- Egyszerű műveleteket tud végezni:
  - Ablakozás, ablakozott aggregációs műveletek
  - Számlálók, összegzések
  - Mintafelismerés betanított modell alapján

Stream Processzor

- Nagy hangsúly:
  - Skálázódás: egyidejű párhuzamos forgalom kezelése
  - Hiba tolerancia és zéró adatvesztés, helyreállás
  - Minél kevesebb állapotot (contextus) tart fenn a rendszer
- Példák
  - Apache Storm
  - Apache Spark Streaming
  - Azure Stream Analytics

Példa egy stream pipeline-ra



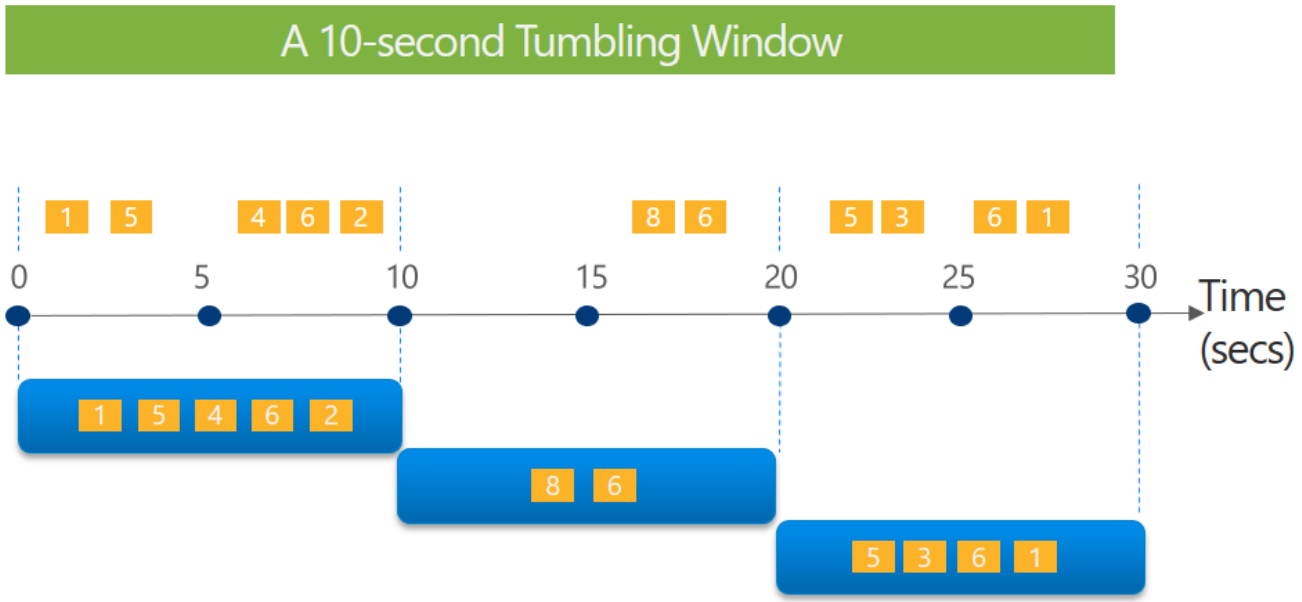
Kimenet

Adatforrások: periodikus „touple” források

Feldolgozási lépések irányított, hurokmentes gráfban szétosztva a hálózatban

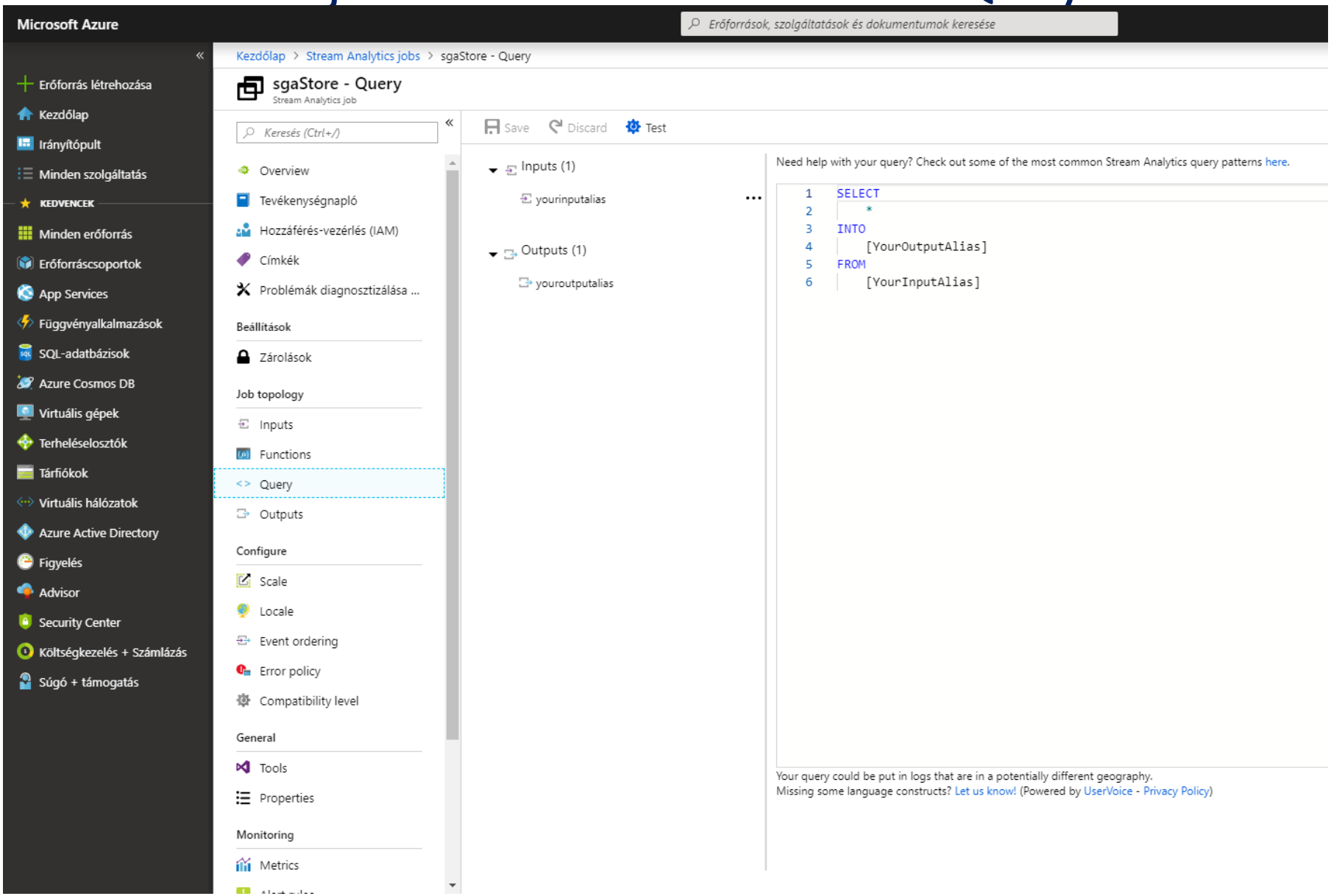
Példa egy stream processing-ra

Tell me the count of tweets per time zone every 10 seconds




```
SELECT TimeZone, COUNT(*) AS Count
FROM TwitterStream TIMESTAMP BY CreatedAt
GROUP BY TimeZone, TumblingWindow(second,10)
```

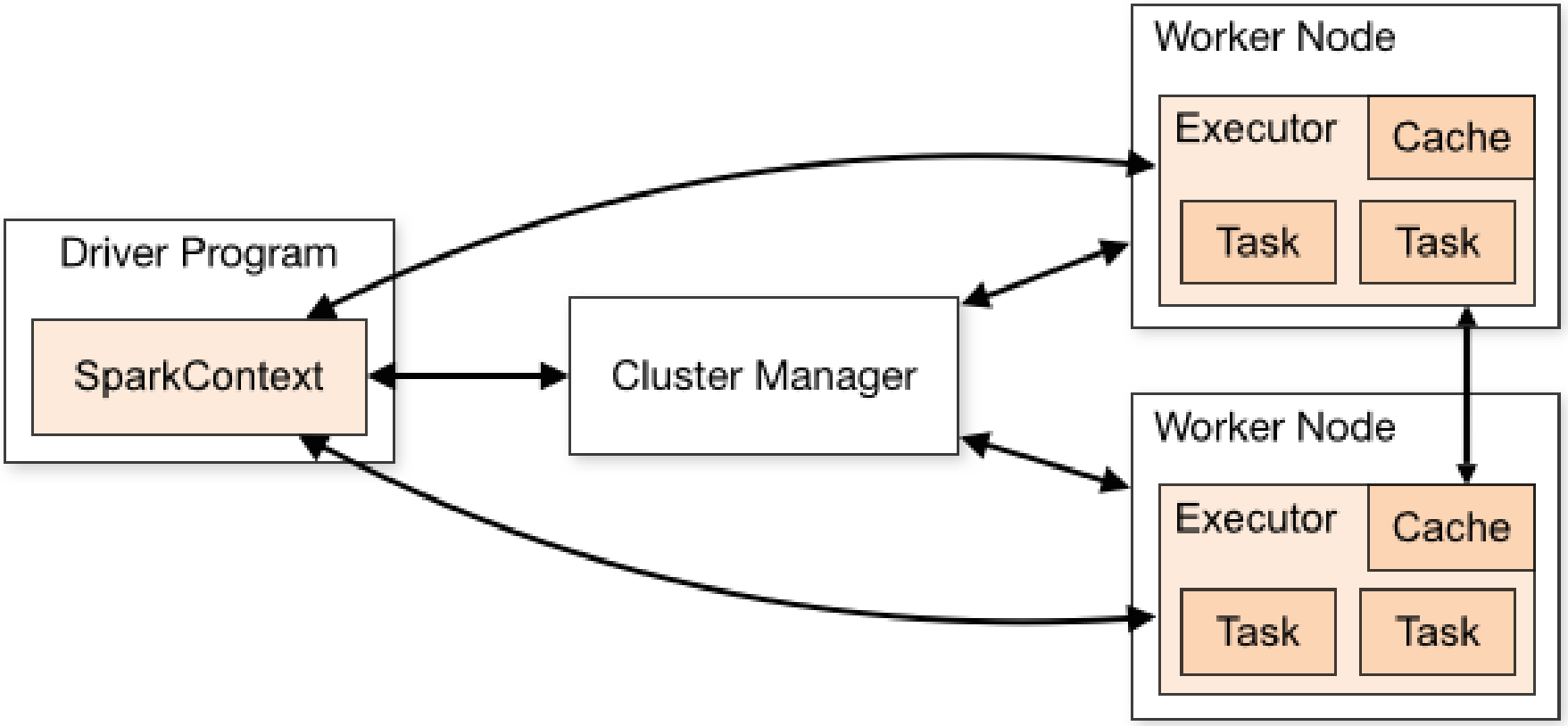
Azure SA job szerkesztő felülete: SQL syntax



Batch Processzor

- Aszinkron, kötegfeldolgozáson alapuló analitika
- Adatbázisból és egyéb perzisztens tárhelyekből dolgozik
- Promise jellegű programozás általában
- Cluster-en futó elosztott alkalmazás
  - vezérlő és
  - worker node-ok, amik darabszáma változhat a terhelés függvényében
- Felhasználási példák:
  - Machine learning modellezés és training
  - Adatbányászkodás, pl. szöveg extraction

Batch Processzor



Batch Processzorra látott példák

- MapReduce
  - Apache projekt volt Hadoop alapokon
- Apache Spark
  - State of the art, open source
  - Igen fejlett és szerteágazó adatfeldolgozó API
- Azure Functions
  - Microsoft Azure szoftverstack része
- R server + scheduled scripting
  - Hackathlonra megfelelő

Machine learning in Apache Spark

```
// Every record of this DataFrame contains the label and
// features represented by a vector.
StructType schema = new StructType(new StructField[]{
  new StructField("label", DataTypes.DoubleType, false, Metadata.empty()),
  new StructField("features", new VectorUDT(), false, Metadata.empty()),
});
DataFrame df = jsql.createDataFrame(data, schema);

// Set parameters for the algorithm.
// Here, we limit the number of iterations to 10.
LogisticRegression lr = new LogisticRegression().setMaxIter(10);

// Fit the model to the data.
LogisticRegressionModel model = lr.fit(df);

// Inspect the model: get the feature weights.
Vector weights = model.weights();

// Given a dataset, predict each point's label, and show the results.
model.transform(df).show();
```

Dataframe: immutable

Input: pl SQL

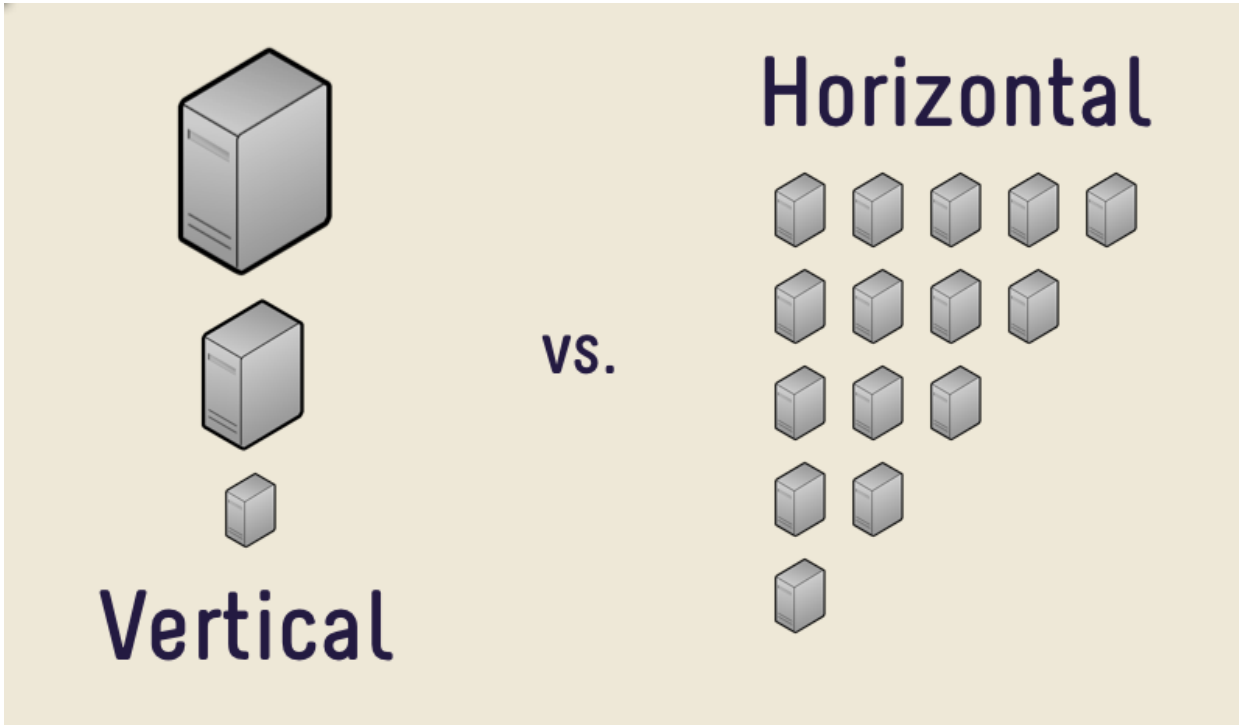
Művelet: logisztikus regressziószámítás

A show függvény fogja érvényre juttatni a pipeline-t.

Belső elosztóhálózat

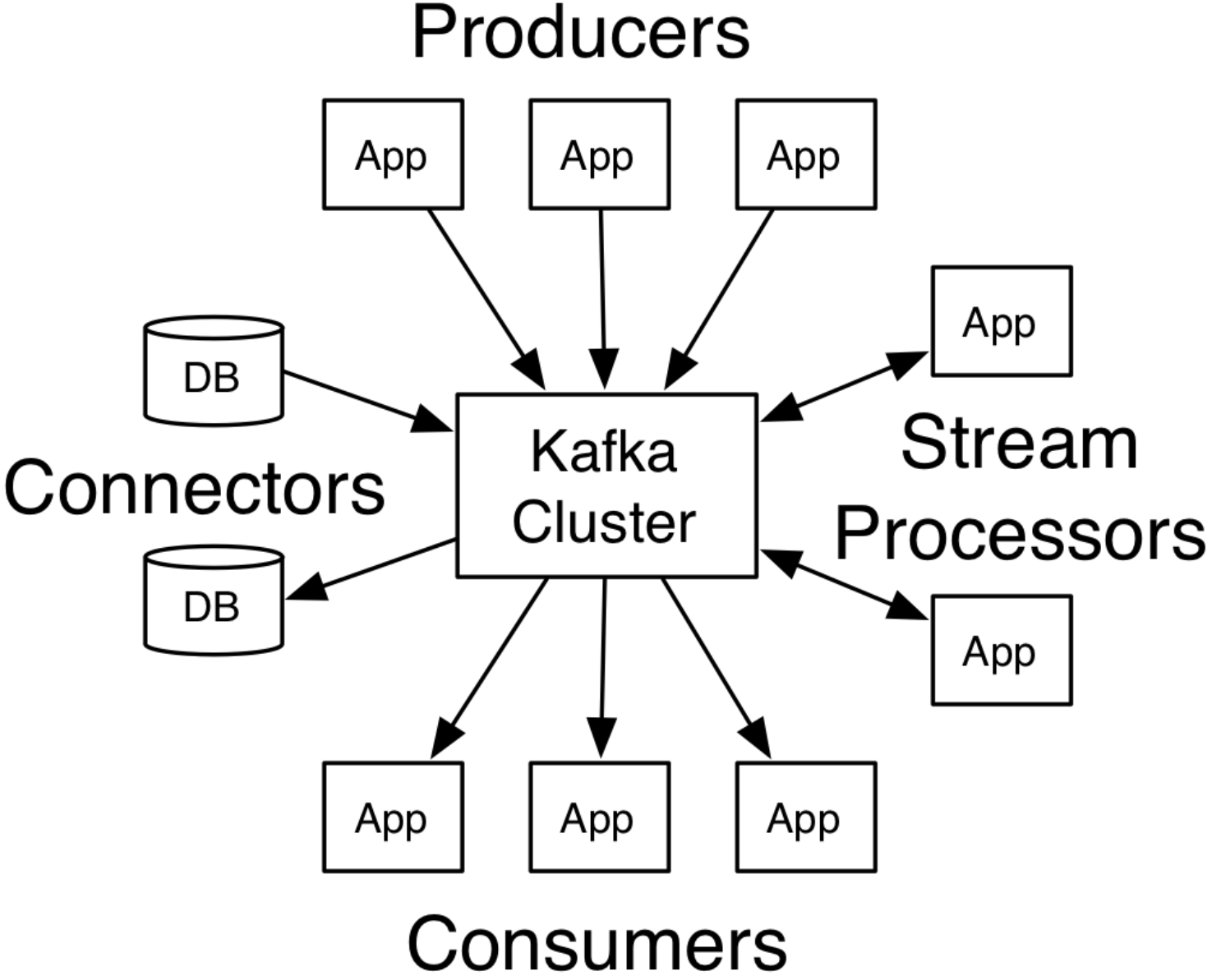
- Hasonló az edge brókerhez: pub-sub megoldás
- Igen magasan skálázható üzenetbusz rendszerek ezek is
- Példák
  - Apache Kafka
  - Azure Event Hub
  - RabbitMQ

Horizontális vs vertikális skálázódás



- Horizontal scaling: scale by adding more machines into your pool of resources
- Vertical scaling means that you scale by adding more power (CPU, RAM) to an existing machine.

Apache Kafka



Kafka has four core APIs

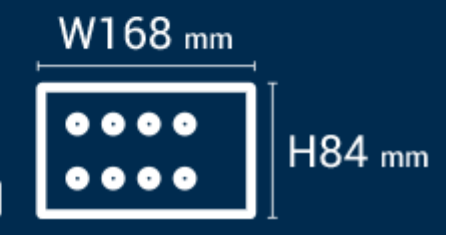
- The Producer API allows an application to publish a stream of records to one or more Kafka topics.
- The Consumer API allows an application to subscribe to one or more topics and process the stream of records produced to them.
- The Streams API allows an application to act as a *stream processor*, consuming an input stream from one or more topics and producing an output stream to one or more output topics, effectively transforming the input streams to output streams.
- The Connector API allows building and running reusable producers or consumers that connect Kafka topics to existing applications or data systems. For example, a connector to a relational database might capture every change to a table.

Egyszerű IoT keretrendszer példa

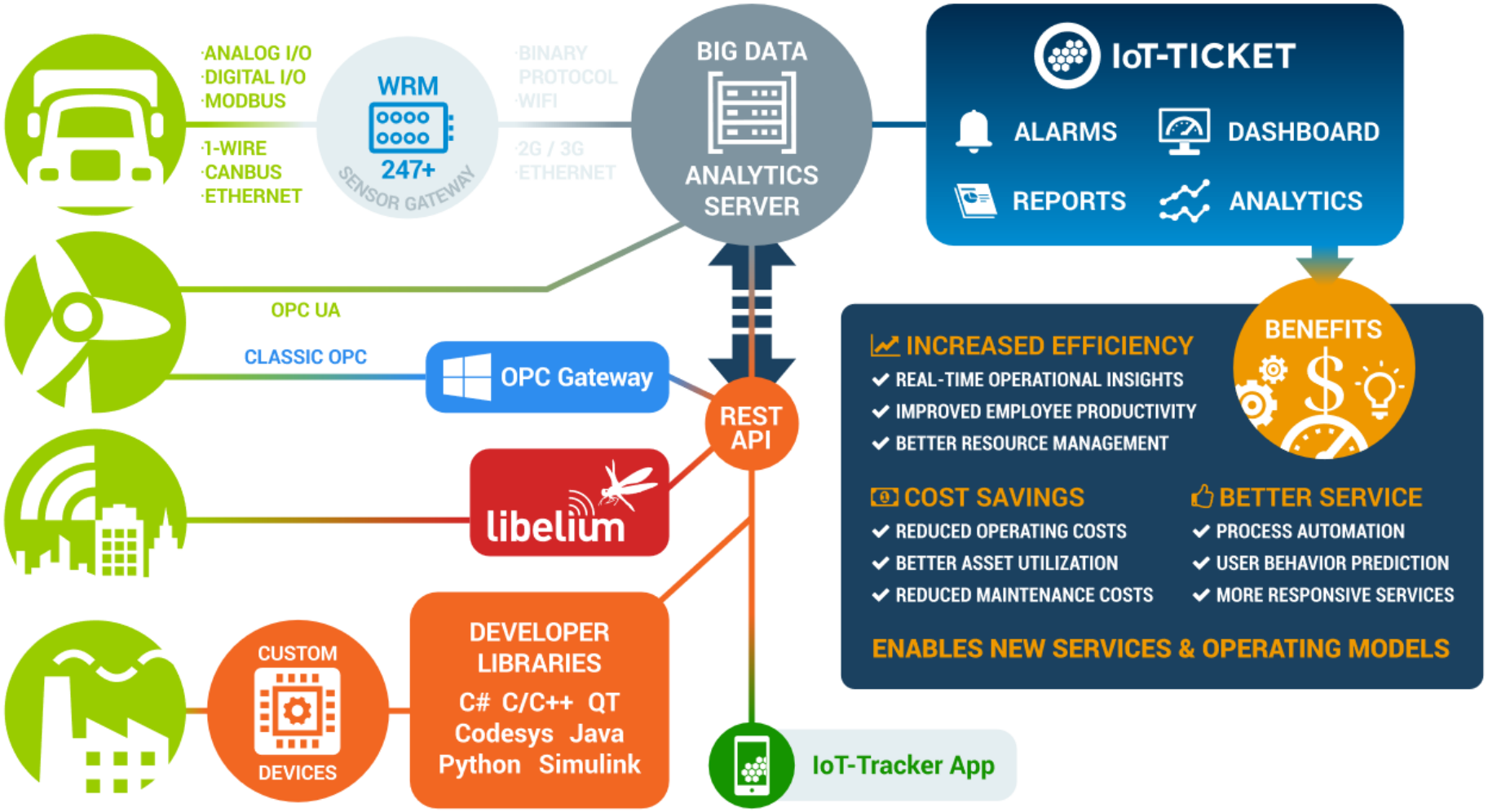
- Finn KKV: Wapice, a termékük az IoT Ticket
- Ehhez tartozó hardver: WRM247 méródoboz
  - Egyedi hardver, nem túl egyedi ötlet
- Ipari partnereknek egyszerűbben kezelhető, nagy ipari berendezésekhez lehet csapni ezt a mérőberendezést

Connector 1	Connector 2	Connector 3	Connector 4
Power Input	Analog input	USB	RS-485
Digital out Supply	Current input	Digital out 2-4	Digital in 2-4
Digital out 1			
Digital in 1			

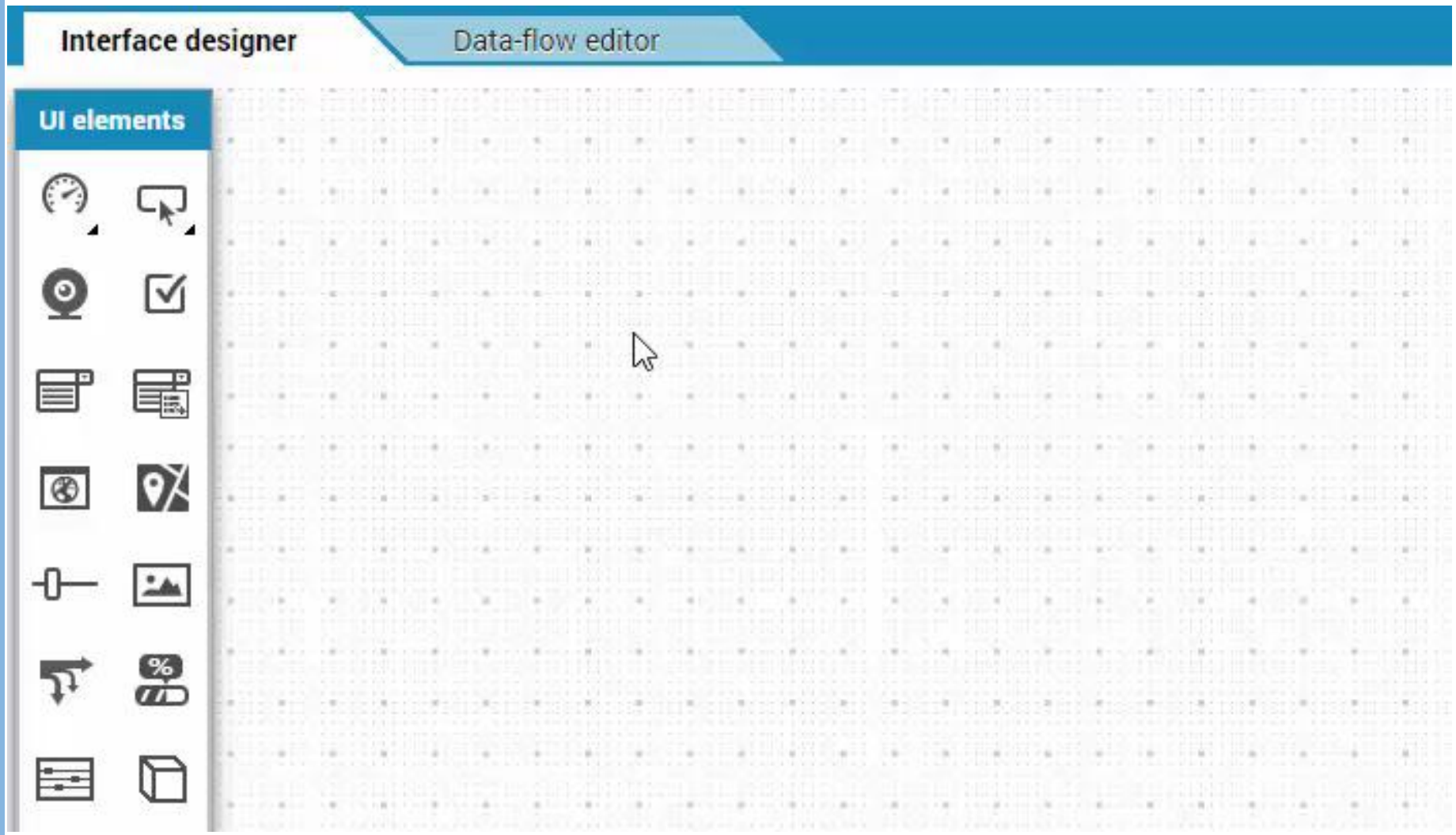
Connector 5	Connector 6	Connector 7	Connector 8
CAN 1	CAN 2	Ethernet 1	Ethernet 2
RS-232	1-Wire		
+5V Out	Debug RS-232		



Egyszerű IoT keretrendszer példa



Wapice Videjó



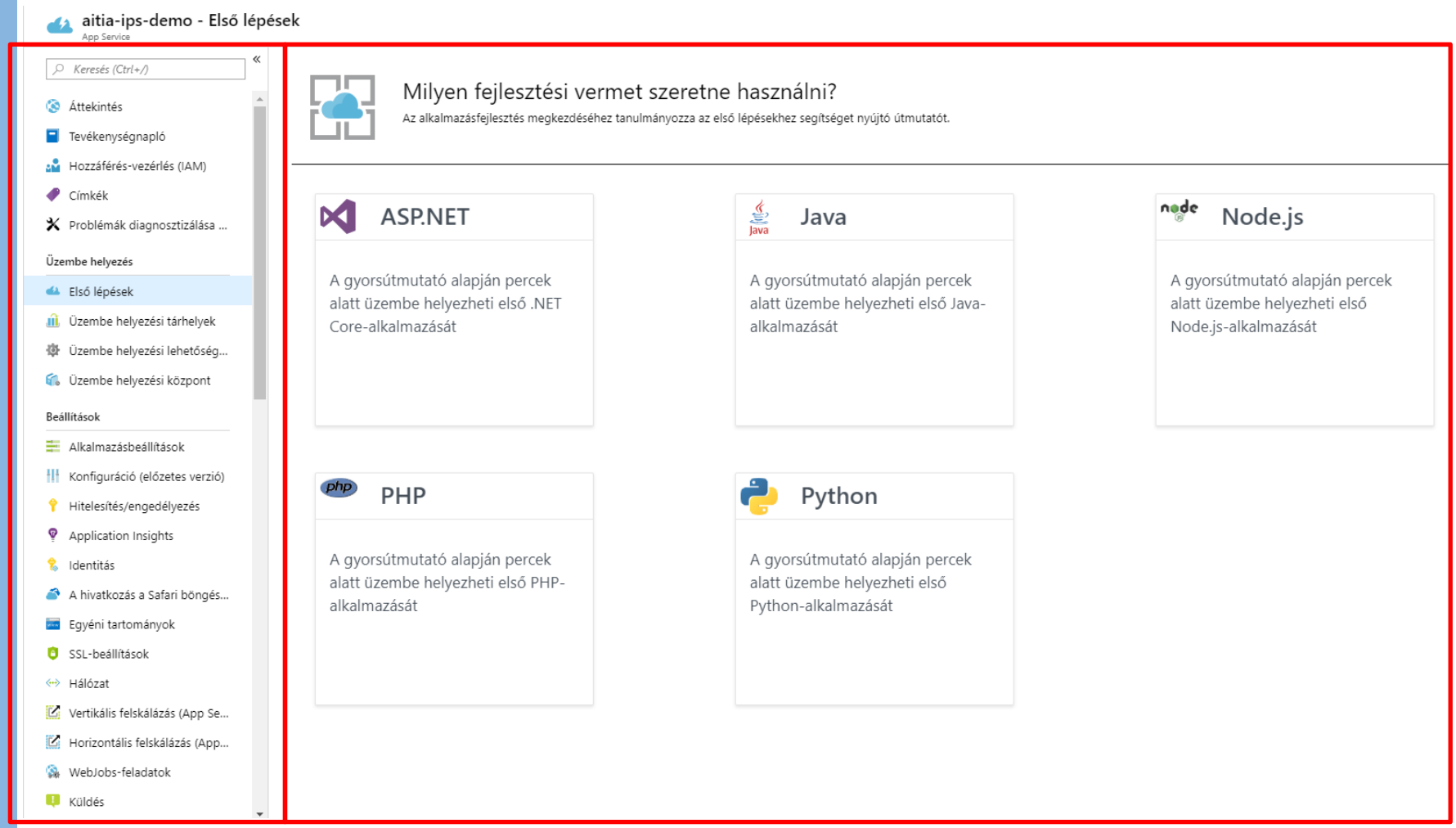
Konténer rendszerek

- Sokféle konténer rendszer elérhető egy felhőben
- Standard, nyílt forráskódú rendszerek:
  - Web Service konténerek
  - Node.JS
  - Docker
- Felhő specifikus konténerek, például:
  - Azure Functions

Web Service konténerek

- Professzionális webszerver technológiák hosztolása
  - RESTful web service interfészek
  - DNS cím biztosítása
- Konténer menedzsment támogatás: újraindítás, skálázás, load balancing automatikusan
- Számlázás használat és erőforrás igények alapján
- Például Azure App Services

Azure App Services példa



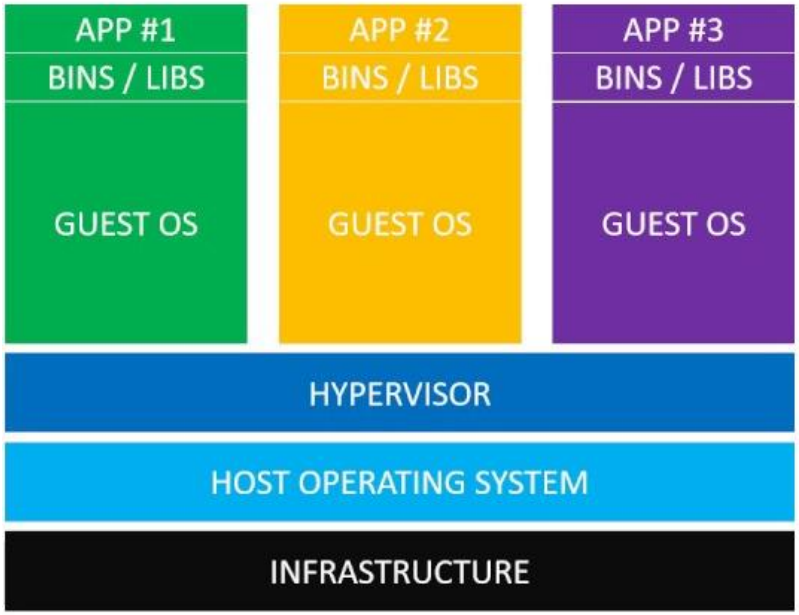
Konténer beállítások

Lehetséges technológiák

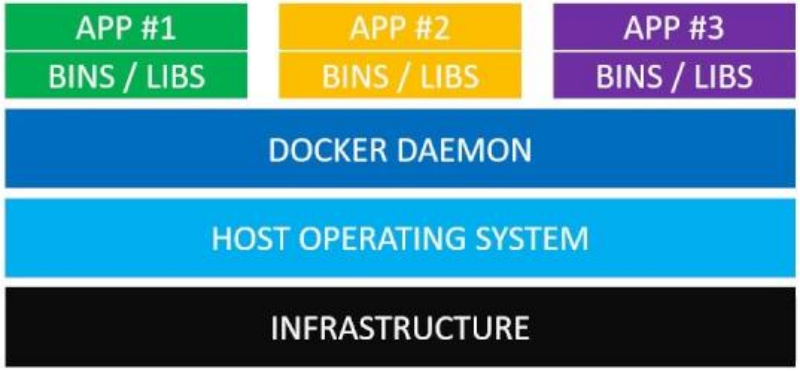
Docker

- Operációs rendszerszintű virtualizáció
- Zárt futtatási környezet ez is
  - Saját, izolált erőforrás keretek: CPU, memory, I/O,network
  - DE: mindenki a hoszt OS kerneljén osztozkodik
- Pro
  - lightweight, fast deployment time, portable, flexible
  - quick scaling
- Kontra
  - Biztonság: root jogú processz kell hozzá globálisan (Docker engine)
  - A konténeren belüli default user is root
  - Szoftveres izoláció: Linux cgroups & más kernel tool-ok
  - Szoftveres izoláció ←→ VM-ek: HyperVisor hardveres izolációja
- Application packaging:
  - mindenre, amire az appnak szüksége van (libs, OS, utils, …)

VM vs konténerek



Virtual Machines



Docker Containers