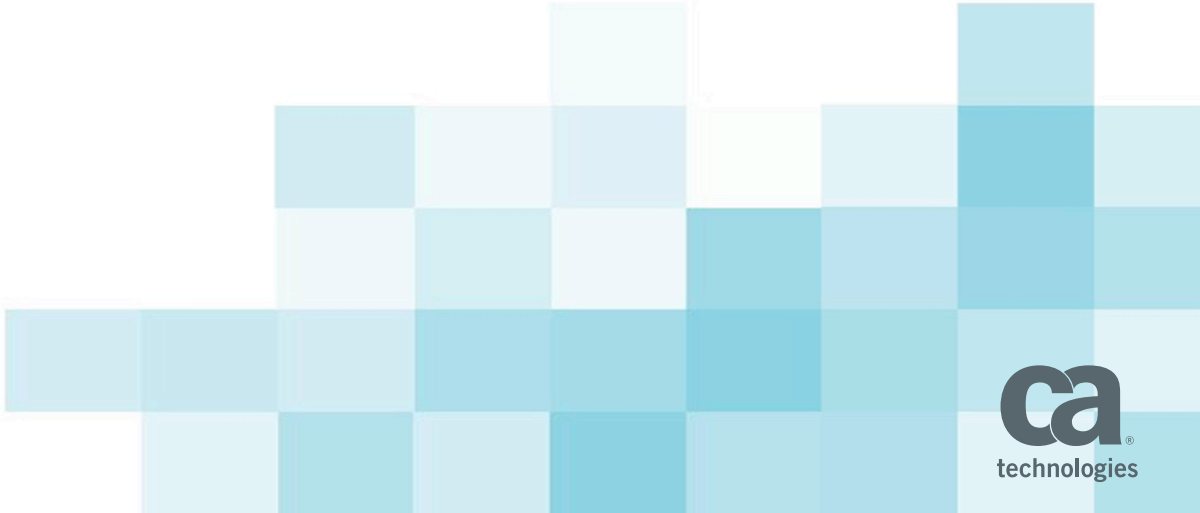
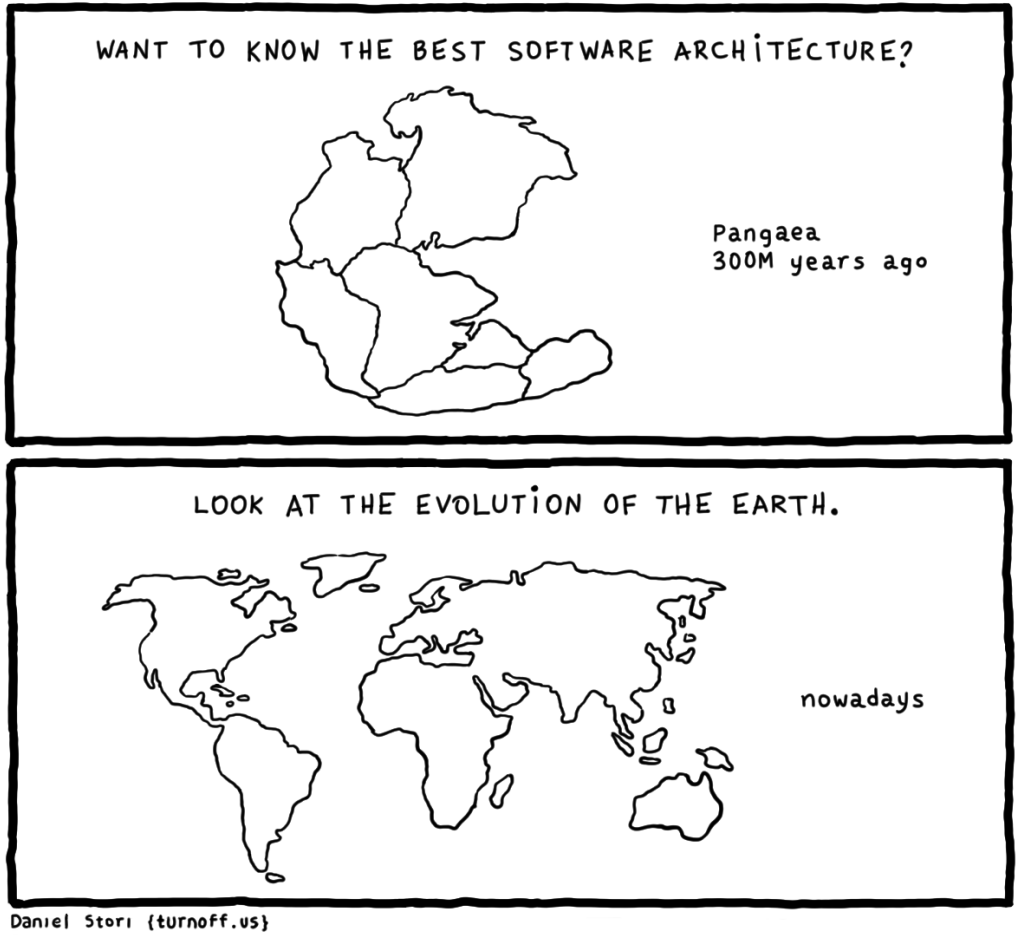
Do we need API Gateways when using Microservices?

API Days Zürich 2017

Sven Walther



ca technologies

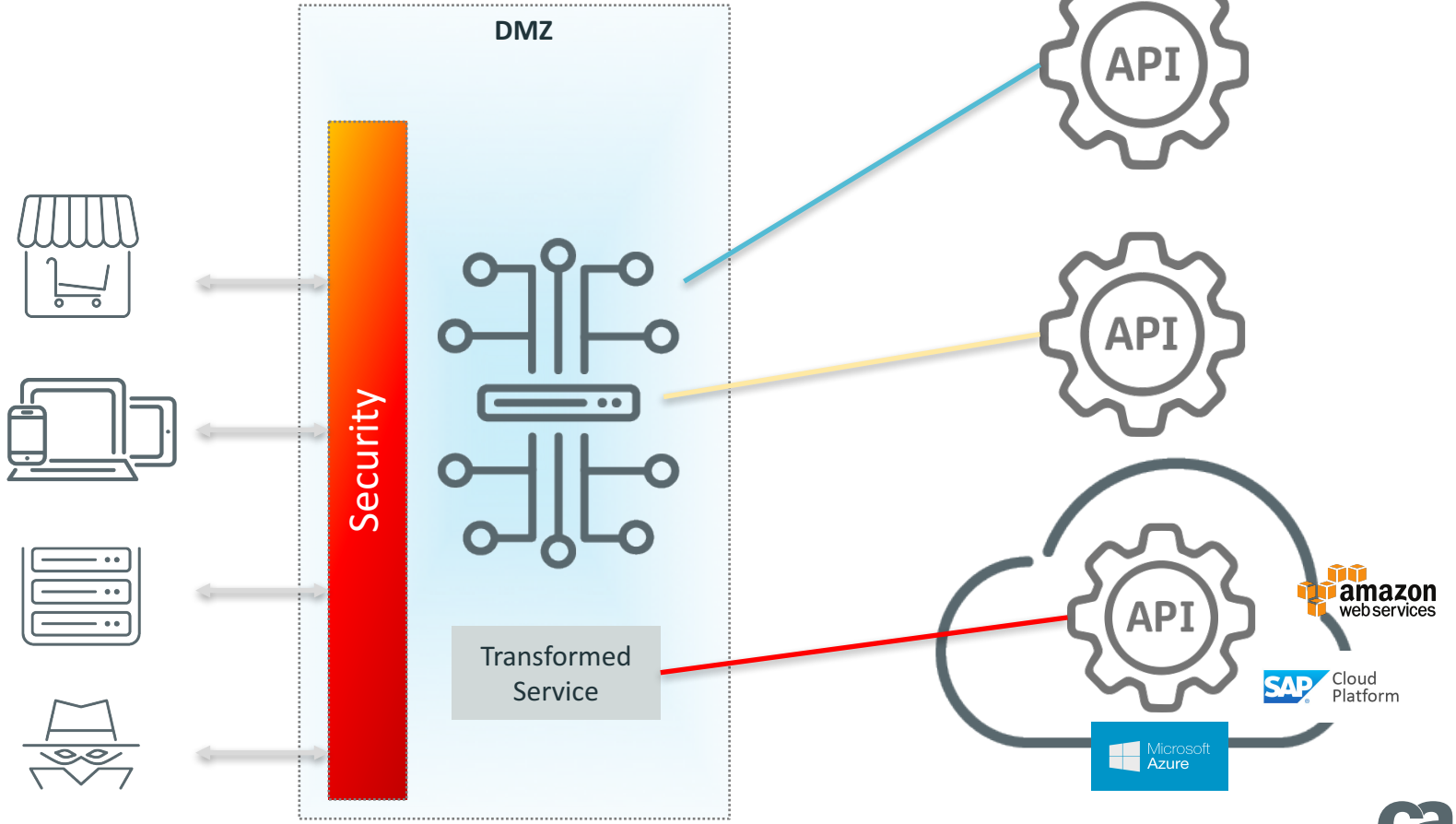


Do we need API Gateways when using Microservices?

YES!

API Gateway?

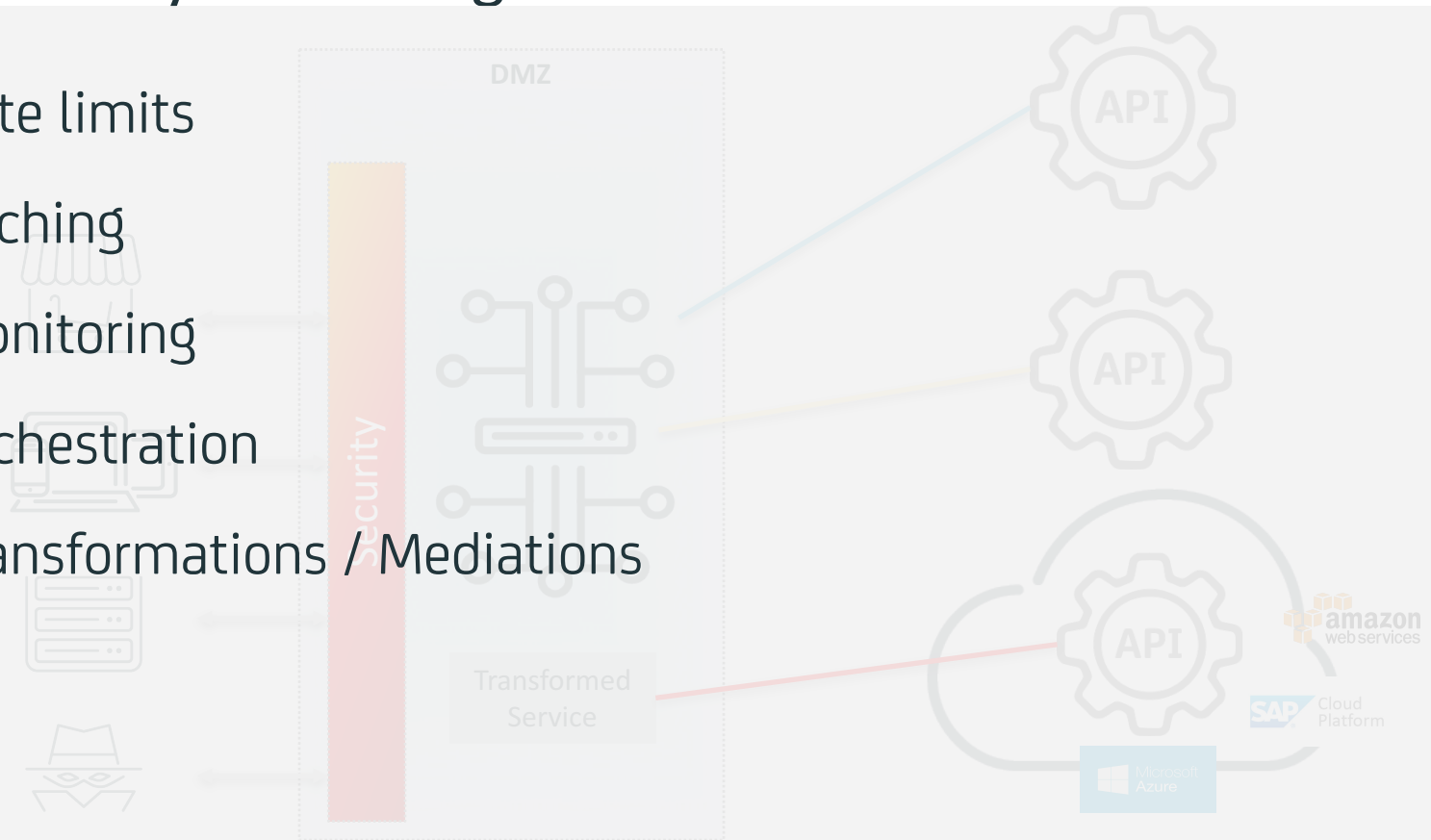
API Gateway at the edge



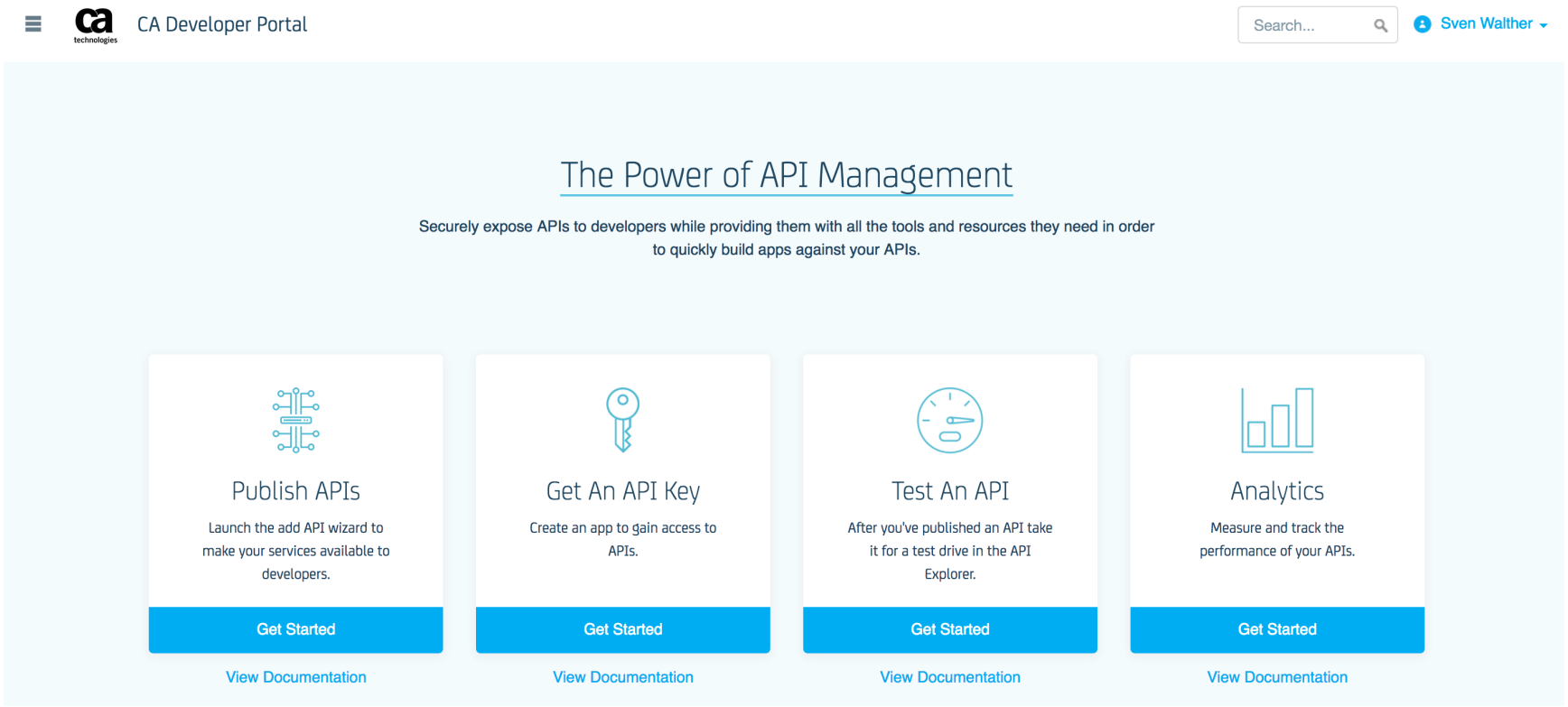
ca technologies

API Gateway at the edge

- Rate limits
- Caching
- Monitoring
- Orchestration
- Transformations / Mediations
- …



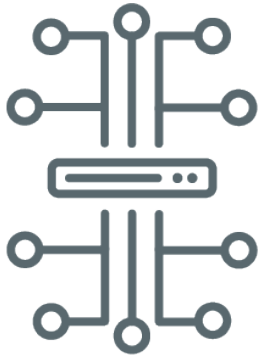
Developer Portal



But aren’t microservices somehow different?

What makes a microservice different?

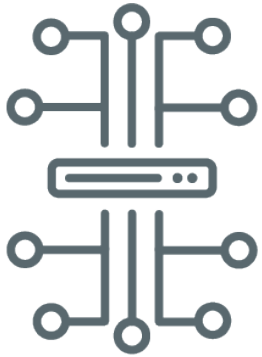
- It’s micro
- It’s not only micro – it’s lightweight
- It’s typically deployed in a containerized infrastructure
- DevOps will start up and shut down instances all the time
- It depends on a whole ecosystem



So what does a API Gateway add to it?

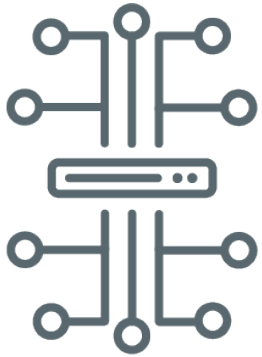
API Gateway functions for microservices

- It’s micro
  - Published services might need orchestration
- It’s not only micro – it’s lightweight
  - Security needs to be taken care of
- It’s typically deployed in a containerized infrastructure
  - Unified access at fixed hosts and URLs



API Gateway functions for microservices (cont.)

- DevOps will start up and shut down instances all the time
  - Routing to only active nodes
- It depends on a whole ecosystem
  - Central point of truth on infrastructure and documentation

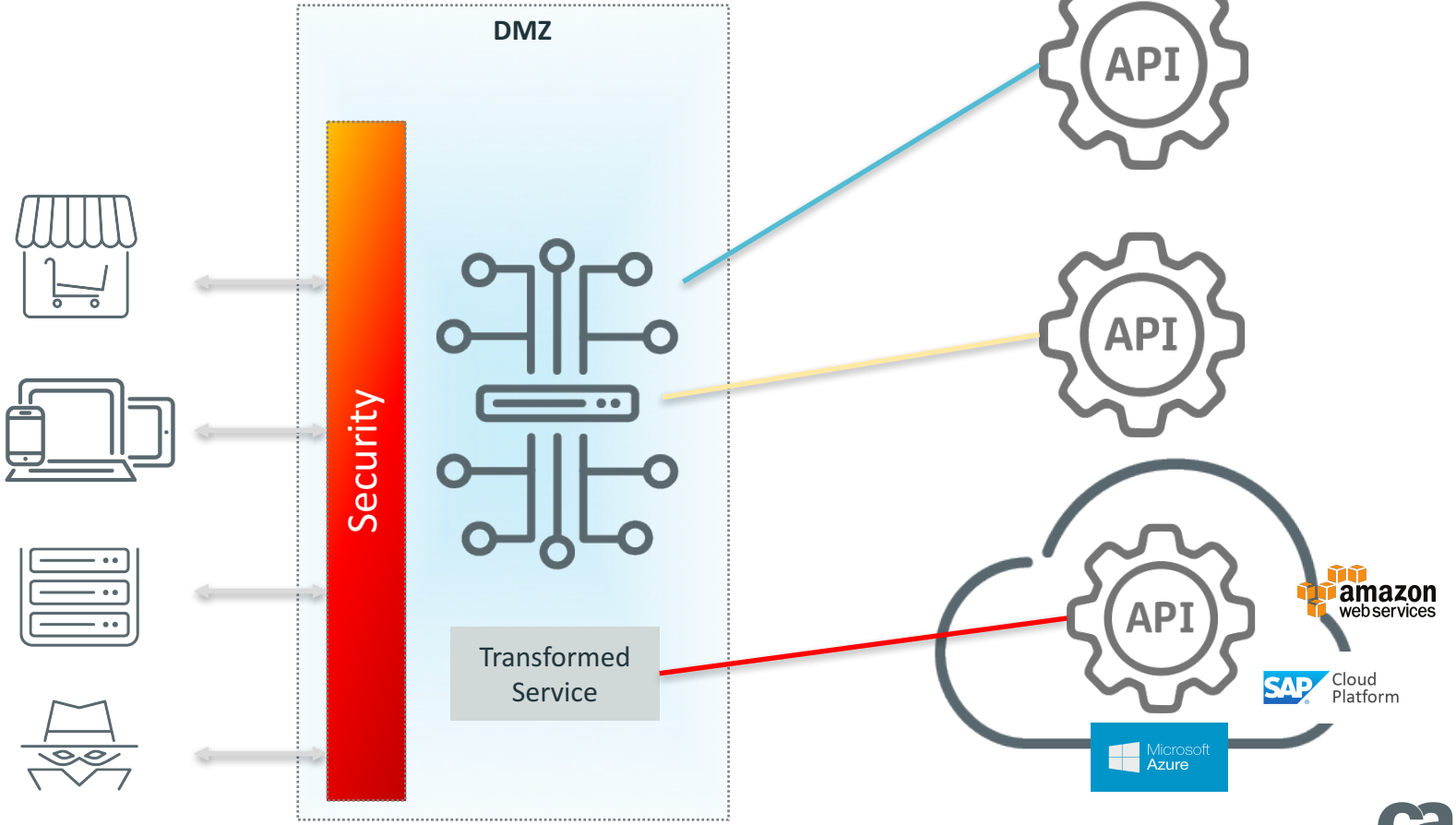


But aren’t gateways to heavy for microservices?

To heavy?

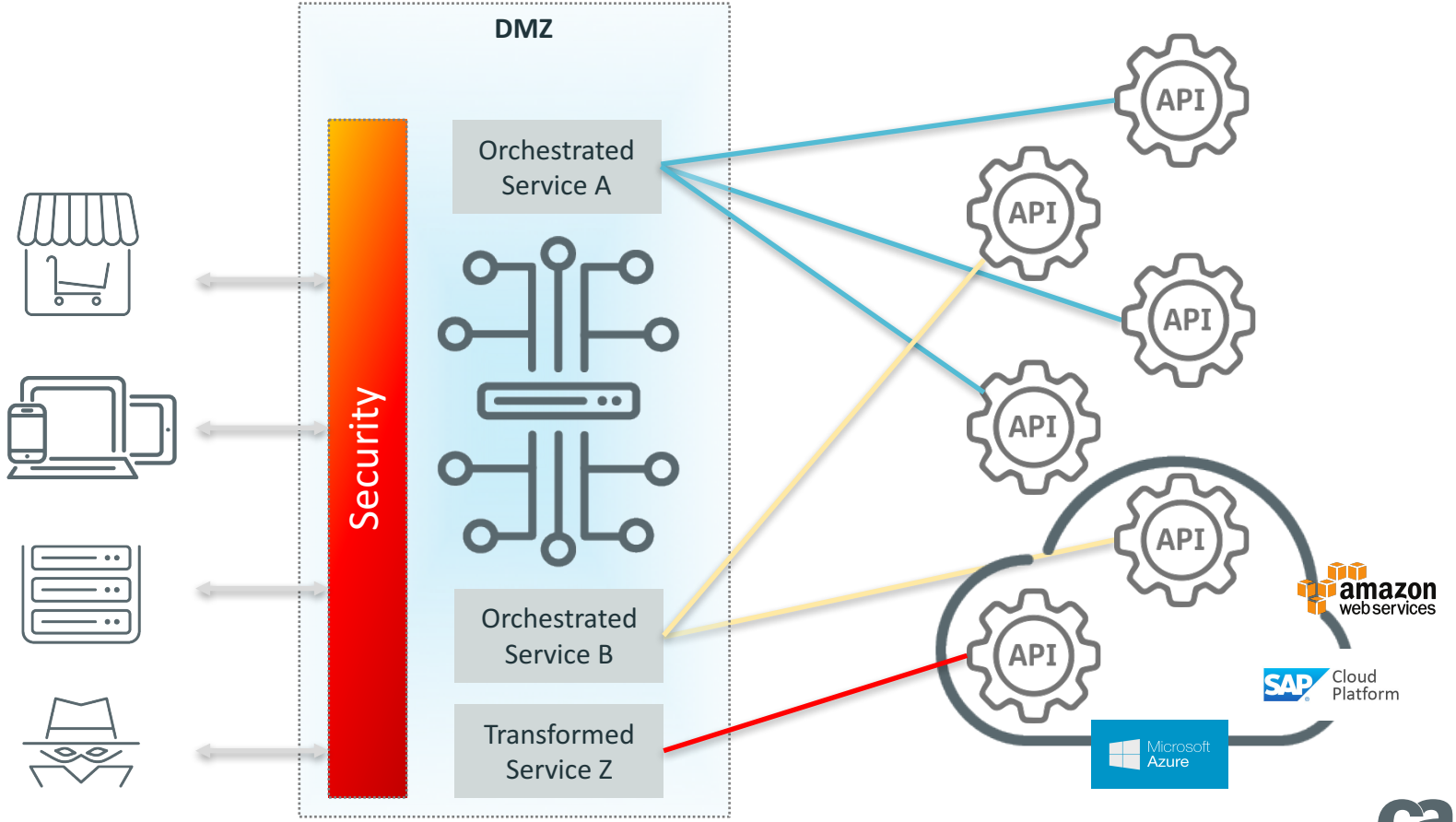
- Well, yes and no
- No: securing the edges of networks stays an important and big task

API Gateway at the edge



ca technologies

API Gateway at the edge for microservices

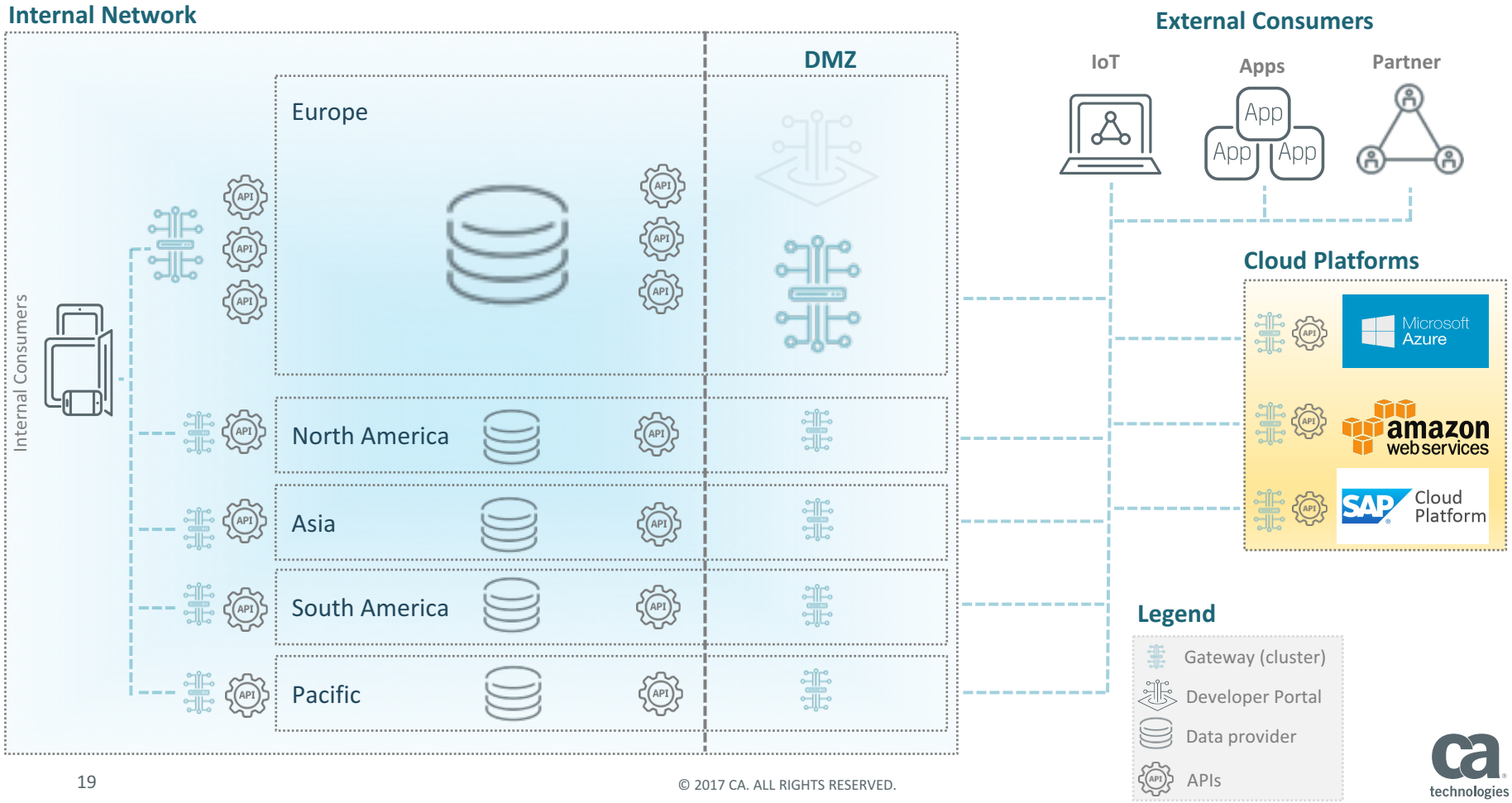


ca technologies

To heavy?

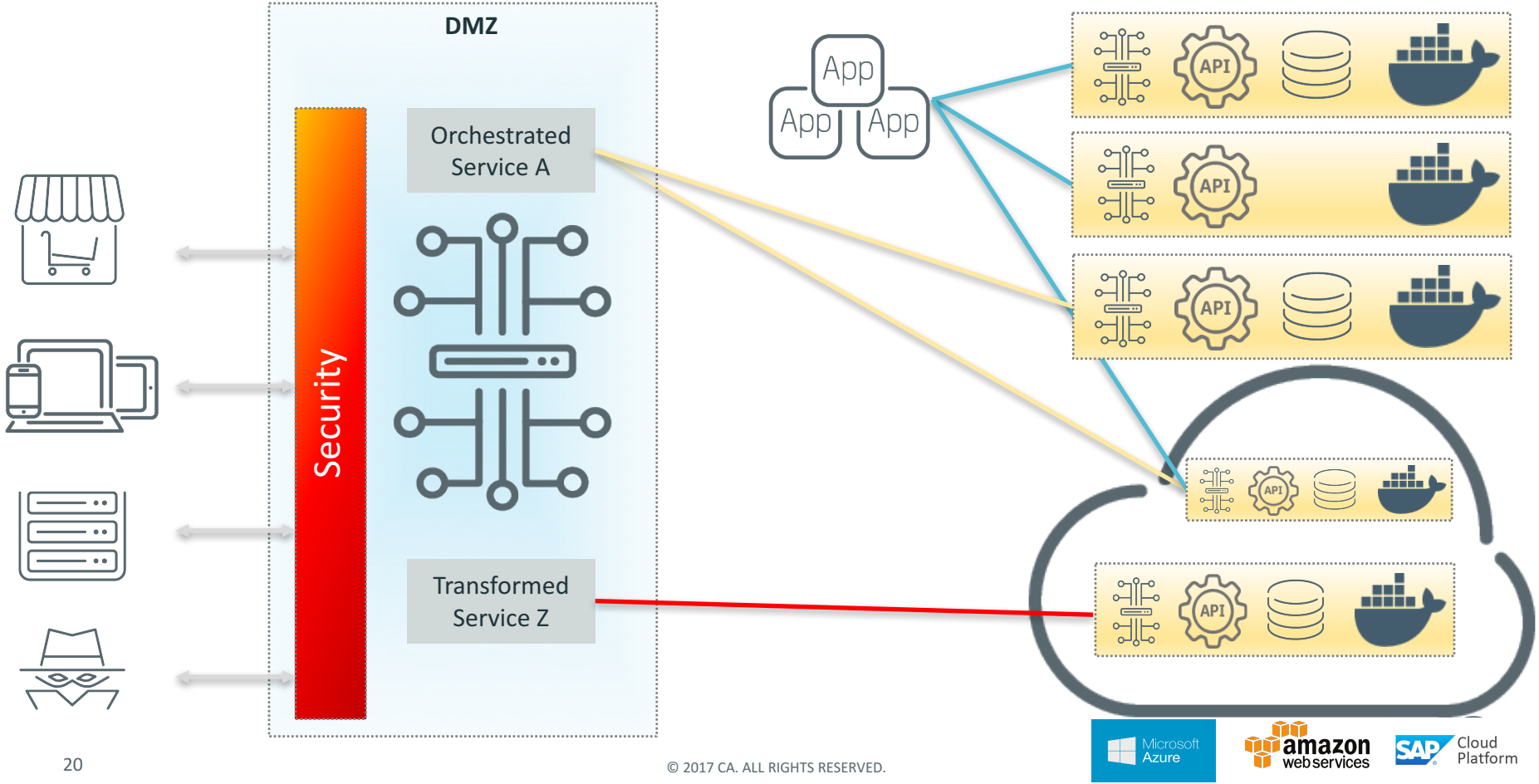
- Well, yes and no
- No: securing the edges of networks stays an important and big task
- Yes, a typical today's API Gateway might be to heavy to be flexible deployed on each container host

A real world example of a deployment



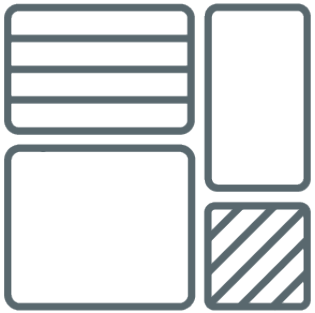
ca technologies

API Gateways managing microservices



Challenges for an API Gateway serving microservices

Internal services?



- Couldn’t we use a shrinked down version of the API Gateway for internal microservices use?

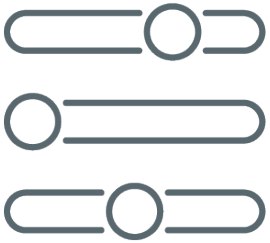
- Just reducing size und removing functionality won’t bring any real benefit – we have to rethink an API Gateway here!

A gateway for microservices: size

- An API gateway focused on microservices needs to have
  - A smaller download
  - Fewer components
  - A small footprint

- git clone https://github.com/CAAPIM/Microgateway.git

A gateway for microservices: usage



- The classical configuration with complex user interfaces cannot be adopted to microservices
- What we need is
  - Declarative config
  - Common tooling and methodologies
  - Manageable by different teams without the need of in detail knowledge of an API Gateway

Deploy a new service

Need some policies?

walsv01 — sven@docker: ~ — ssh docker

```
{
    "Service": {
    "name": "Employess",
    "gatewayUri": "/employees",
    "httpMethods": [ "get" ],
    "policy": [
      {
        "RouteHttp" : {
          "targetUrl": "http://myInternalHost/employees${request.url.query}",
          "httpMethod" : "Automatic",
          "useAuthenticationHeader" : true
        }
      }
      {
        "RateLimit" : {
          "maxRequestsPerSecond": 1,
          "hardLimit": true,
          "counterName": "RateLimit-${request.clientId}-b0938b7ad6ff"
        }
      }
    ]
  }
}
(END)
```

Template based policies

- Let central gateway administer your corporate standards
- Let DevOps apply those standards easily

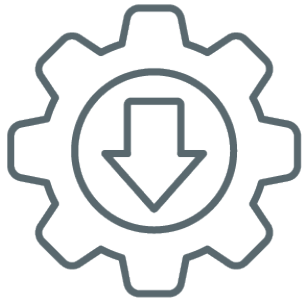
Available Encapsulated Assertions:

1. Encapsulated Assertion: CodeInjectionProtection
2. Encapsulated Assertion: Cors
3. Encapsulated Assertion: CredentialSourceHttpBasic
4. Encapsulated Assertion: RateLimit
5. Encapsulated Assertion: RequireOauth2Token
6. Encapsulated Assertion: RequireSsl
7. Encapsulated Assertion: RouteHttp
8. Encapsulated Assertion: RouteOrchestrator

Encapsulated Assertion: CodeInjectionProtection

Description: Immediately fail messages that contain injected code: HTML/JavaScript, PHP eval, shell, LDAP DN and search, and XPath.

A gateway for microservices: deployment

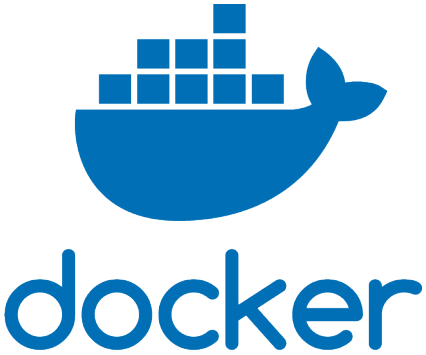


- Big containers and manual installation scripts are not fitting in the microservices space
  - Common container platforms
    - Docker
    - Openshift
    - …
  - It must be manageble by standard tools

All Docker based – one line to start your gateway

```
> docker-compose -f docker-compose.yml -f docker-compose.dockercloudproxy.yml up -d --build
```

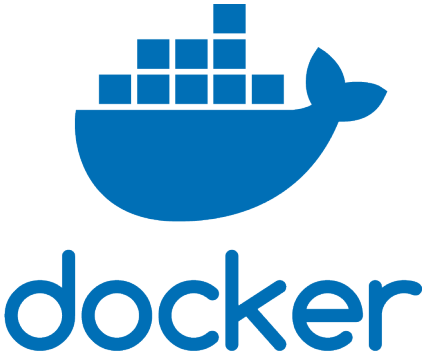
```
Creating network "dockercompose_public" with the
Building cadbhost
Step 1/2 : FROM postgres:9.6
 ---> 1227c4263c8c
Step 2/2 : ADD ./liquibase/scalerDbSchemaPostgreS
 ---> Using cache
 ---> 2f5884b4ab85
Successfully built 2f5884b4ab85
Creating dockercompose_cadbhost_1 ...
Creating dockercompose_cadbhost_1 ... done
Creating dockercompose_ssg_1 ...
Creating dockercompose_ssg_1 ... done
Creating dockercompose_proxy_1 ...
Creating dockercompose_proxy_1 ... done
>
```



Need to scale? Let's add 3 more nodes

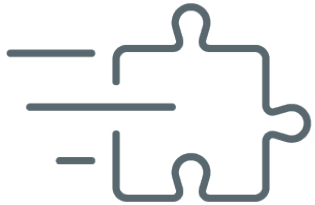
```
> docker-compose -f docker-compose.yml -f docker-compose.dockercloudproxy.yml scale ssg=4
```

```
Starting dockercompose_ssg_1 ... done
Creating dockercompose_ssg_2 ...
Creating dockercompose_ssg_3 ...
Creating dockercompose_ssg_4 ...
Creating dockercompose_ssg_2 ... done
Creating dockercompose_ssg_3 ... done
Creating dockercompose_ssg_4 ... done
>
```

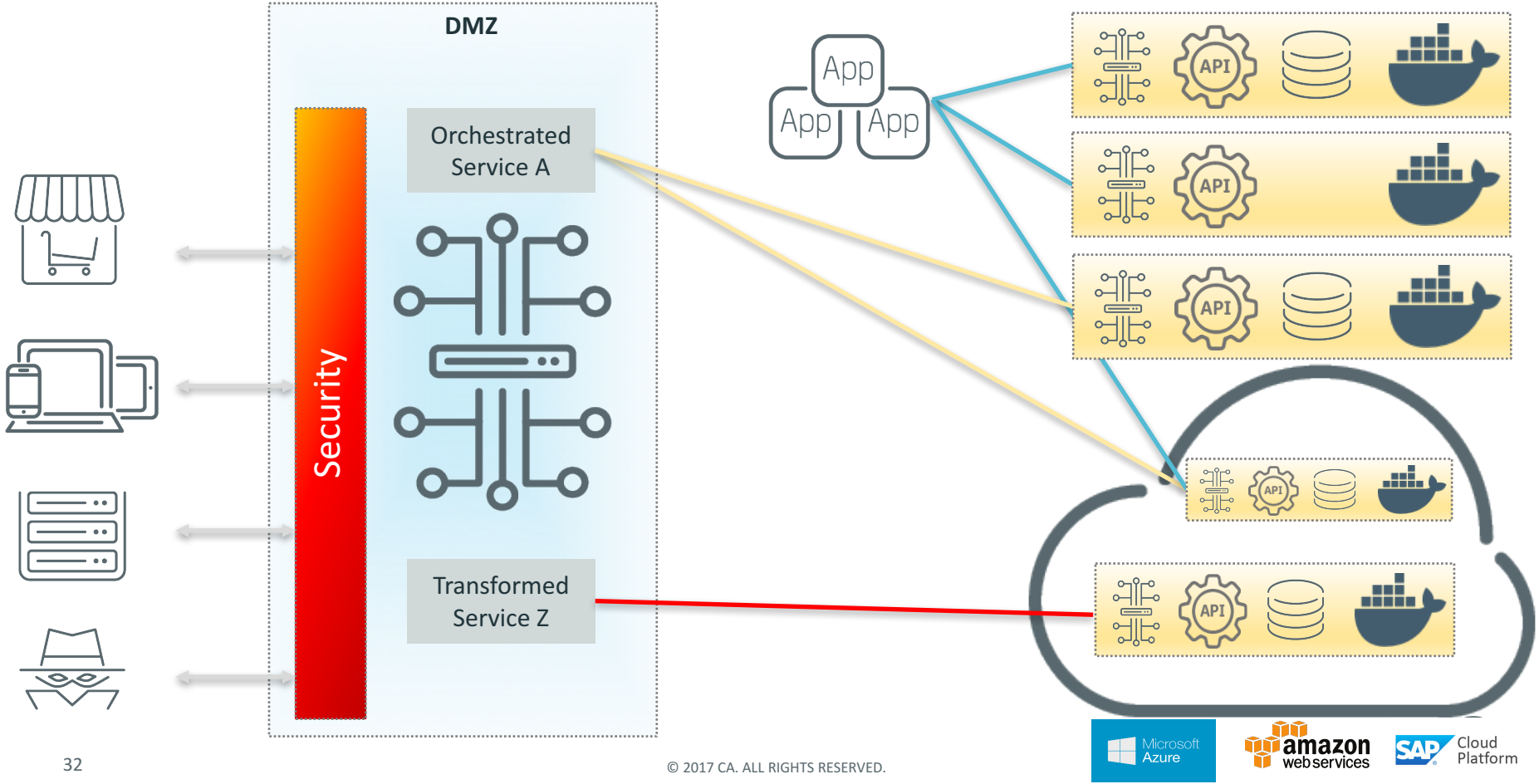


Speed and reliability

- The central goals of microservices
  - Fast provisioning
  - Fast implementation
  - Fast deployment



API Gateways managing microservices



YES!

But probably not “One Size Fits All”

Thank you

sven.walther@ca.com

@SvenWal (next to everywhere)

https://www.linkedin.com/in/svenwal/

CA API Microgateway: https://github.com/CAAPIM/Microgateway