Improving the API Gateway Pattern

Kevin Dockx
ARCHITECT

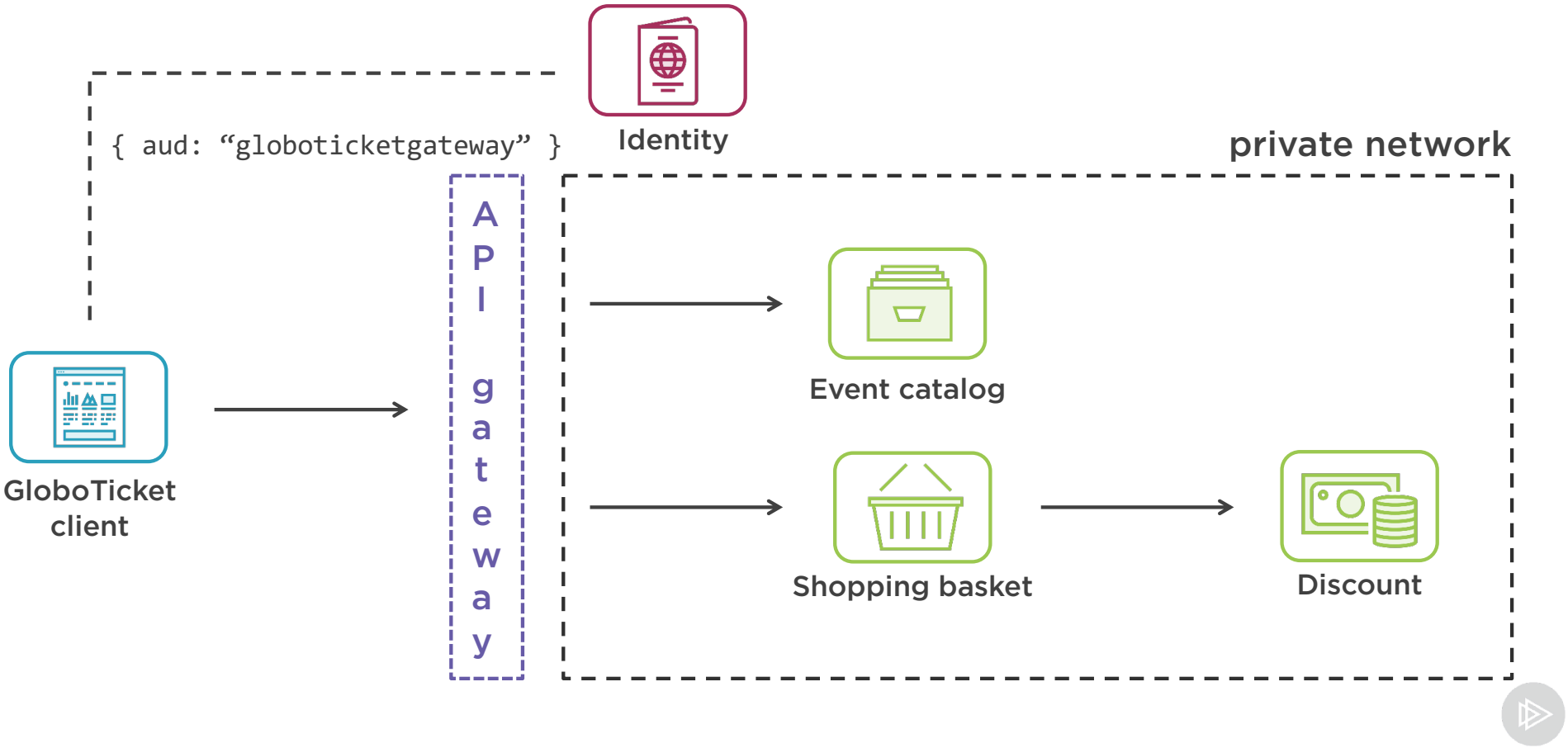
@KevinDockx https://www.kevindockx.com

Coming Up

Improving the shortcomings of the common API gateway security pattern

- Learning about the risks
- Learning how to avoid them
- Improving our code right up to a best-of-class implementation

Improving the API Gateway Pattern (Part 1)

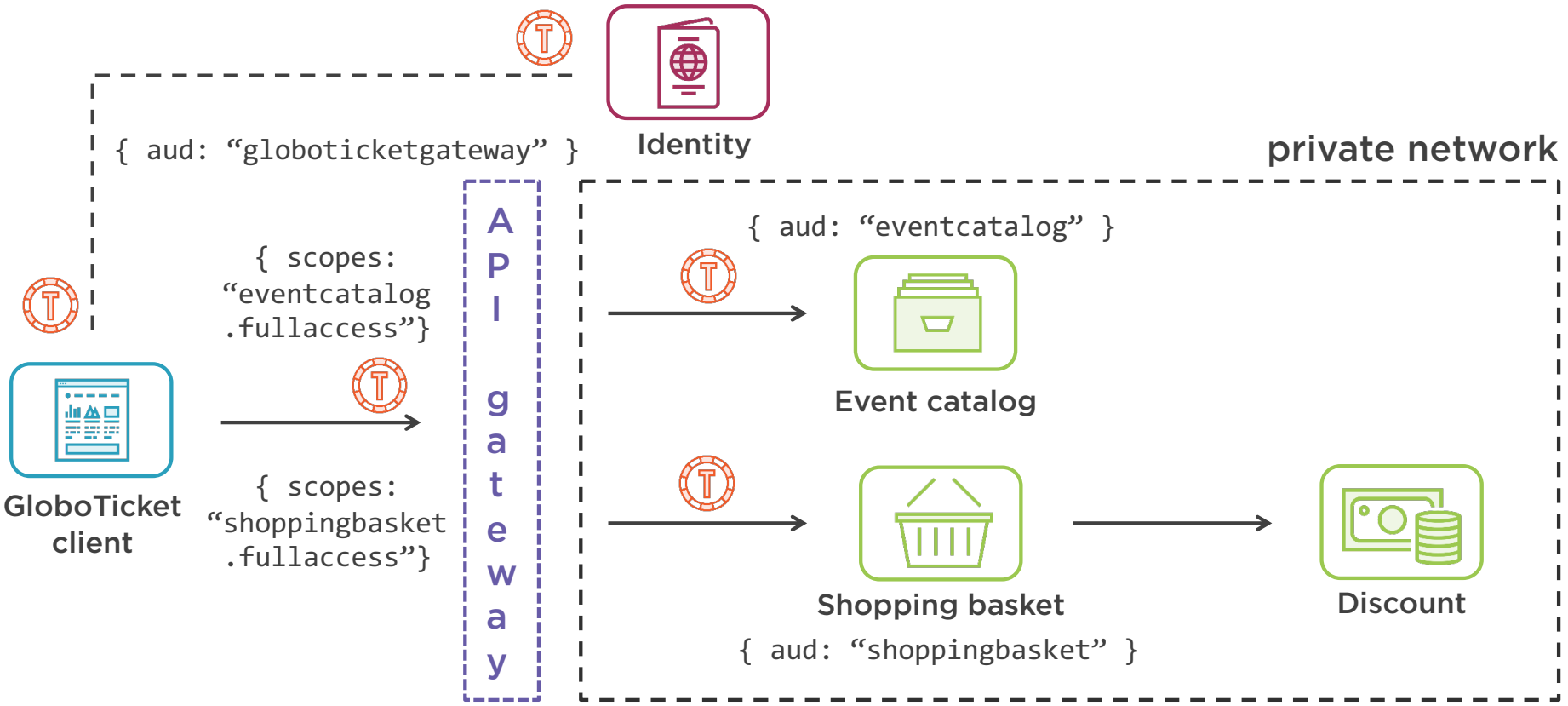


Improving the API Gateway Pattern (Part 1)

Problems with this approach

- Once inside the private network, it's free-for-all
- Our microservices, which are physically separate entities, rely on out of band mechanisms for their security...

Improving the API Gateway Pattern (Part 1)



```
{
  “aud”: [“globoticketgateway”, “eventcatalog”, “shoppingbasket”, ...]
  “scopes”: [“globoticketgateway.fullaccess”, “eventcatalog.fullaccess”,
             “shoppingbasket.fullaccess”, ...]
  ...
}
```

Example Access Token

Token contains

```
{
  “aud”: [“globoticketgateway”, “eventcatalog”, “shoppingbasket”, ...]

  “scopes”: [“globoticketgateway.fullaccess”, “eventcatalog.fullaccess”,
             “shoppingbasket.fullaccess”, ...]

  ...

}
```

Example Access Token

Token contains

- Audiences for the gateway and each microservice

```
{
  “aud”: [“globoticketgateway”, “eventcatalog”, “shoppingbasket”, ...]
  “scopes”: [“globoticketgateway.fullaccess”, “eventcatalog.fullaccess”,
             “shoppingbasket.fullaccess”, ...]
  ...
}
```

Example Access Token

Token contains

- Audiences for the gateway and each microservice
- Scopes for the gateway and microservices

Improving the API Gateway Pattern (Part 1)

Combining strategies to improve security

- Our client still only has direct access to the gateway
- With this approach we can ensure additional clients only get access to the set of microservices they really need

Upcoming Demos

At level of shopping basket and event catalog services

- Check for the respective audiences

At level of the API gateway

- Check whether the incoming token is meant for the gateway
- (Dis)allow downstream access depending on the scope(s)

Demo

Making microservices responsible for validating incoming tokens

Demo

Configuring the gateway for scope-based microservice access authorization

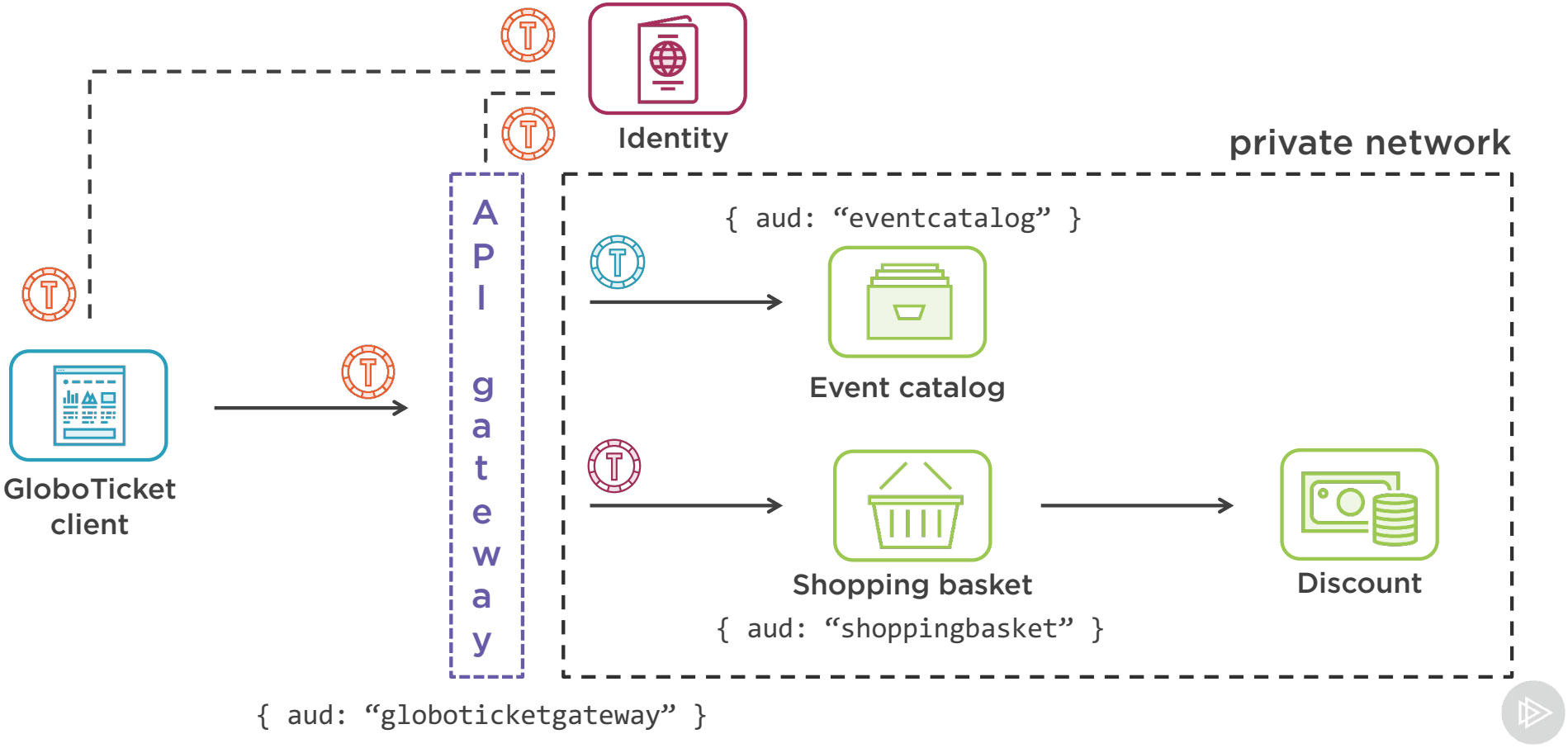
```
{
  “aud”: [“globoticketgateway”, “eventcatalog”, “shoppingbasket”, ...]
  “scopes”: [“globoticketgateway.fullaccess”, “eventcatalog.fullaccess”,
             “shoppingbasket.fullaccess”, ...]
  ...
}
```

Improving the API Gateway Pattern (Part 2)

The token is very permissive

- Can be solved with token exchange at level of the gateway

Improving the API Gateway Pattern (Part 2)



```
{
  “aud”: [“globoticketgateway”, “eventcatalog”, “shoppingbasket”, ...]
  “scopes”: [“globoticketgateway.fullaccess”, “eventcatalog.fullaccess”,
             “shoppingbasket.fullaccess”, ...]
  ...
}
```

Improving the API Gateway Pattern (Part 2)

Implementation details are exposed

```
{
  “aud”: [“globoticketgateway”, “eventcatalog”, “shoppingbasket”, ...]
  “scopes”: [“globoticketgateway.fullaccess”, “eventcatalog.fullaccess”,
             “shoppingbasket.fullaccess”, ...]
  ...
}
```

Improving the API Gateway Pattern (Part 2)

Implementation details are exposed

- Can be solved with token exchange at level of the gateway

Demo

Making the gateway responsible for exchanging tokens

Summary

A private or protected network does not mean we shouldn't have other security measures in place

- Make microservices responsible for checking incoming tokens
- Optionally use scope checks at level of the gateway to (dis)allow downstream access
- Use token exchange at level of the gateway for less permissive tokens