

OAuth 2 und OpenID Connect

JAX 2015

23.04.2015, Mainz

Simon Kölsch & Christoph Iserlohn



Agenda

- › OAuth 1, OAuth 2, OpenID Connect, O...?
- › How does it work?
- › Use Cases
- › Should I use it

“Web 2.0” Platforms



and many more...

“Web 2.0” Platforms

= User content on big platforms with APIs

flickr



and many more...

“Web 2.0” Platforms

= User content on big platforms with APIs

2010 - ~ 550.000 “Facebook Apps”

2015 - ~ 30 Flickr Apps tagged “geo”



...



and many more...

The access problem



The access problem



Resource Owner



The access problem



Resource Owner



Resource/Auth Server

The access problem



Resource Owner



Resource/Auth Server

Client

The access problem



Resource Owner



Resource/Auth Server

needs Access

Client

The access problem



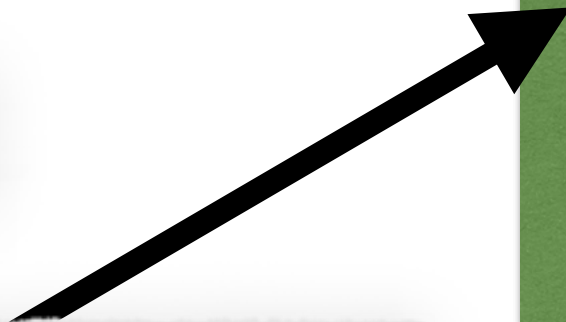
Resource Owner

gives Permission



Client

needs Access



Resource/Auth Server

The access problem



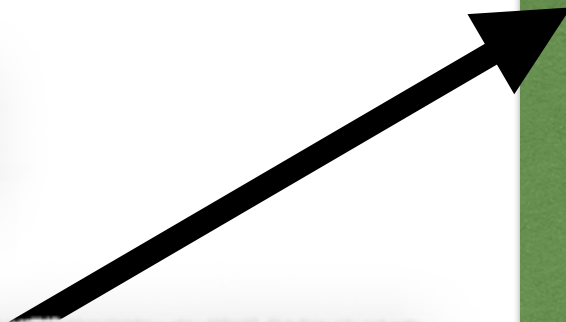
Resource Owner

gives Permission



Client

needs Access



Resource/Auth Server

checks Permission
provides Access

OAuth o



OAuth 2.0

- › Client gets user password

OAuth

- › Client gets user password
- › Client impersonates user and gets resource

OAuth 1

- › Client registers at the service

OAuth 1

- › Client registers at the service
- › Client asks resource owner for permission via the service

OAuth 1

- › Client registers at the service
- › Client asks resource owner for permission via the service
- › Client gets a token

OAuth 1

- › Client registers at the service
- › Client asks resource owner for permission via the service
- › Client gets a token
- › Client can access the resource with the token

OAuth 1

- Authorization Protocol

OAuth 1

- › Authorization Protocol
- › RFC 5849 (38 pages spec)

OAuth 1

- › Authorization Protocol
- › RFC 5849 (38 pages spec)
- › One security issue (fixed 2009)

OAuth 1

- › Authorization Protocol
- › RFC 5849 (38 pages spec)
- › One security issue (fixed 2009)
- › non-trivial request / response creation

OAuth 1

- › Authorization Protocol
- › RFC 5849 (38 pages spec)
- › One security issue (fixed 2009)
- › non-trivial request / response creation
- › Scope: “web applications”

Why OAuth 2?

- › Simplify client development process

Why OAuth 2?

- › Simplify client development process
- › Desktop Applications / Single-Page-Apps

Why OAuth 2?

- › Simplify client development process
- › Desktop Applications / Single-Page-Apps
- › Mobile Phones

Why OAuth 2?

- › Simplify client development process
- › Desktop Applications / Single-Page-Apps
- › Mobile Phones
- › “Living room devices”

OAuth 2 Roles

- › Resource Owner
- › Client
- › Resource Server
- › Authentication Server

OAuth 2 Tokens

> Access Token

`"An access token is a string representing an authorization issued to the client."`

OAuth 2 Tokens

- › Access Token

`"An access token is a string representing an authorization issued to the client."`

- › Token Type (e.g. "Bearer")

OAuth 2 Tokens

- › Access Token

`"An access token is a string representing an authorization issued to the client."`

- › Token Type (e.g. "Bearer")

- › usually short-living

OAuth 2 Tokens

- › Access Token

`"An access token is a string representing an authorization issued to the client."`

- › Token Type (e.g. "Bearer")

- › usually short-living

- › Refresh Token

OAuth 2 Tokens

- › Access Token

`"An access token is a string representing an authorization issued to the client."`

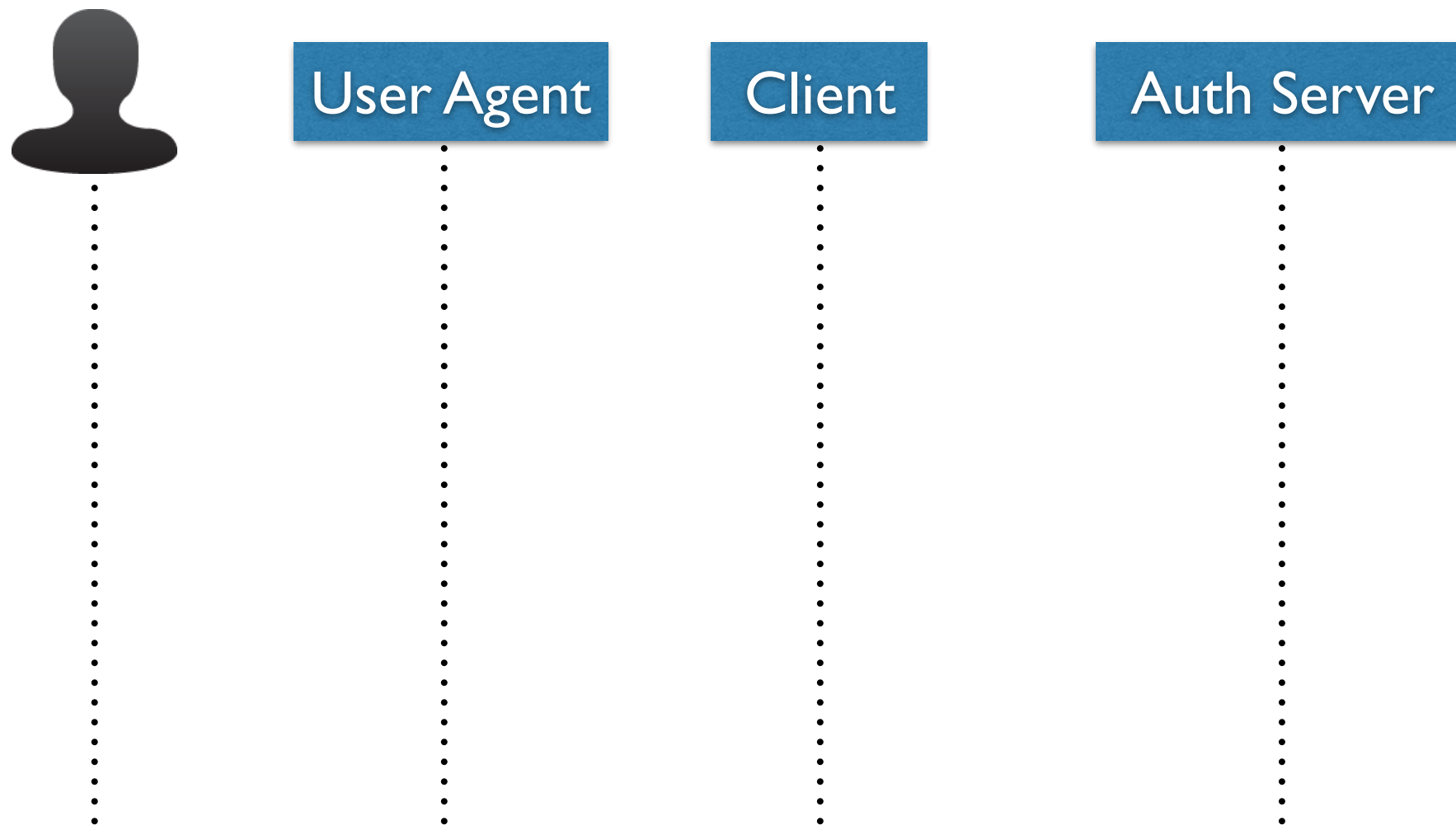
- › Token Type (e.g. "Bearer")

- › usually short-living

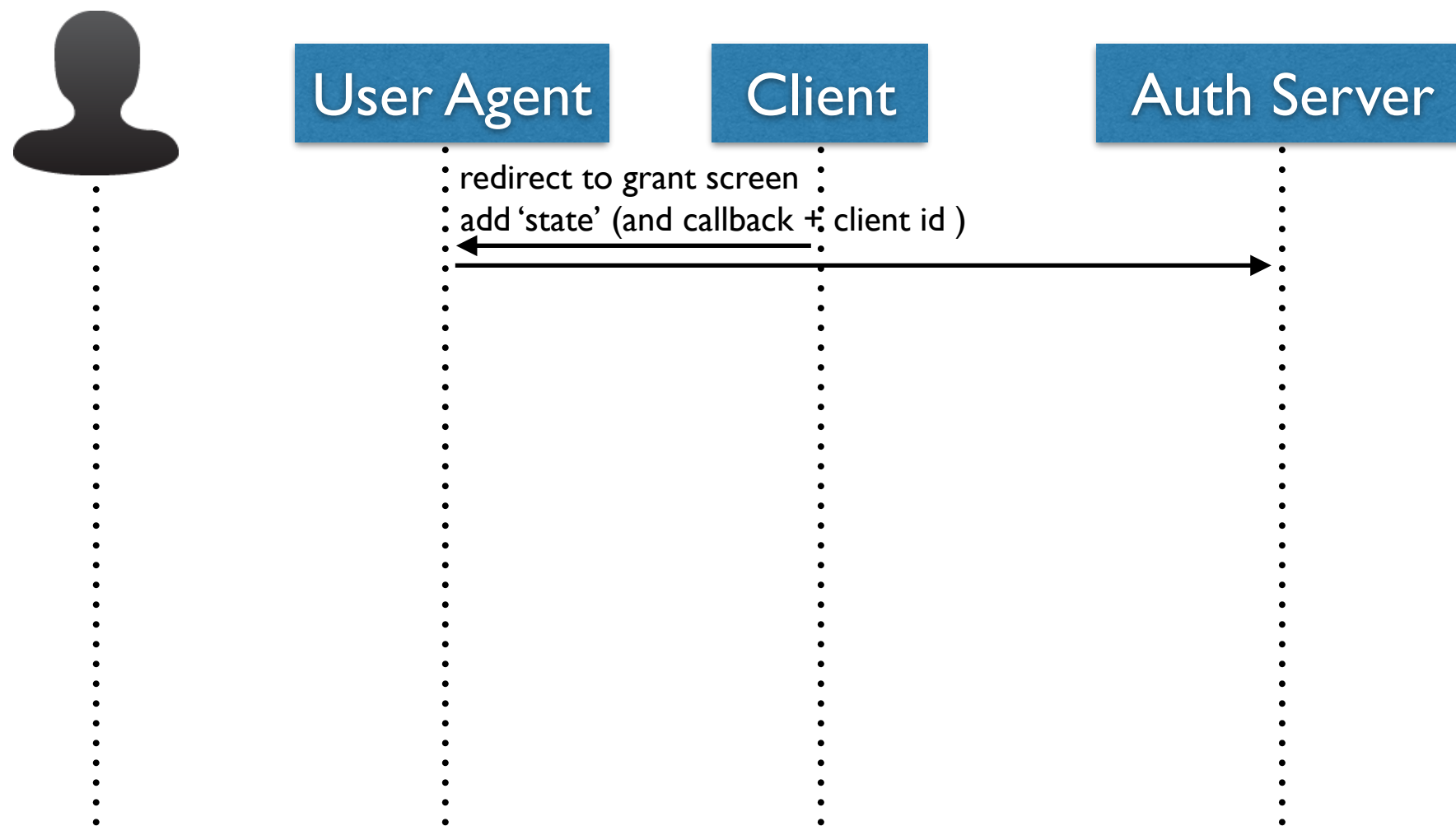
- › Refresh Token

- › Scope

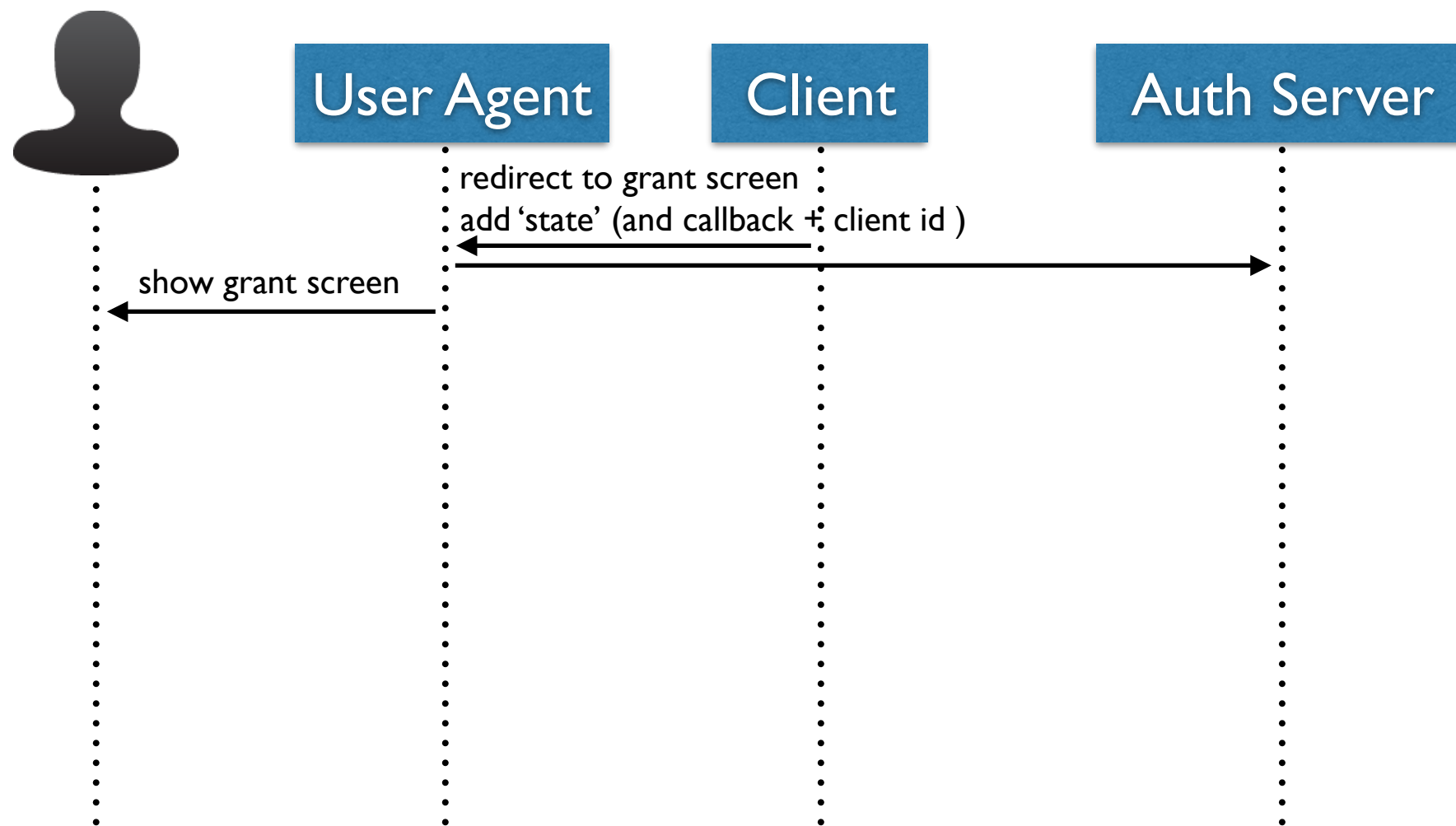
Authorization Code Grant



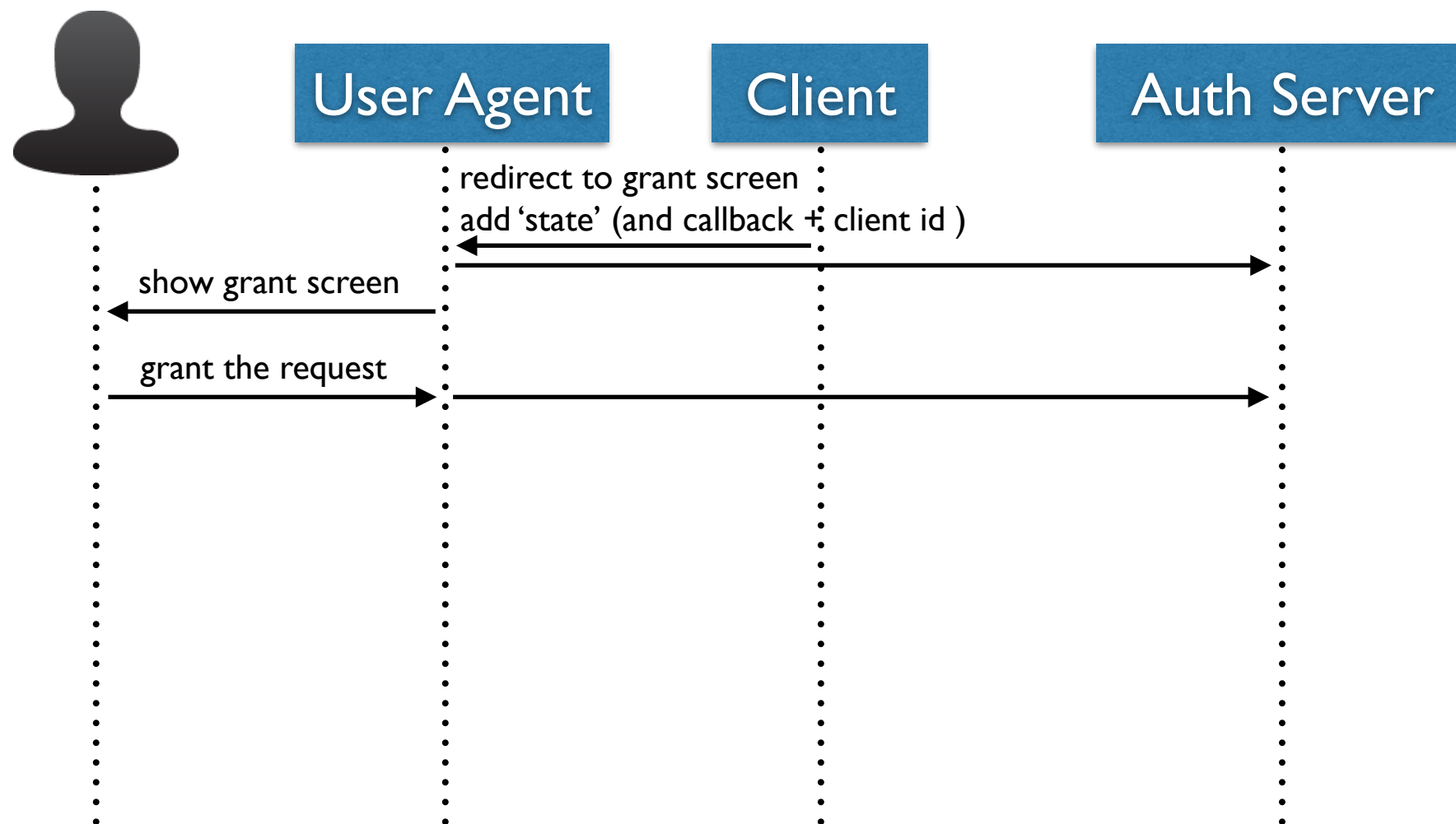
Authorization Code Grant



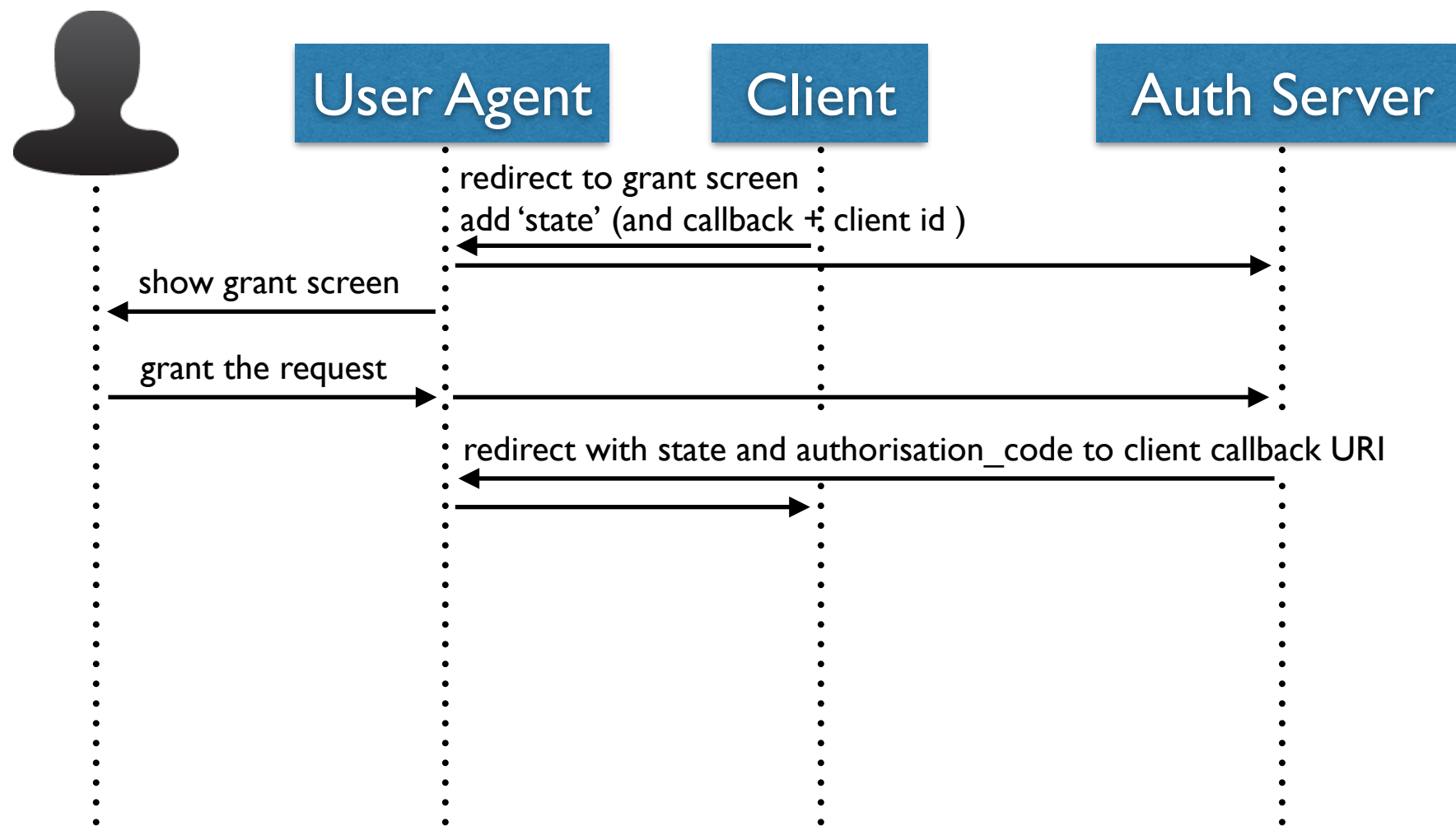
Authorization Code Grant



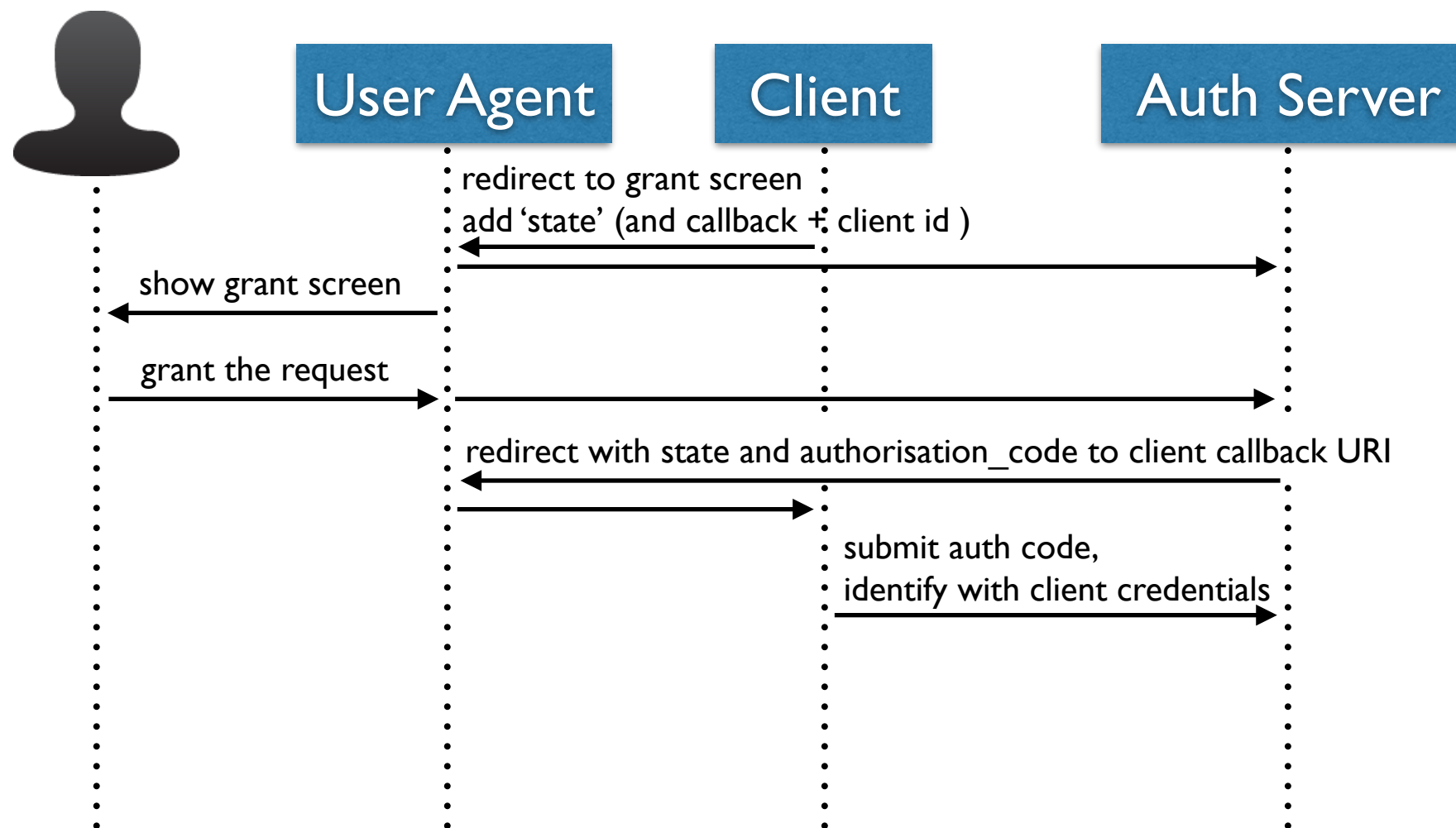
Authorization Code Grant



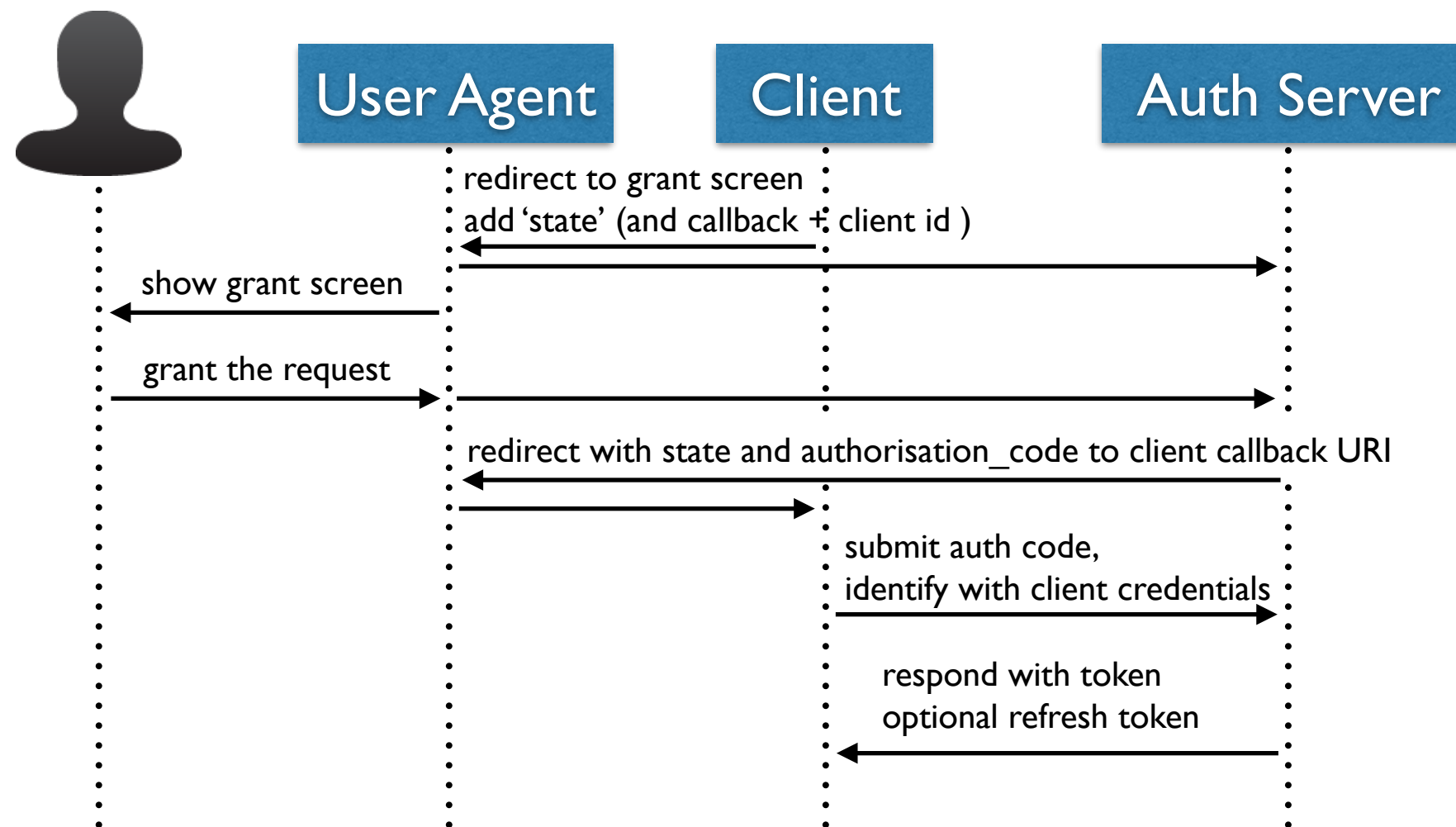
Authorization Code Grant



Authorization Code Grant



Authorization Code Grant



OAuth 2 Grant Types

- Authorization Code

OAuth 2 Grant Types

- › Authorization Code
- › Implicit

OAuth 2 Grant Types

- › Authorization Code
- › Implicit
- › User Credentials

OAuth 2 Grant Types

- › Authorization Code
- › Implicit
- › User Credentials
- › (Client Credentials)

OAuth 2 Spec Overview

Framework

OAuth 2 Spec Overview

Framework

Security

OAuth 2 Spec Overview

Framework

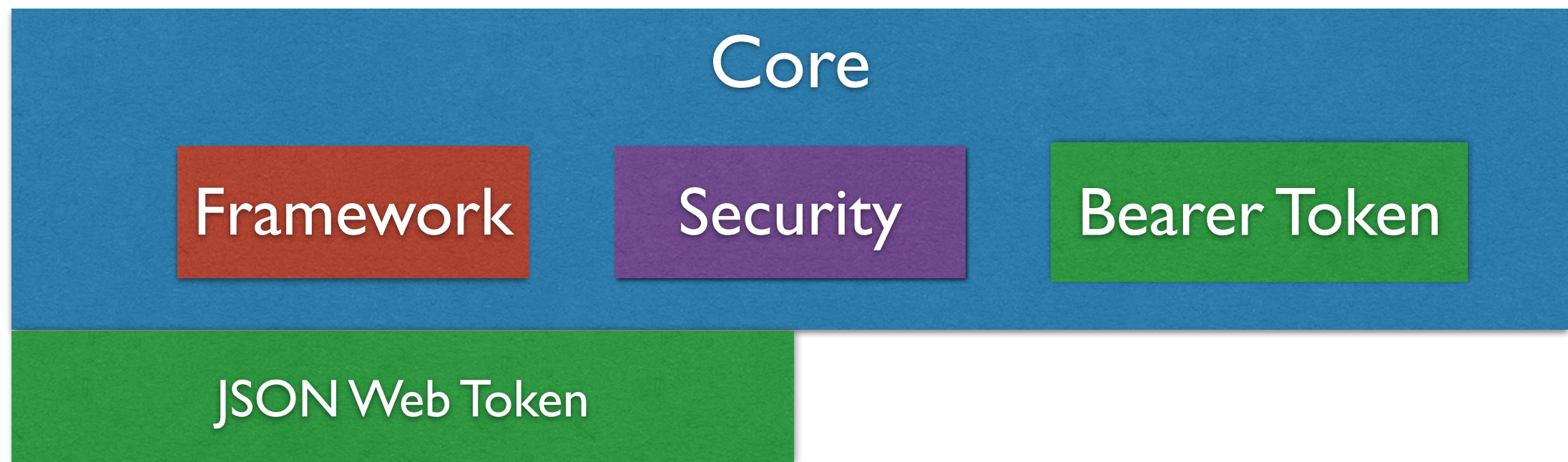
Security

Bearer Token

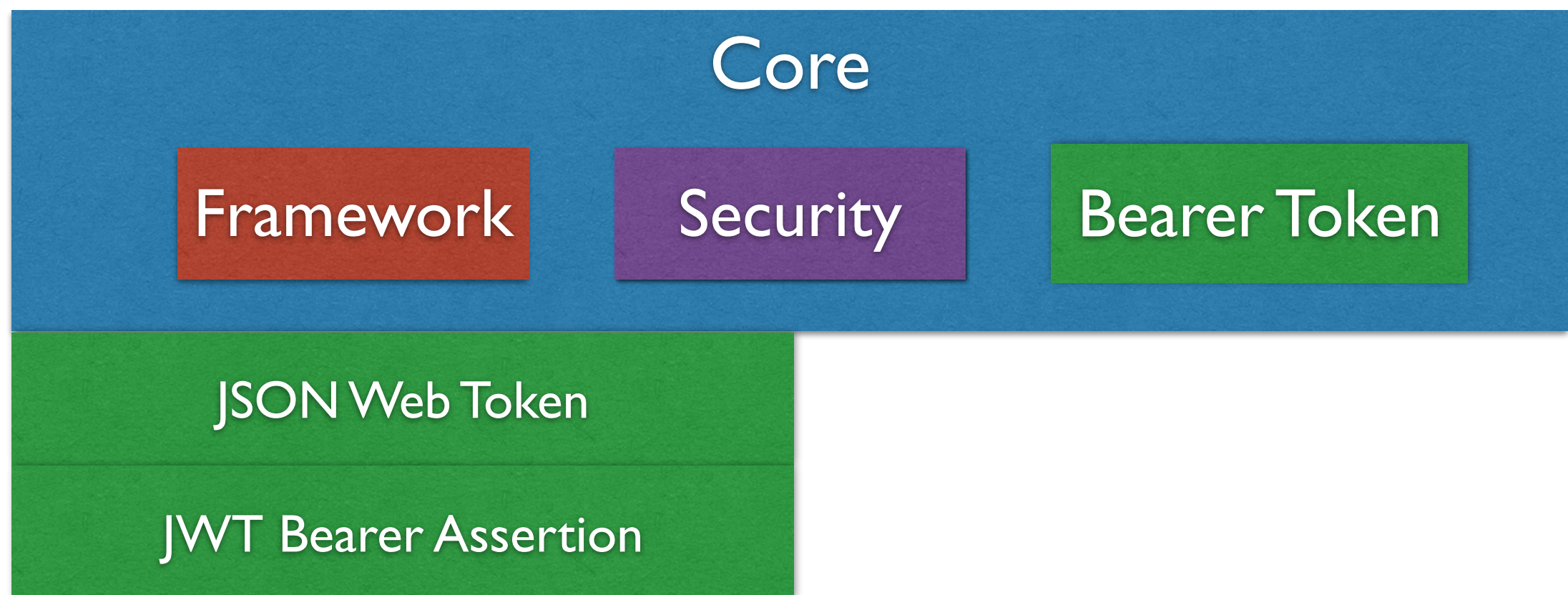
OAuth 2 Spec Overview



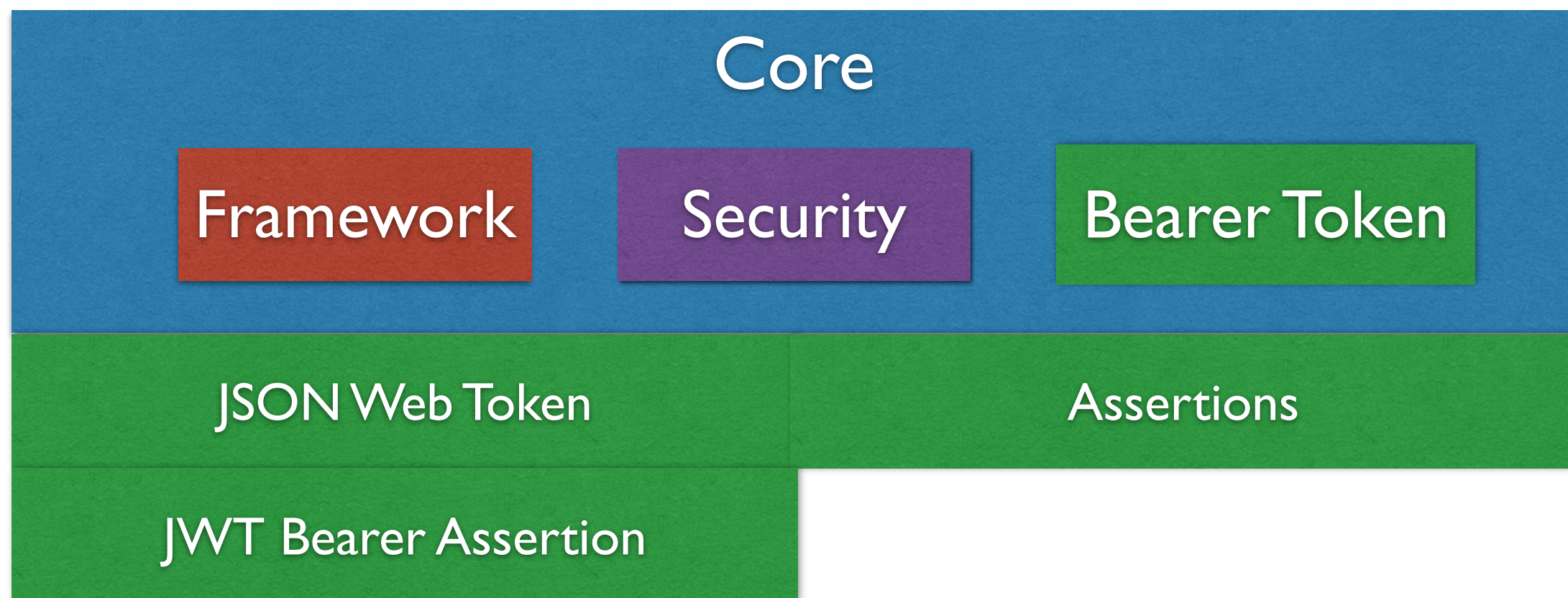
OAuth 2 Spec Overview



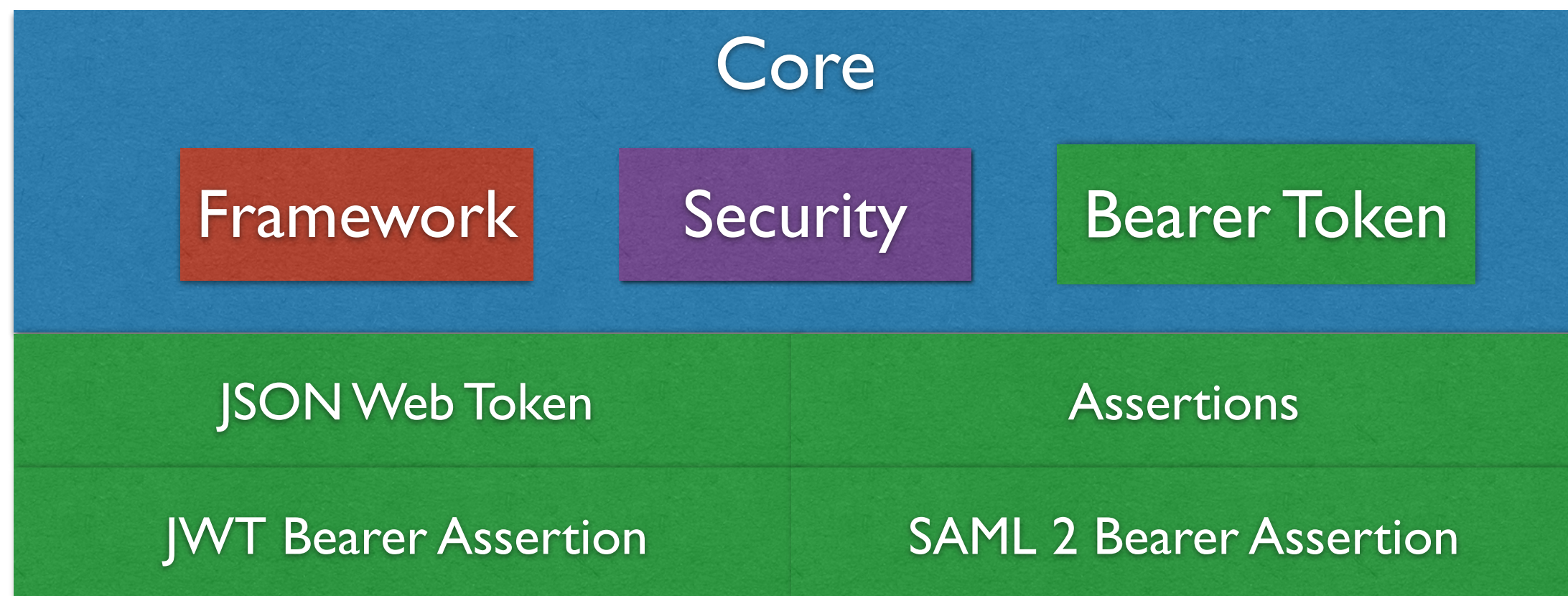
OAuth 2 Spec Overview



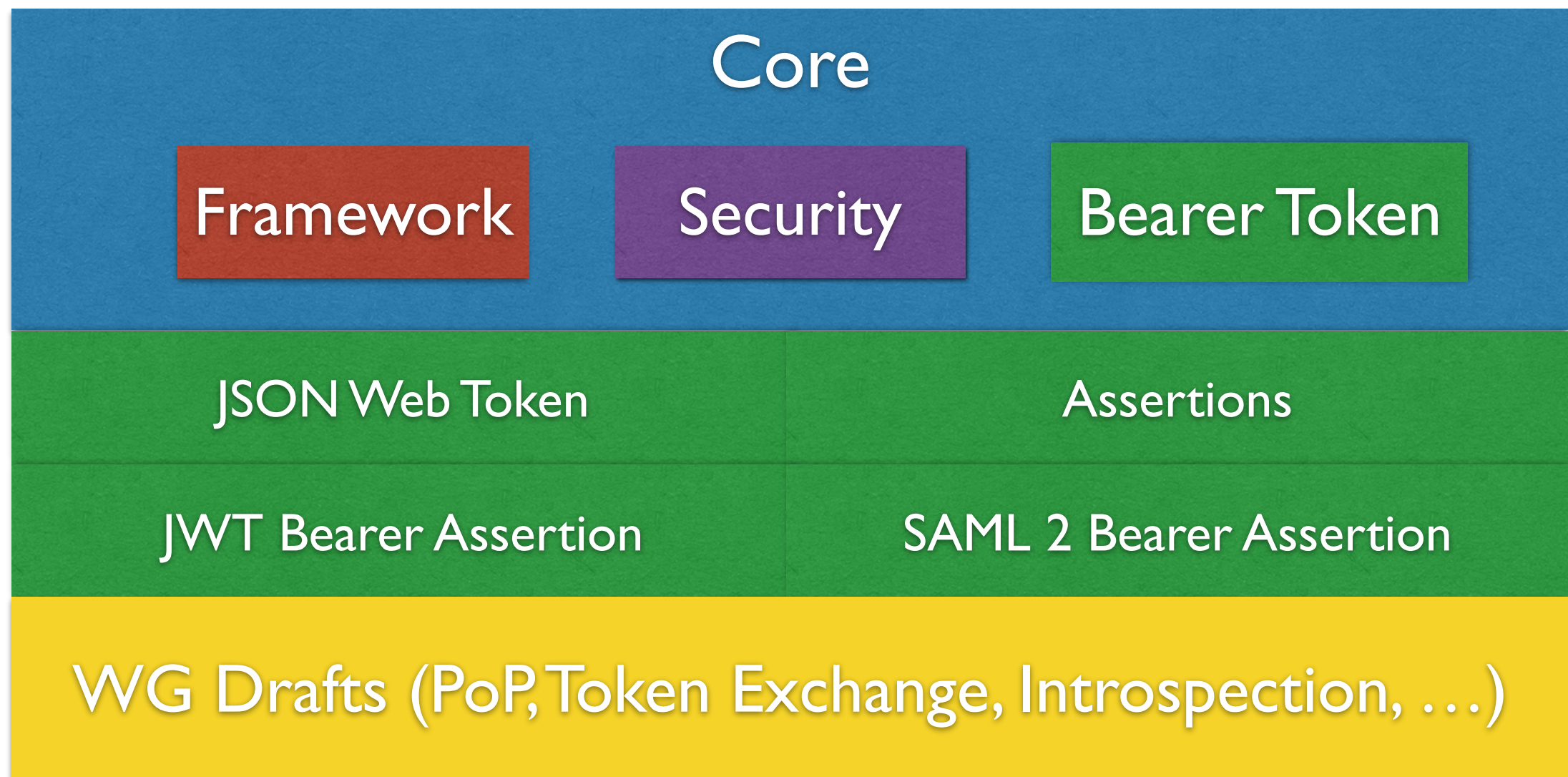
OAuth 2 Spec Overview



OAuth 2 Spec Overview



OAuth 2 Spec Overview



Short Comparison

OAuth 1	OAuth 2

Short Comparison

OAuth 1	OAuth 2
Authorization protocol	

Short Comparison

OAuth 1	OAuth 2
> Authorization protocol	> Delegation framework

Short Comparison

OAuth 1	OAuth 2
> Authorization protocol > Spec finalized	> Delegation framework

Short Comparison

OAuth 1	OAuth 2
> Authorization protocol > Spec finalized	> Delegation framework > Many extensions

- > Authorization protocol
- > Spec finalized

- > Delegation framework
- > Many extensions

Short Comparison

OAuth 1

- › Authorization protocol
- › Spec finalized
- › Additional crypto to TLS

OAuth 2

- › Delegation framework
- › Many extensions

Short Comparison

OAuth 1

- › Authorization protocol
- › Spec finalized
- › Additional crypto to TLS

OAuth 2

- › Delegation framework
- › Many extensions
- › TLS only (Core)

Short Comparison

OAuth 1

- › Authorization protocol
- › Spec finalized
- › Additional crypto to TLS
- › Production ready libs

OAuth 2

- › Delegation framework
- › Many extensions
- › TLS only (Core)

Short Comparison

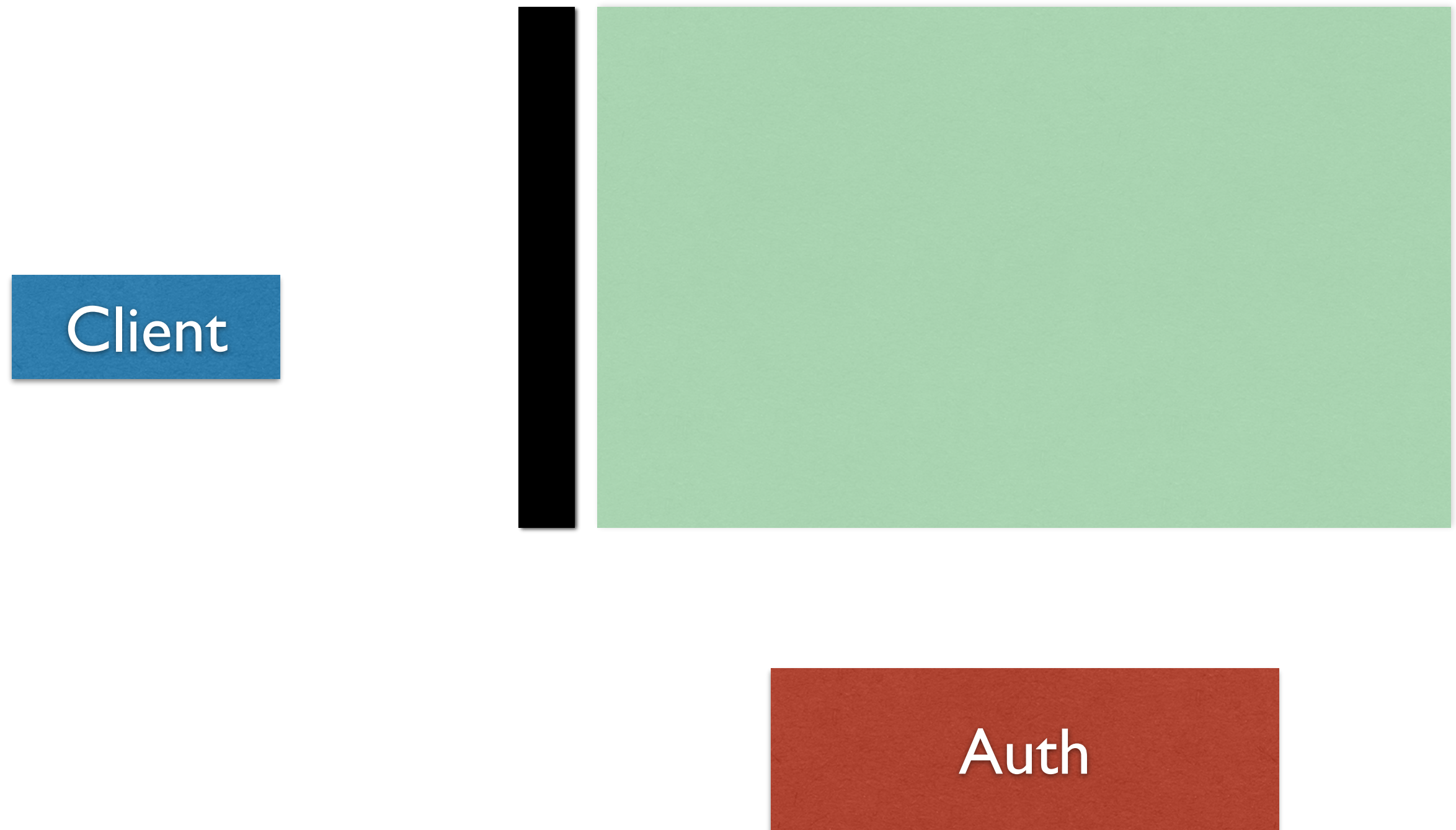
OAuth 1

- › Authorization protocol
- › Spec finalized
- › Additional crypto to TLS
- › Production ready libs

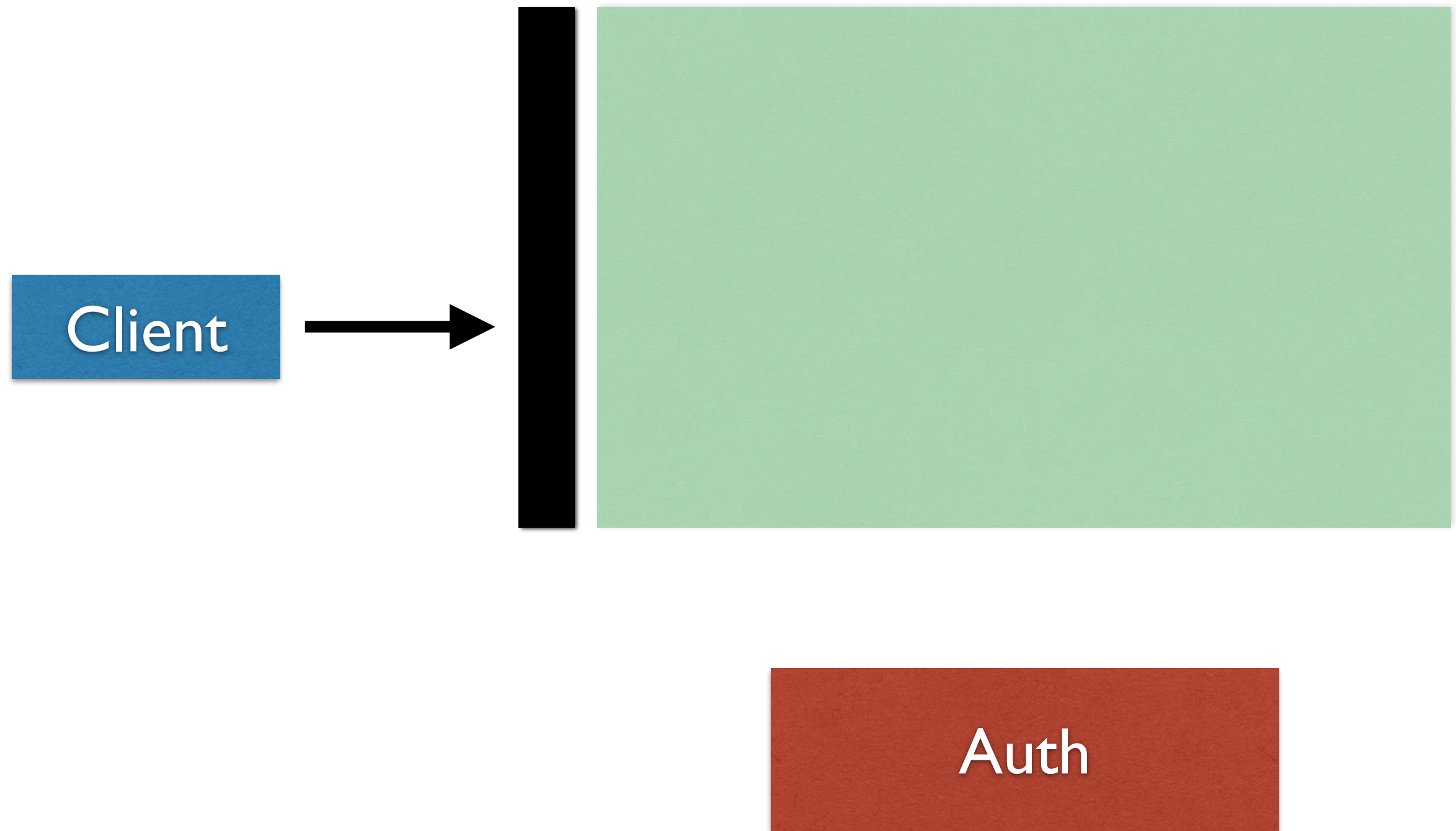
OAuth 2

- › Delegation framework
- › Many extensions
- › TLS only (Core)
- › Incompatible implementations

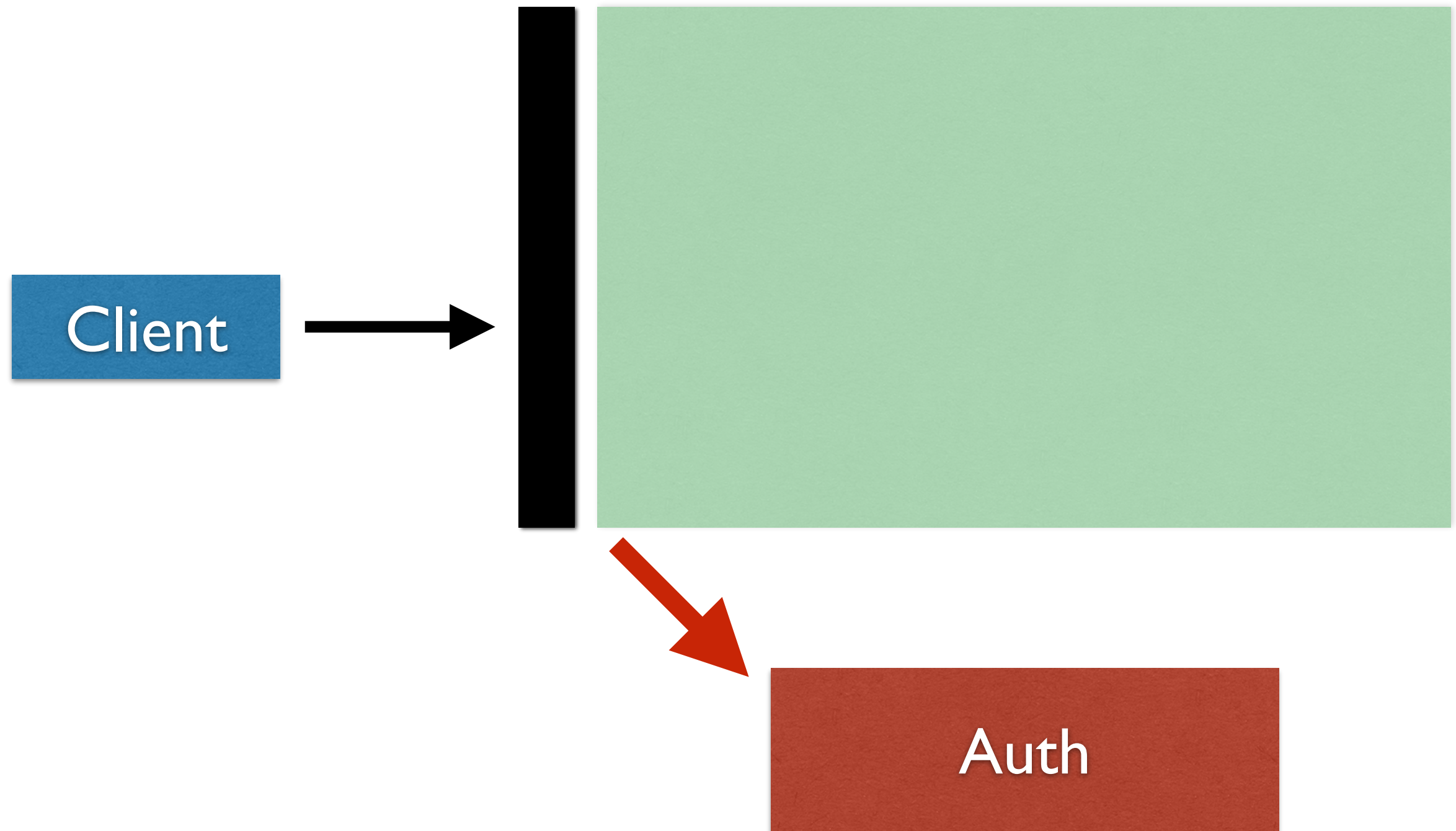
Example: Traditional System



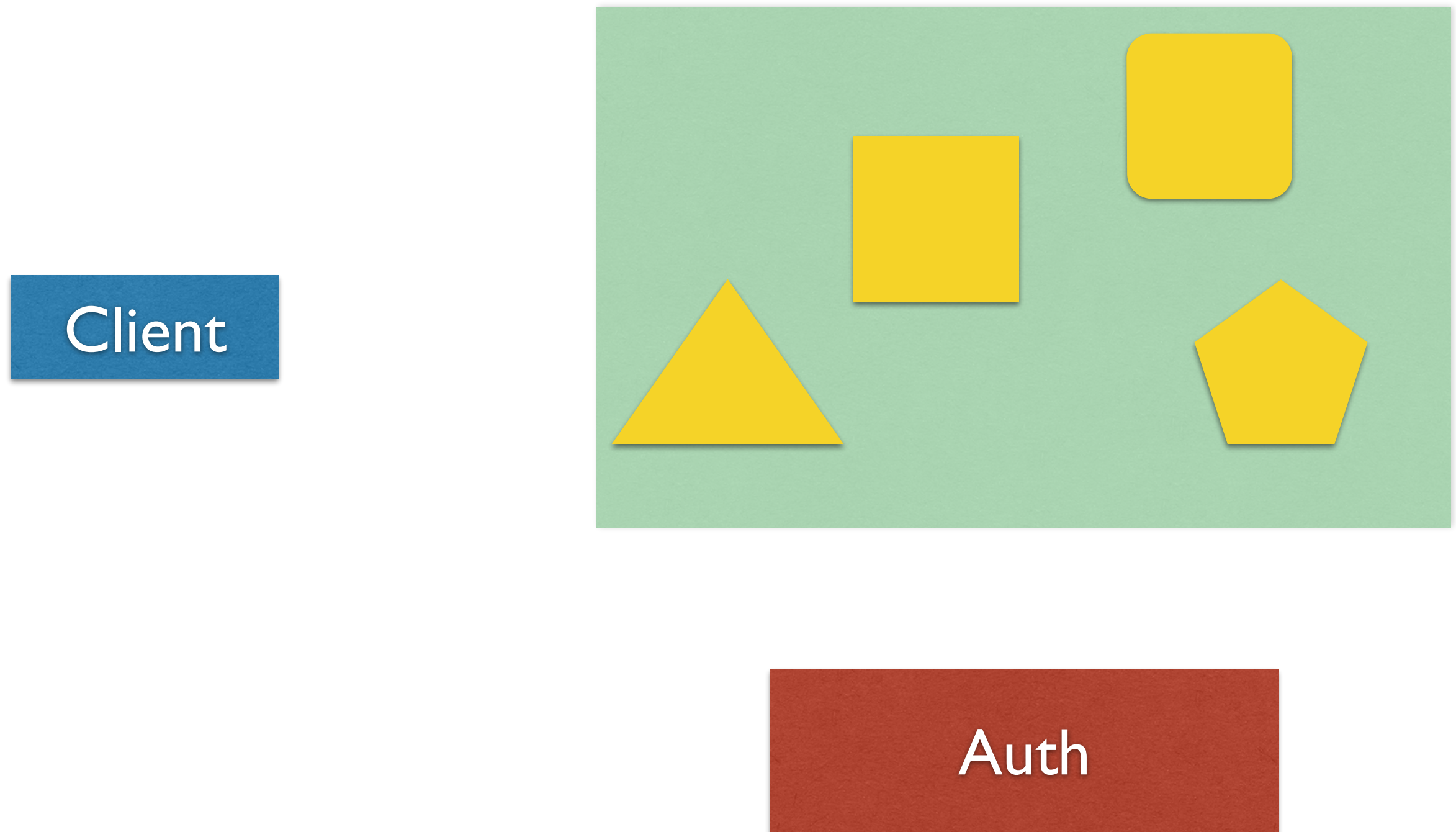
Example: Traditional System



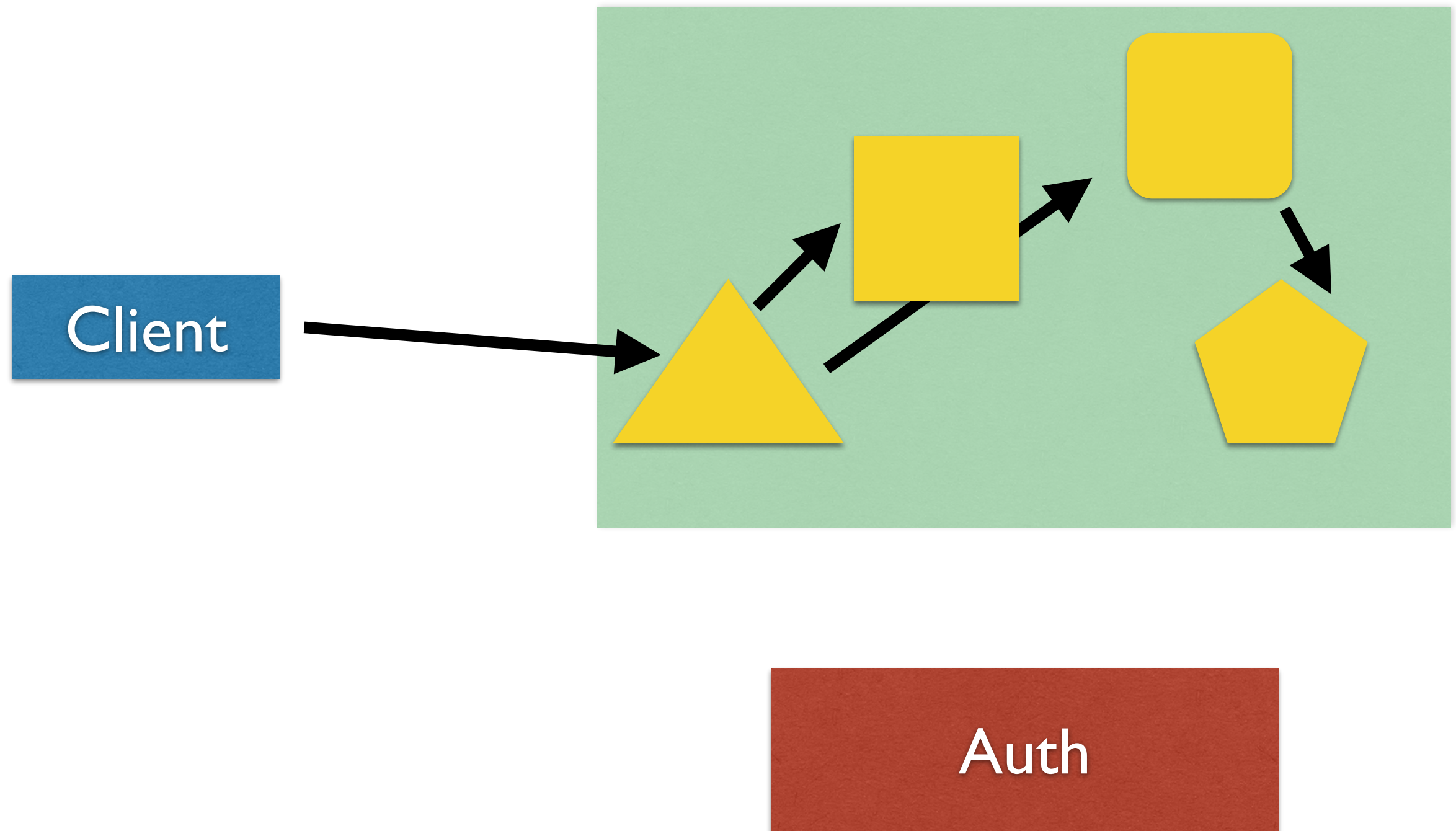
Example: Traditional System



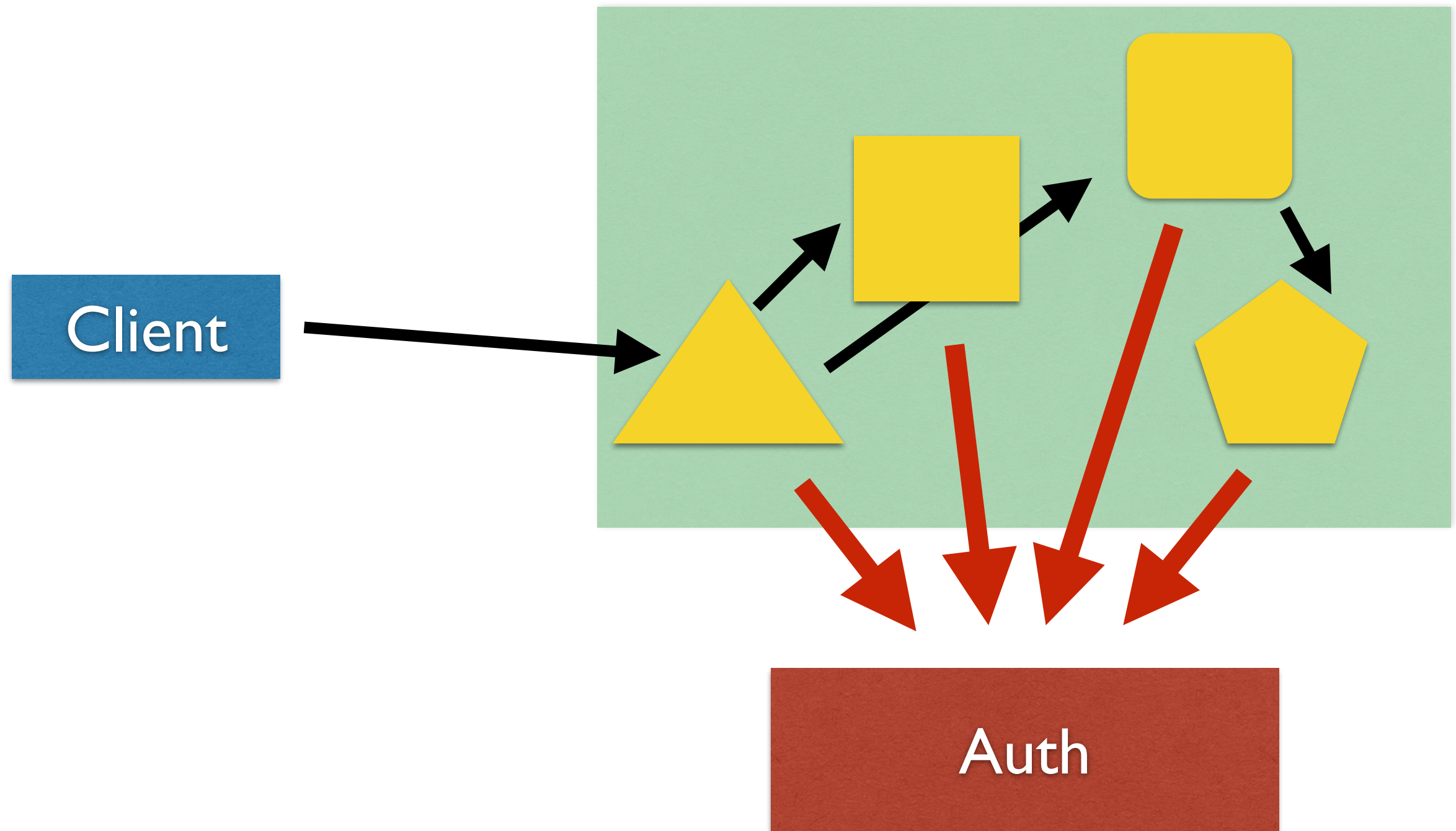
Example: Microservices



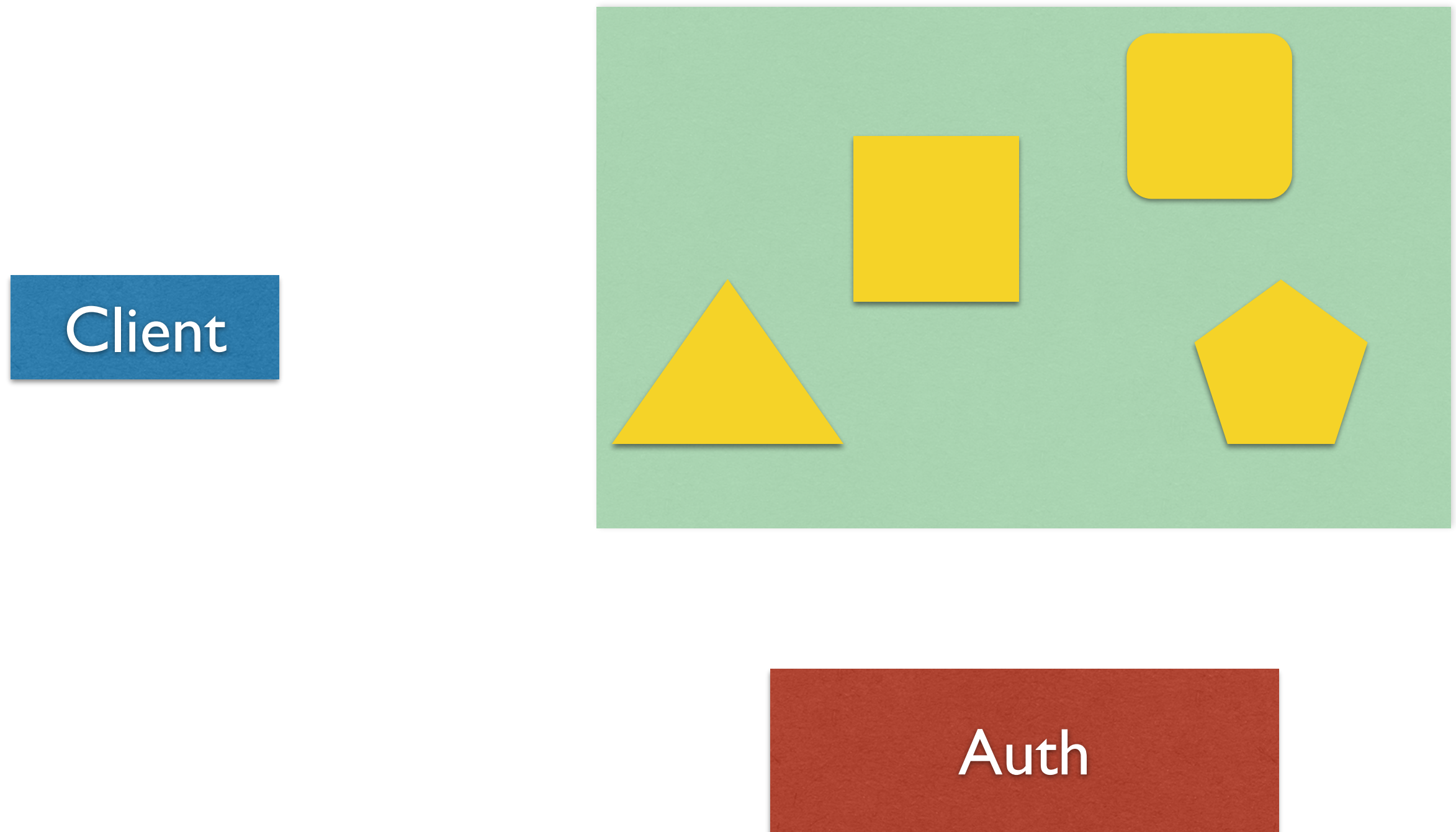
Example: Microservices



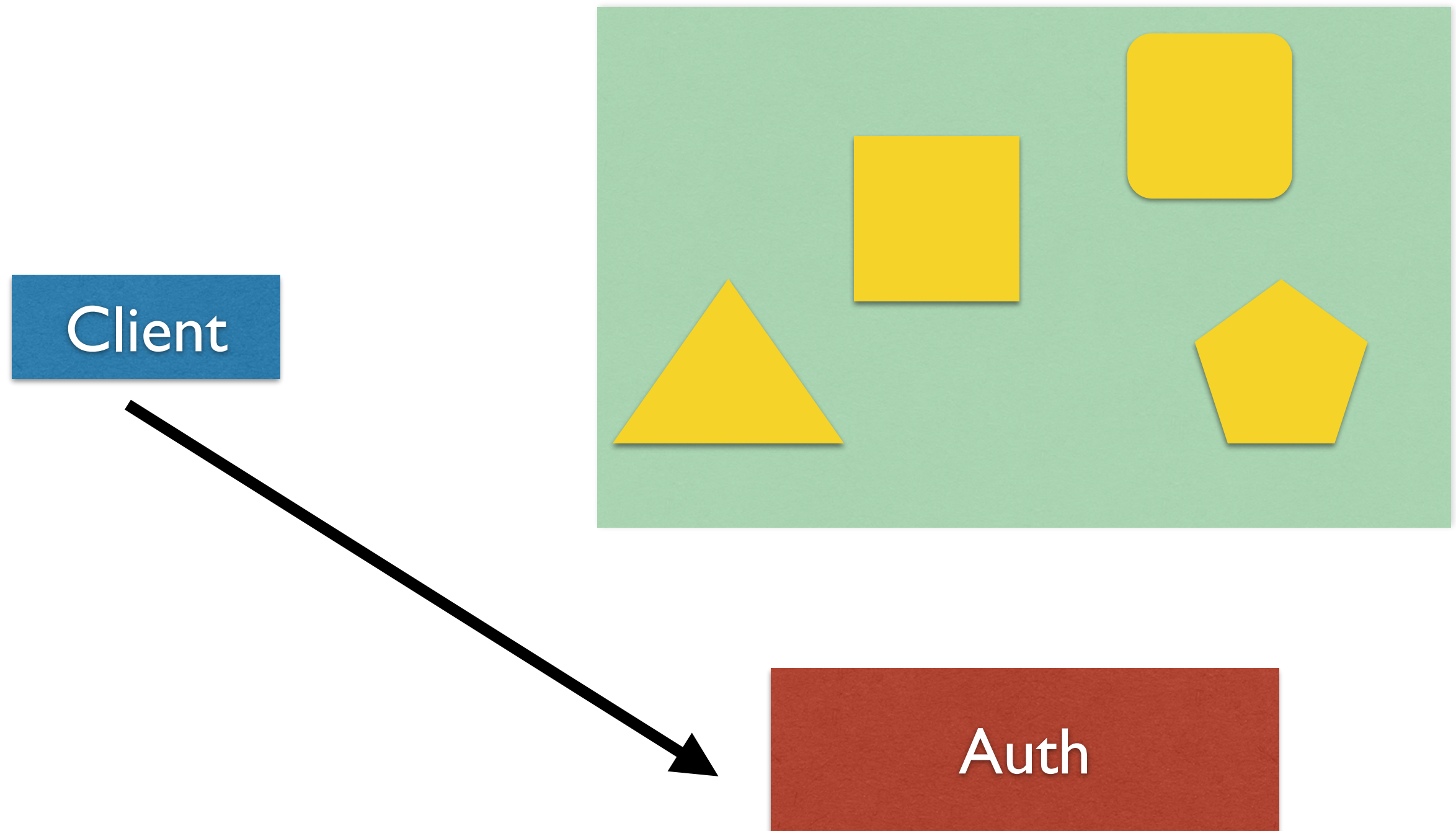
Example: Microservices



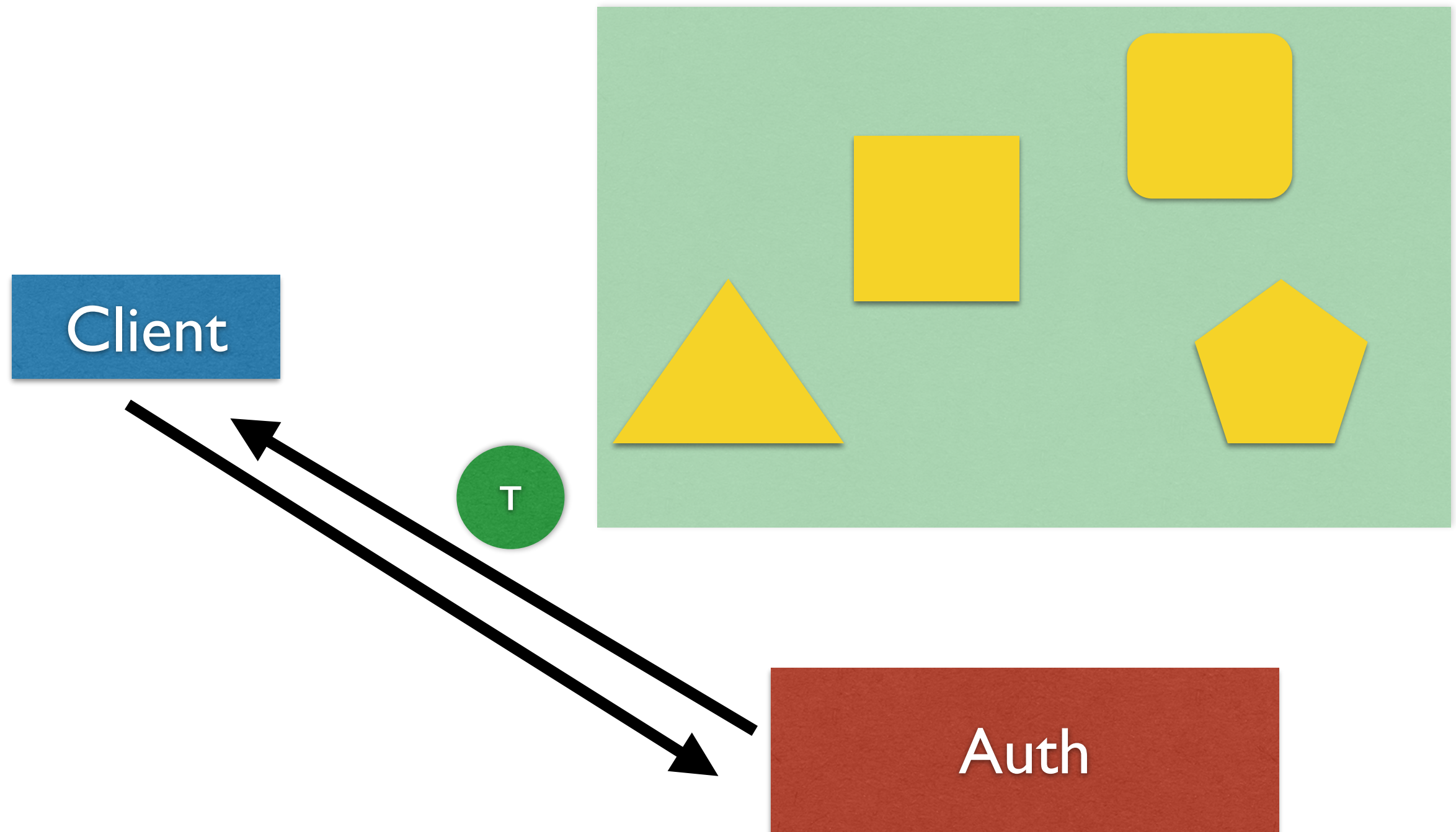
Example: Microservices



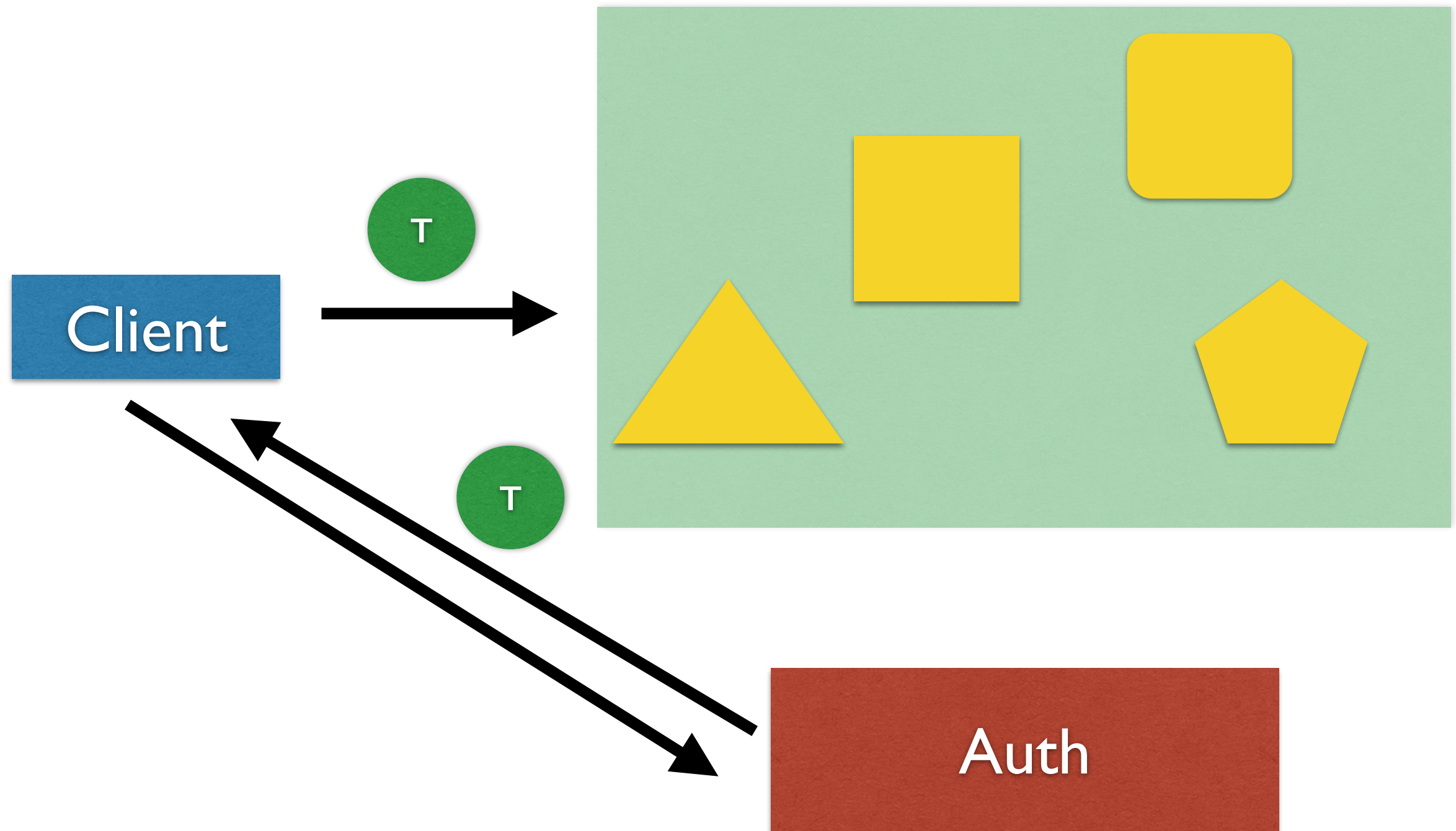
Example: Microservices



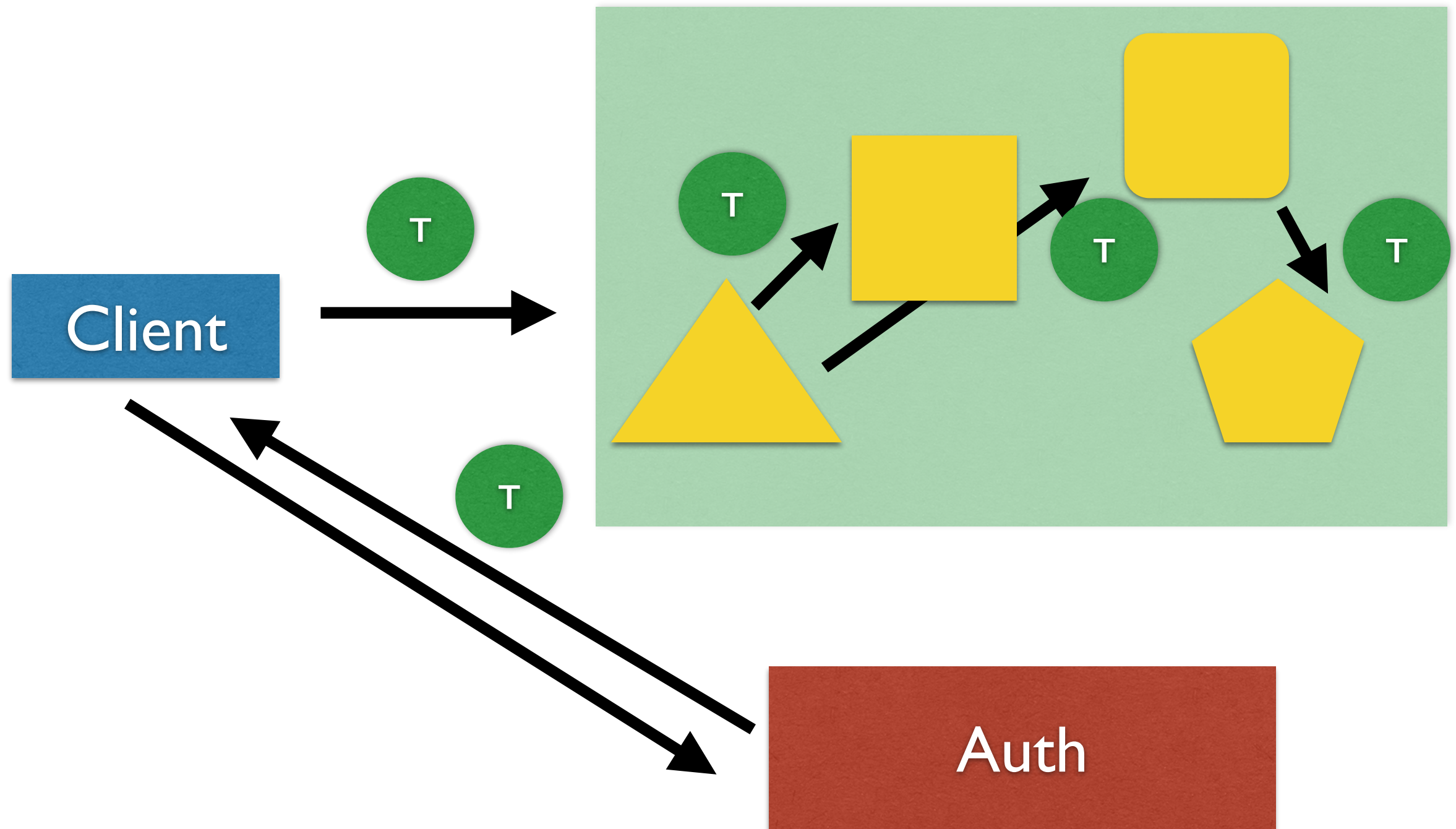
Example: Microservices



Example: Microservices



Example: Microservices



OAuth 2 for Single-Sign-On?

OAuth 2 for Single-Sign-On?

- › Token = “Authorization to access Resource”
Resource could be = Login to Service

OAuth 2 for Single-Sign-On?

- › Token = “Authorization to access Resource”
Resource could be = Login to Service
- › Missing:

When?

OAuth 2 for Single-Sign-On?

- › Token = “Authorization to access Resource”
Resource could be = Login to Service
- › Missing:

Who?

When?

OAuth 2 for Single-Sign-On?

- › Token = “Authorization to access Resource”
Resource could be = Login to Service
- › Missing:

Who?

When?

Where?

OAuth 2 for Single-Sign-On?

- › Token = “Authorization to access Resource”
Resource could be = Login to Service
- › Missing:

Who?

When?

How?

Where?

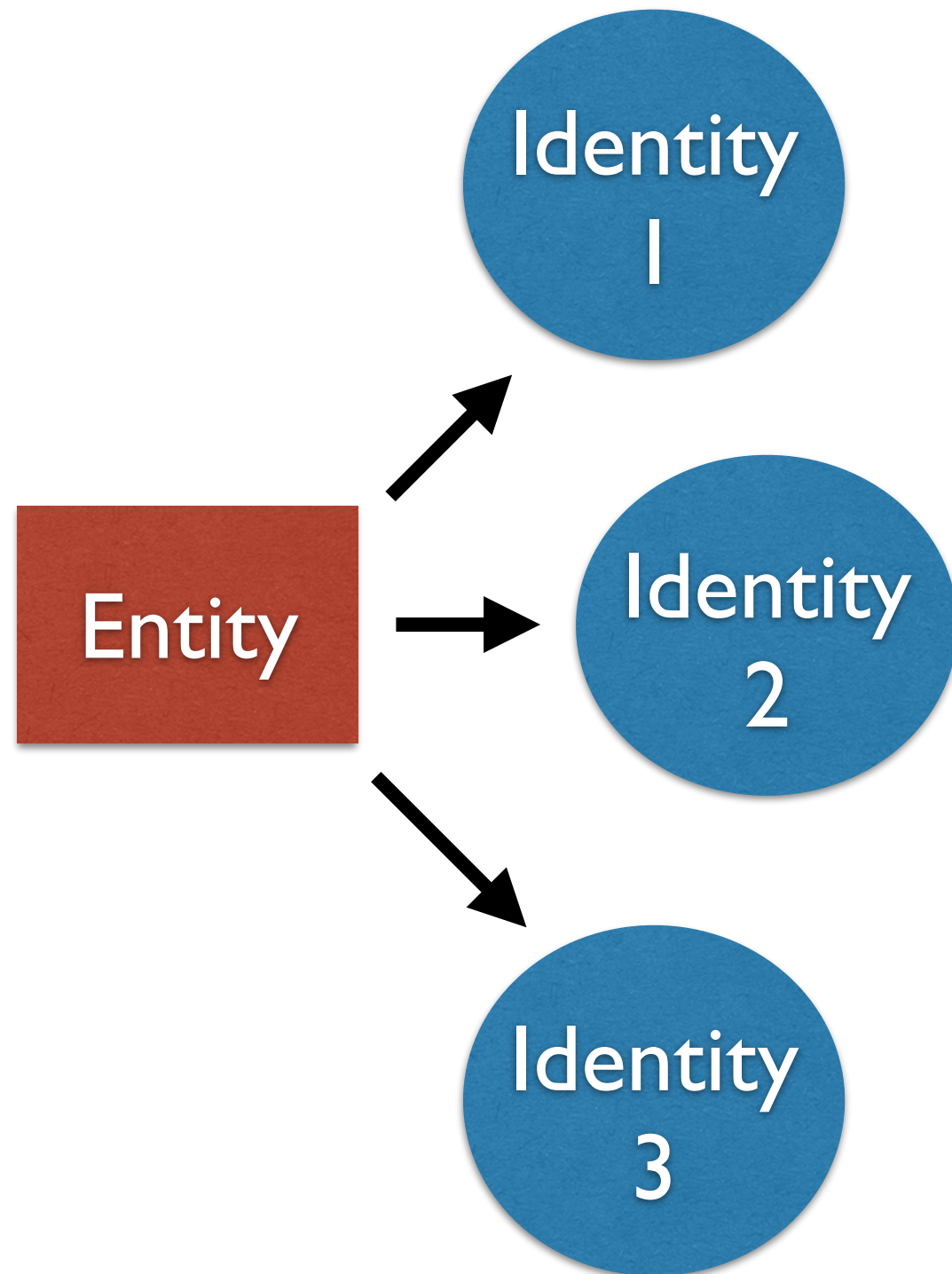
... ?

Identities

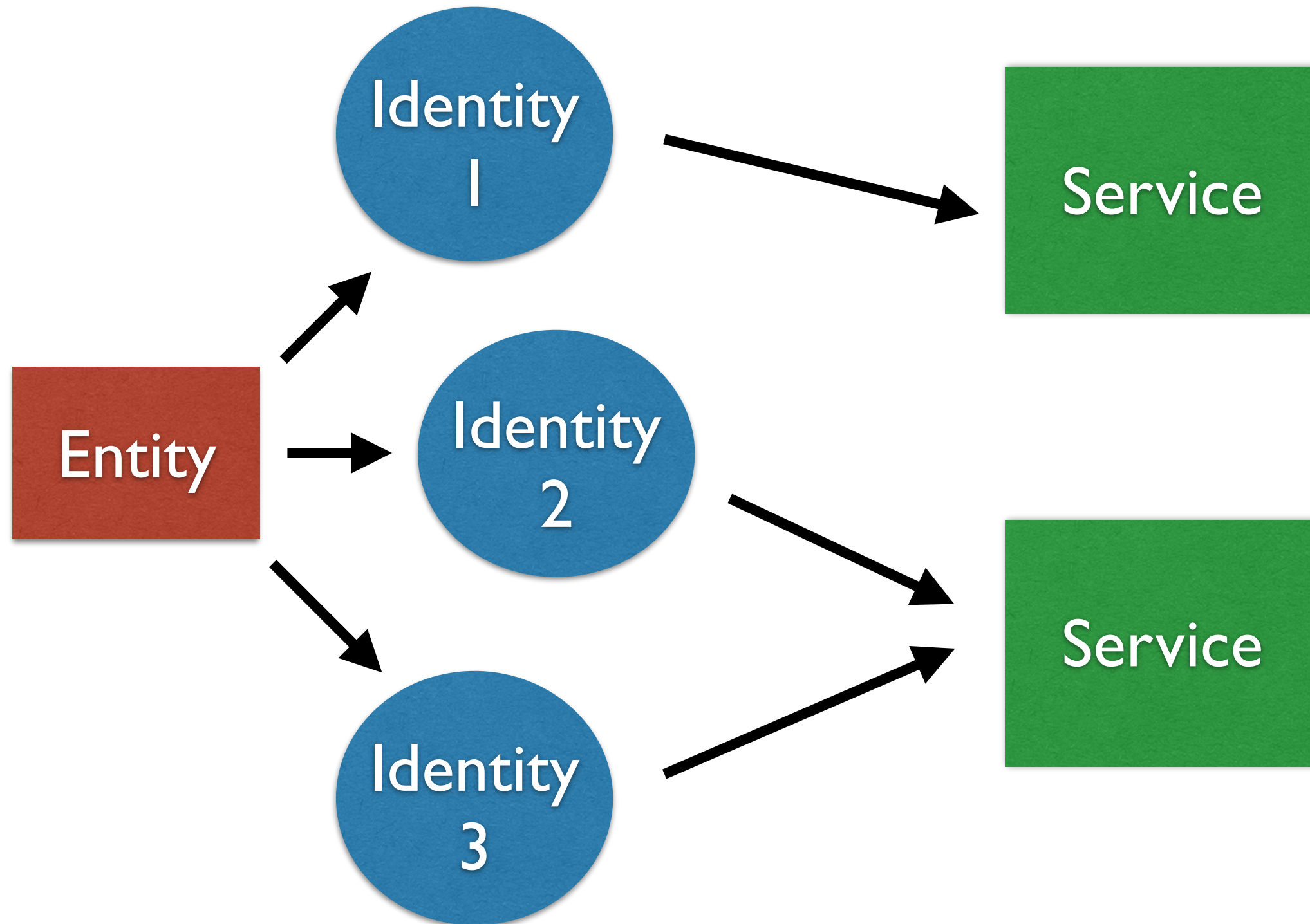


Entity

Identities



Identities



OpenID Connect

OpenID Connect

OpenID Connect

OpenID Connect

OAuth 2

OpenID Connect

OpenID Connect

OAuth 2

JOSE

OpenID Connect

Identity Layer

OpenID Connect

OAuth 2

JOSE

JWT Example

```
{
  "typ": "JWT",
  "alg": "HS256"
}{
  "iss": "http://server.example.com",
  "sub": "248289761001",
  "aud": "s6BhdRkqt3",
  "exp": 1311281970,
  "iat": 1311280970,
  "nonce": "n-0S6_WzA2Mj",
  "name": "Jane Doe",
  "given_name": "Jane",
  "family_name": "Doe",
  "gender": "female",
  "birthdate": "0000-10-31",
  "email": "janedoe@example.com",
  "email_verified": {"essential": true}
}
```

JWT Example

```
{  
  "typ": "JWT",  
  "alg": "HS256"  
}
```

JWT Header

```
{  
  "iss": "http://server.example.com",  
  "sub": "248289761001",  
  "aud": "s6BhdRkqt3",  
  "exp": 1311281970,  
  "iat": 1311280970,  
  "nonce": "n-0S6_WzA2Mj",  
  "name": "Jane Doe",  
  "given_name": "Jane",  
  "family_name": "Doe",  
  "gender": "female",  
  "birthdate": "0000-10-31",  
  "email": "janedoe@example.com",  
  "email_verified": {"essential": true}  
}
```

JWT Example

```
{  
  "typ": "JWT",  
  "alg": "HS256"
```

```
}{
```

```
  "iss": "http://server.example.com",  
  "sub": "248289761001",  
  "aud": "s6BhdRkqt3",  
  "exp": 1311281970,  
  "iat": 1311280970,  
  "nonce": "n-0S6_WzA2Mj",  
  "name": "Jane Doe",  
  "given_name": "Jane",  
  "family_name": "Doe",  
  "gender": "female",  
  "birthdate": "0000-10-31",  
  "email": "janedoe@example.com",  
  "email_verified": {"essential": true}
```

```
}
```

JWT Payload

JWT Example

Signature

```
{
  "typ": "JWT",
  "alg": "HS256"
}{
  "iss": "http://server.example.com",
  "sub": "248289761001",
  "aud": "s6BhdRkqt3",
  "exp": 1311281970,
  "iat": 1311280970,
  "nonce": "n-0S6_WzA2Mj",
  "name": "Jane Doe",
  "given_name": "Jane",
  "family_name": "Doe",
  "gender": "female",
  "birthdate": "0000-10-31",
  "email": "janedoe@example.com",
  "email_verified": {"essential": true}
}
```

JWT Example Encoded

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
eyJpc3MiOiJodHRwczovL2p3dC1pZHAuZXhhbXBsZS5jb20iLCJzdWIiOiJtYWlsdG86bWlrZUBleGFtcGxlLmNvbSI6Im5iZiI6MTQyOTY5MzY3MSwiZXhwIjoxNDI5Njk3Mjc5LCJpYXQiOiJlOTM2NzEsImp0aSI6Im1kMTIzNDU2IiwidHlwIjoiaHR0cHM6Ly9leGFtcGxlLmNvbS9yZWdpc3RlciJ9.
A3DVPjcIeQPGOkMcABwAe_8lWHvPG9dFhNyskwVfsxIL
t6SKtYGxYz0m7V-
DjzjYlXqzSwycwlRJBuYr_vdLRA9aoGsGQpP5-
SAiA5SdLMMk3MMZTioSHgZrC8TeZx8bJB1PzkSu91dJI
uzKI8PRPp3DH8Tum-XDsCmqu3_uI12633Mb1Bg4HKEz-
q2L2Y6k2Z1bqFxRn2GfV3ziQ8uqGOp3V_UlwvPccX8F3
m-
qe3MrF5aPSFGoU9bZDcQcBQ2ypGTluBNYnPzuMx9EdET
PJ0IxA1awgP74tFS27rt8KLUDnBvWVATNfYDFrqAcFCj
zk49znd4JLvNObbDebka3_g

JWT Example Encoded

Header

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.

~~cyJpc3MiOiJodHRwezoVL2p3dC1pZiIAuZXhhbXBsZS5j~~
b20iLCJzdWIiOiJtYWlscG86bWlrZUBleGFtcGxlLmNv
bSIscIm5iZiI6MTQyOTY5MzY3MSwiZXhwIjoxNDI5Njk3
MjcxCjYXQiOiJlOTM2NzEsImp0aSI6Im1kMTIz
NDU2IiwidHlwIjoiaHR0cHM6Ly9leGFtcGxlLmNvbS9y
ZWdpc3RlciJ9.

A3DVPjcIeQPGOkMcABwAe_8lWHvPG9dFhNyskwVfsxIL
t6SKtYGxYz0m7V-

DjzjYlXqzSwycw1RJBuYr vdLRA9aoGsGQpP5-

SAiA5SdLMMk3MMZTioSHgZrC8TeZx8bJB1PzkSu91dJI
uzKI8PRPp3DH8Tum-XDsCmqu3_uI12633Mb1Bg4HKEz-
q2L2Y6k2Z1bqFxRn2GfV3ziQ8uqGOp3V_UlwvPccX8F3
m-

```
qe3MrF5aPSFGou9bZDcQcBQ2ypGTluBNYnPzuMx9EdET
PJ0IxAlawgP74tFS27rt8KLUDnBvWVATNfYDFrqAcFCj
zk49znd4JLvNObbDebka3  g
```

JWT Example Encoded

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.

eyJpc3MiOiJodHRwczovL2p3dC1pZHAuZXhnbXBBSZS5jb20iLCJzdWIiOiJtYWlsdG86bWlrZUBleGFtcGxlLmNvbSIsIm5iZiI6MTQyOTY5MzY3MSwiZXhwIjoxNDI5Njk3MjcxCmJpYXQiOiJlOTM2NzEsImp0aSI6Im1kMTIzNDU2IiwidHlwIjoiaHR0cHM6Ly9leGFtcGxlLmNvbS9yZWdpc3RlciJ9.

Payload

A3DVPjcIeQPGOkMcABwAe_8lWHvPG9dFhNyskwVfsxIL
t6SKtYGxYz0m7V-

DjzjYlXqzSwycwlRJBuYr_vdLRA9aoGsGQpP5-
SAiA5SdLMMk3MMZTioSHgZrC8TeZx8bJB1PzkSu91dJI
uzKI8PRPp3DH8Tum-XDsCmqu3_uI12633Mb1Bg4HKEz-
q2L2Y6k2Z1bqFxRn2GfV3ziQ8uqGOp3V_UlwvPccX8F3
m-

qe3MrF5aPSFGoU9bZDcQcBQ2ypGTluBNYnPzuMx9EdET
PJ0IxA1awgP74tFS27rt8KLUDnBvWVATNfYDFrqAcFCj
zk49znd4JLvNObbDebka3_g

JWT Example Encoded

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
eyJpc3MiOiJodHRwczovL2p3dC1pZHAuZXhhbXBsZS5j
b20iLCJzdWIiOiJtYWlsdG86bWlrZUBleGFtcGxlLmNv
bSI6Im5iZiI6MTQyOTY5MzY3MSwiZXhwIjoxNDI5Njk3
MjcxCjE0Mjc2OTM2NzEsImp0aSI6Im1kMTIz
NDU2IiwidHlwIjoiaHR0cHM6Ly9leGFtcGxlLmNvbS9y
ZWdpc3RlciJ9.

A3DVPjcIeQPGOkMcABwAe_8lWHvPG9dFhNyskwVfsxIL
t6SKtYGxYz0m7V-
DjzjYlXqzSwycwlRJBuYr_vdLRA9aoGsGQpP5-
SAiA5SdLMMk3MMZTioSHgZrC8TeZx8bJB1PzkSu91dJI
uzKI8PRPp3DH8Tum-XDsCmqu3_uI12633Mb1Bg4HKEz-
q2L2Y6k2Z1bqFxRn2GfV3ziQ8uqGOp3V_UlwwPccX8F3
m-
qe3MrF5aPSFGoU9bZDcQcBQ2ypGTluBNYnPzuMx9EdET
PJ0IxA1awgP74tFS27rt8KLUDnBvWVATNfYDFrqAcFCj
zk49znd4JLvNObbDebka3_g

Signature

JWT Example Encoded

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.  
eyJpc3MiOiJodHRwczovL2p3dC1pZHAuZXhhbXBsZS5j  
b20iLCJzdWIiOiJtYWlsdG86bWlrZUBleGFtcGxlLmNv  
bSI6Im5iZiI6MTQyOTY5MzY3MSwiZXhwIjoxNDI5Njk3  
MjcxCmJpcyYXQiOiJlOTM2OTM2NzEsImp0aSI6Im1kMTIz  
NDU2IiwidHlwIjoiaHR0cHM6Ly9leGFtcGxlLmNvbS9y  
ZWdpc3RlciJ9.  
A3DVPjcIeQPGOkMcABwAe_8lWHvPG9dFhNyskwVfsxIL  
t6SKtYGxYz0m7V-  
DjzjYlXqzSwycw1RJBuYr_vdLRA9aoGsGQpP5-  
SAiA5SdLMMk3MMZTioSHgZrC8TeZx8bJB1PzkSu91dJI  
uzKI8PRPp3DH8Tum-XDsCmqu3_uI12633Mb1Bg4HKEz-  
q2L2Y6k2Z1bqFxRn2GfV3ziQ8uqGOp3V_UlwvPccX8F3  
m-  
qe3MrF5aPSFGoU9bZDcQcBQ2ypGTluBNYnPzuMx9EdET  
PJ0IxA1awgP74tFS27rt8KLUDnBvWVATNfYDFrqAcFCj  
zk49znd4JLvNObbDebka3_g
```

ID Token Example

```
{
  "typ": "JWT",
  "alg": "HS256"
}{
  "iss": "http://server.example.com",
  "sub": "248289761001",
  "aud": "s6BhdRkqt3",
  "exp": 1311281970,
  "iat": 1311280970,
  "nonce": "n-0S6_WzA2Mj",
  "name": "Jane Doe",
  "given_name": "Jane",
  "family_name": "Doe",
  "gender": "female",
  "birthdate": "0000-10-31",
  "email": "janedoe@example.com",
  "email_verified": {"essential": true}
}
```

ID Token Example

```
{  
  "typ": "JWT",  
  "alg": "HS256"  
}{  
  "iss": "http://server.example.com",  
  "sub": "248289761001",  
  "aud": "s6BhdRkqt3",  
  "exp": 1311281970,  
  "iat": 1311280970,  
  "nonce": "n-0S6_WzA2Mj",  
  "name": "Jane Doe",  
  "given_name": "Jane",  
  "family_name": "Doe",  
  "gender": "female",  
  "birthdate": "0000-10-31",  
  "email": "janedoe@example.com",  
  "email_verified": {"essential": true}  
}
```

Claim

ID Token Example

```
{  
  "typ": "JWT",  
  "alg": "HS256"  
}{  
  "iss": "http://server.example.com",  
  "sub": "248289761001",  
  "aud": "s6BhdRkqt3",  
  "exp": 1311281970,  
  "iat": 1311280970,  
  "nonce": "n-0S6_WzA2Mj",  
  "name": "Jane Doe",  
  "given_name": "Jane",  
  "family_name": "Doe",  
  "gender": "female",  
  "birthdate": "0000-10-31",  
  "email": "janedoe@example.com",  
  "email_verified": {"essential": true}  
}
```

**mandatory
Claims**

ID Token Example

```
{
  "typ": "JWT",
  "alg": "HS256"
}{
  "iss": "http://server.example.com",
  "sub": "248289761001",
  "aud": "s6BhdRkqt3",
  "exp": 1311281970,
  "iat": 1311280970,
  "nonce": "n-0S6_WzA2Mj",
  "name": "Jane Doe",
  "given_name": "Jane",
  "family_name": "Doe",
  "gender": "female",
  "birthdate": "0000-10-31",
  "email": "janedoe@example.com",
  "email_verified": {"essential": true}
}
```

scope E-Mail

ID Token Example

```
{
  "typ": "JWT",
  "alg": "HS256"
}{
  "iss": "http://server.example.com",
  "sub": "248289761001",
  "aud": "s6BhdRkqt3",
  "exp": 1311281970,
  "iat": 1311280970,
  "nonce": "n-0S6_WzA2Mj",
  "name": "Jane Doe",
  "given_name": "Jane",
  "family_name": "Doe",
  "gender": "female",
  "birthdate": "0000-10-31",
  "email": "janedoe@example.com",
  "email_verified": {"essential": true}
}
```

OpenID Connect Modules



Core

The diagram consists of a large, light orange rectangle with a thin, dark orange border. Inside this rectangle, centered, is a smaller, solid orange rectangle. The word "Core" is written in white, sans-serif font in the center of the smaller orange rectangle.

OpenID Connect Modules



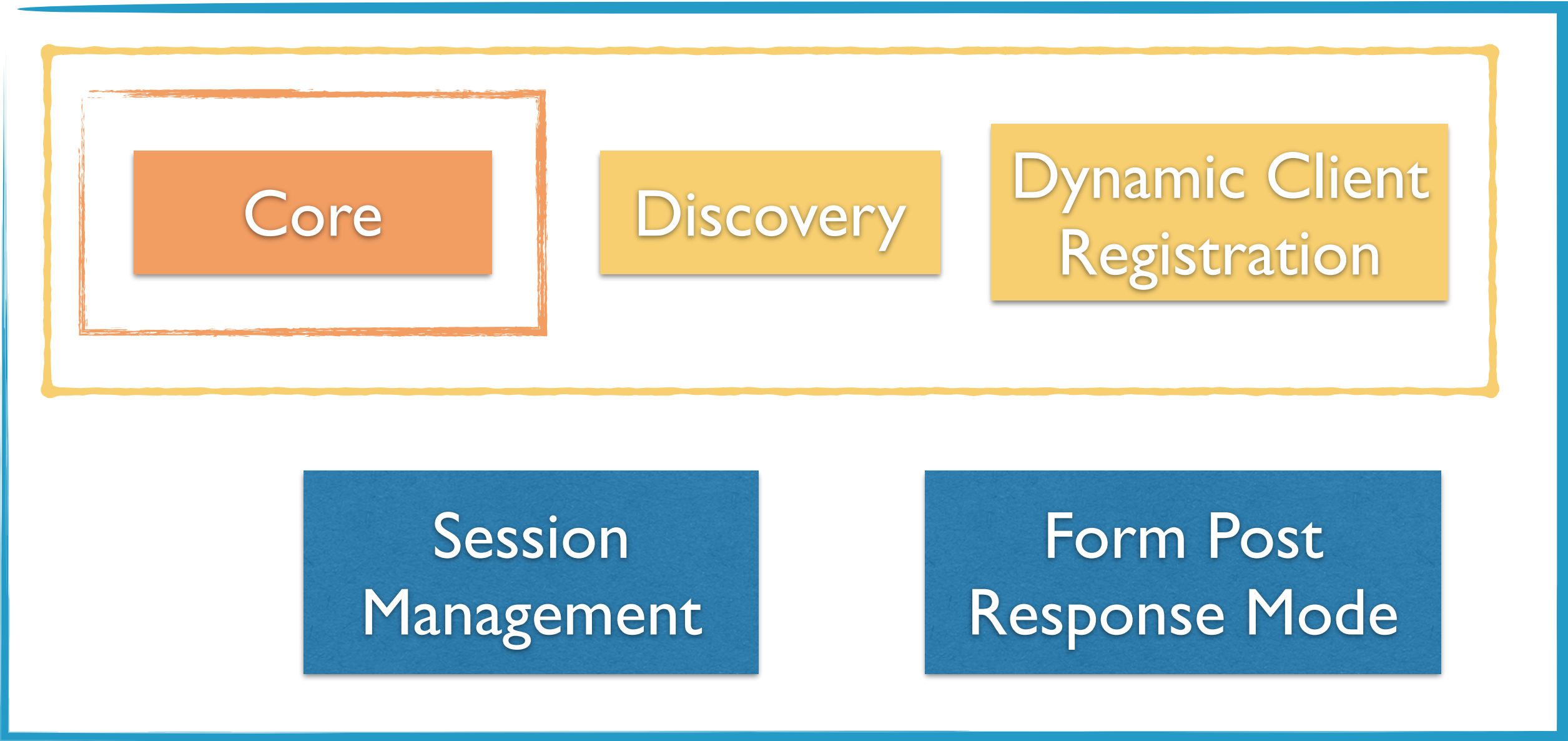
The diagram illustrates the OpenID Connect Modules. It features a large yellow rectangular frame with a hand-drawn style border. Inside this frame, on the left, is a smaller orange rectangle with a white border, containing the word 'Core'. To the right of this are two yellow rectangles. The first yellow rectangle contains the word 'Discovery'. The second yellow rectangle contains the words 'Dynamic Client' on the top line and 'Registration' on the bottom line. The 'Core' module is visually emphasized by being nested within an additional border.

Core

Discovery

Dynamic Client
Registration

OpenID Connect Modules



The diagram illustrates the OpenID Connect Modules. It features a large blue rectangular frame containing five colored boxes. At the top, a yellow brush-stroke border encloses three boxes: 'Core' (orange), 'Discovery' (yellow), and 'Dynamic Client Registration' (yellow). Below this, there are two blue boxes: 'Session Management' on the left and 'Form Post Response Mode' on the right. The 'Core' box is further enclosed by a smaller orange brush-stroke border.

Core

Discovery

Dynamic Client
Registration

Session
Management

Form Post
Response Mode

Enterprise Use Cases

Enterprise Use Cases

- › Securing internal APIs

Enterprise Use Cases

- › Securing internal APIs
- › Single-Sign-On

Enterprise Use Cases

- › Securing internal APIs
- › Single-Sign-On
- › Active Directory

Enterprise Use Cases

- › Securing internal APIs
- › Single-Sign-On
- › Active Directory
- › Mobile Clients

Enterprise Use Cases

- › Securing internal APIs
- › Single-Sign-On
- › Active Directory
- › Mobile Clients
- › White Label Software

Should I use it?

- › OpenID Connect Extensions
still partly Working Drafts

Should I use it?

- › OpenID Connect Extensions
still partly Working Drafts
- › JOSE still Working Draft

Should I use it?

- › OpenID Connect Extensions still partly Working Drafts
- › JOSE still Working Draft
- › OAuth Extensions: PoP, etc.

Should I use it?

- › OpenID Connect Extensions still partly Working Drafts
- › JOSE still Working Draft
- › OAuth Extensions: PoP, etc.
- › Implementation Complexity

Thank you!

Questions?

Comments?

Simon Kölsch  @simkoelsch

simon.koelsch@innoq.com

Christoph Iserlohn

christoph.iserlohn@innoq.com



www.innoq.com

innoQ Deutschland GmbH

Krischerstr. 100
40789 Monheim am Rhein
Germany
Phone: +49 2173 3366-0

Ohlauer Straße 43
10999 Berlin
Germany
Phone: +49 2173 3366-0

Robert-Bosch-Straße 7
64293 Darmstadt
Germany
Phone: +49 2173 3366-0

Radtkoferstraße 2
D-81373 München
Germany
Telefon +49 (0) 89 741185-270

innoQ Schweiz GmbH

Gewerbestr. 11
CH-6330 Cham
Switzerland
Phone: +41 41 743 0116