單一簽入解決方案

透過單一簽入解決方案整合地端應用系統與雲端服務認證

August 2023

鋐迪資訊

介紹

鍾迪 Andy Chung

- 現職：
  - 鋐迪資訊股份有限公司-資深技術顧問
- 學歷：
  - 加州長堤大學 CSULB 碩士
- 專案建置經歷 (20年)：
  - 政府、電信、航空、學校、海外：40個專案+
- 專業能力：
  - 身份管理系統規劃與建置
  - 帳號整合同步規劃與建置
  - 單一簽入系統規劃與建置
  - 特權管理系統規劃與建置
  - 多因素認證系統規劃與建置
  - Java / Linux Shell / SQL/ LDAP 開發
- 專業證照：
  - Access Manager Certified Professional Exam
  - Certified NetIQ Identity Manager Administrator (CNIMA)
  - Micro Focus Access Manager Specialization
  - IBM DB2 UDB Family Fundamentals
  - CompTIA LINUX+
  - Novell Certify Engineer (CLE)
  - Novell Certify Linux Professional (NCLP)
  - Security and Identity Management Technical Specialist
  - Novell Certify Linux Administrator (NCLA)
  - Access Management Technical Certification
  - Identity and Access Management Knowledge Check
  - Security Operations Knowledge Check

Single Sign On

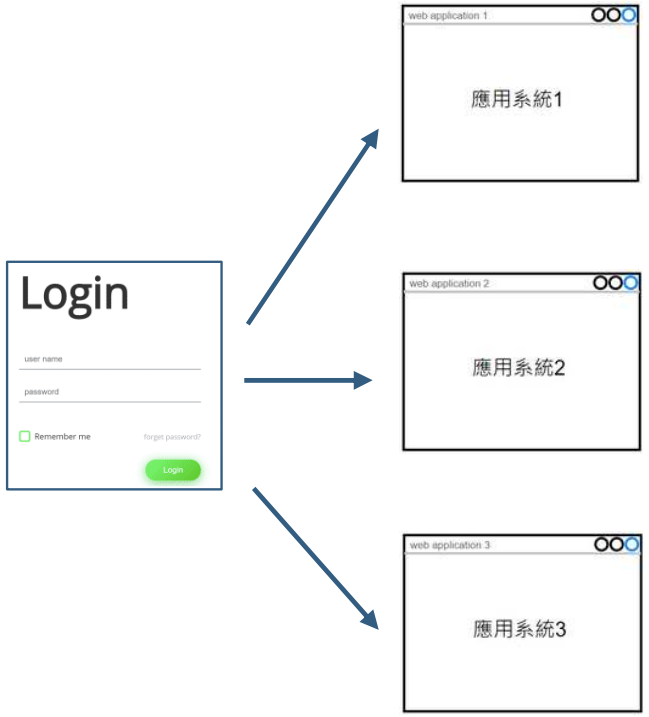
SAML 2.0

OpenID Connect

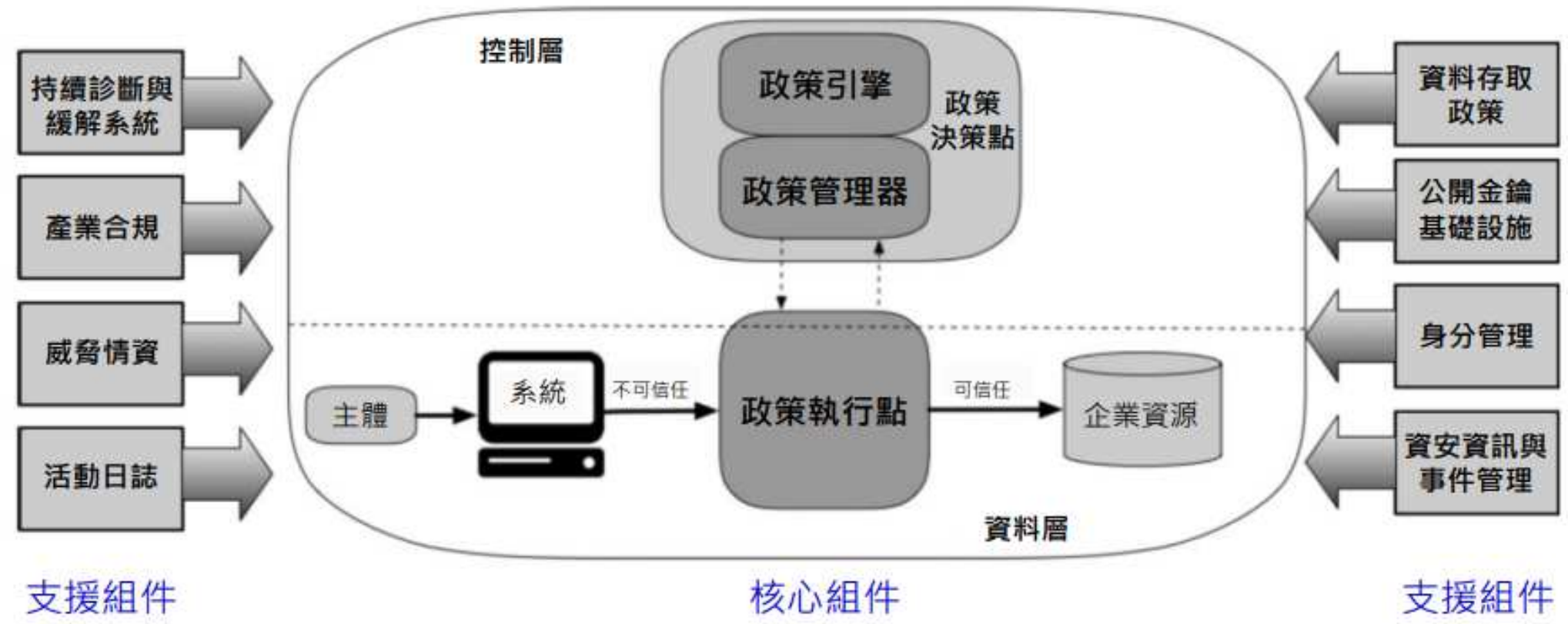
OAuth 2.0

Cookie

Session



國家資通安全研究院 -政府零信任架構說明文件



政府零信任架構說明文件 – 第6頁

NIST SP 800-207

零信任架構 – SSO建議的功能

Reverse Proxy

Identity Federation

User Authentication

Access Control

API Protection

Basic Identity Administration

NIST - 參與推動零信任的廠商

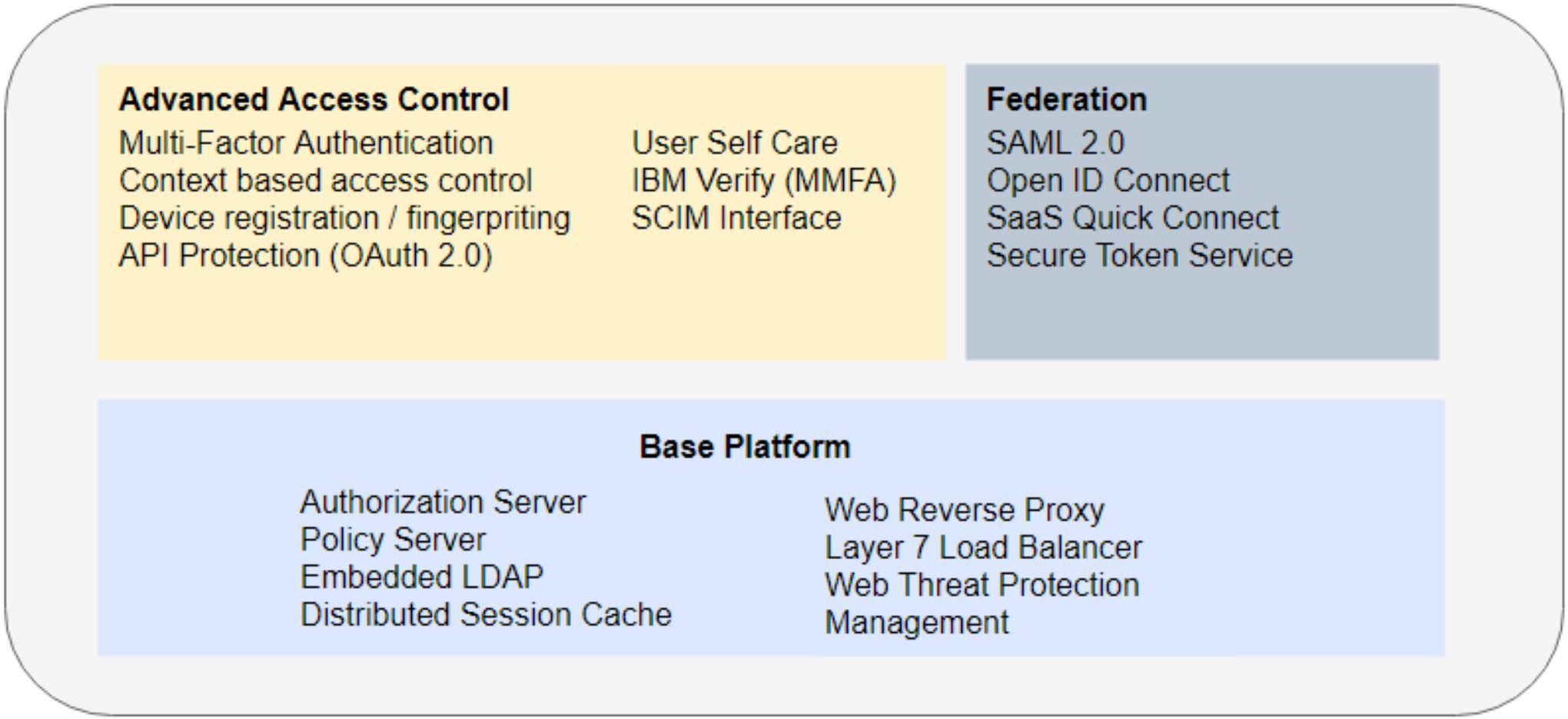
有Reverse Proxy功能，且可做到雲端與地端整合的產品有：

1. IBM Security Verify Access (since 2002)
2. NetIQ Access Manager (2009, 2011, 2014, 2023併購)
3. Broadcom Symantec SiteMinder (2020併購)

Technology Collaborators		
Appgate	IBM	Ping Identity
AWS	Ivanti	Radiant Logic
Broadcom Software	Lookout	SailPoint
Cisco	Mandiant	Tenable
DigiCert	Microsoft	Trellix
F5	Okta	VMware
Forescout	Palo Alto Networks	Zimperium
Google Cloud	PC Matic	Zscaler

NIST SP 1800- 35B : Implementing a Zero Trust Architecture – 第11頁

IBM Security Verify Access 模組

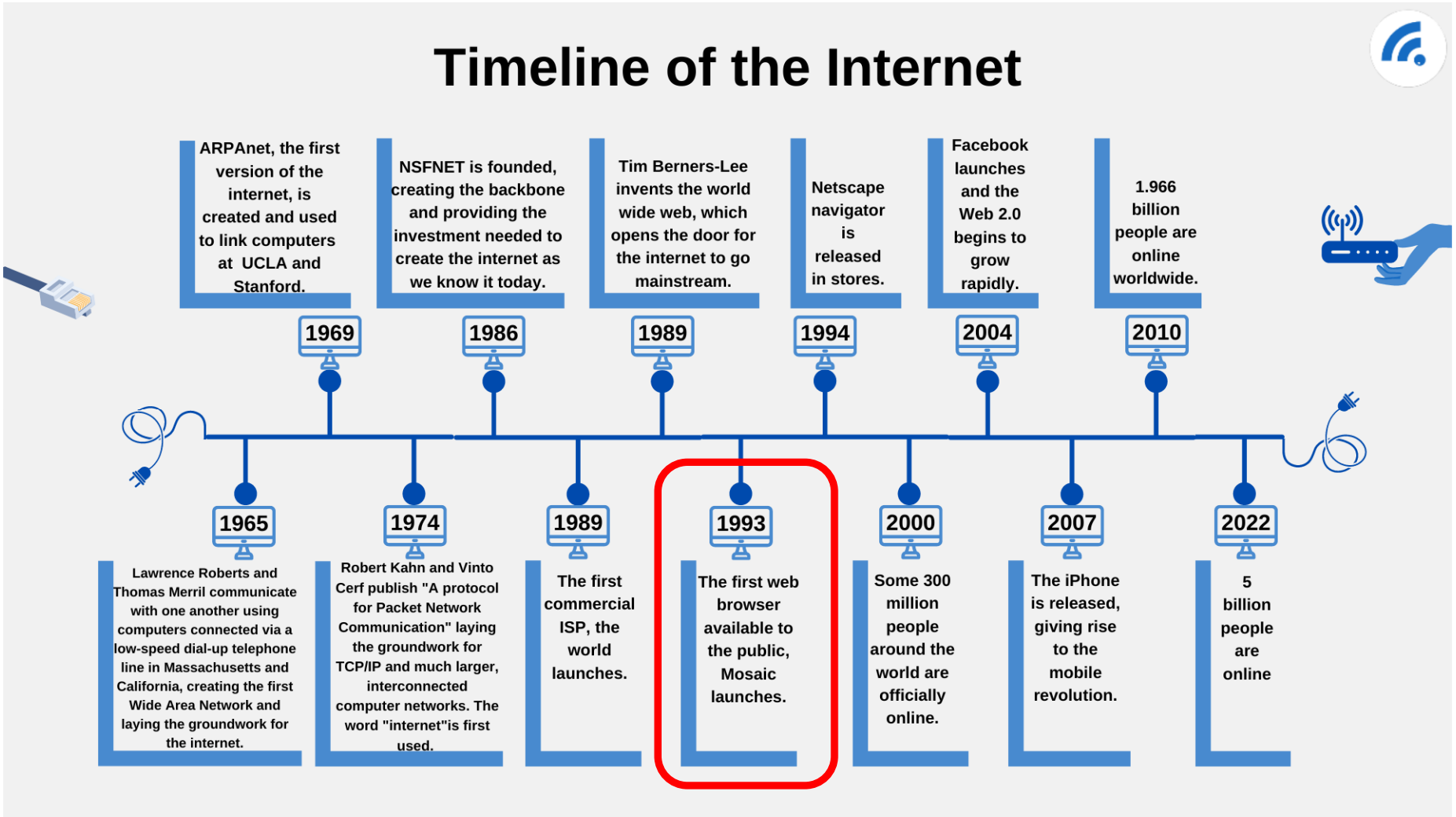


課程

- 傳統式單一簽入 (Reverse Proxy SSO)
- 現代式單一簽入 (OAuth 2.0, OIDC, SAML 2.0)
- 混和雲單一簽入 (Hybrid SSO)
- 單一簽入資安架構規劃 (Fits In Zero Trust)

傳統式單一簽入

Internet時間軸



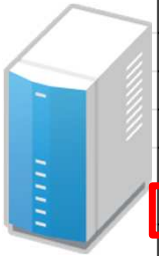
HTTP/1.1，1997 年RFC 2068，1999 年 RFC 2616 取代前一版。HTTP/2，2015 年RFC 7540

認證流程一

userid	loginname	userPwd	fullName	mail
11223344	andywu	**********	吳安迪	andywu@demo.cxm
11223345	nicklu	**********	呂尼克	nicklu@demo.cxm
11223346	jeffreyyang	**********	傑夫瑞	jeffreyyang@demo.cxm
11223347	kenchen	**********	陳肯尼	kenchen@demo.cxm
11223348	debbyweng	**********	翁黛比	debbyweng@demo.cxm
11223349	dorischung	**********	桃樂絲	dorischung@demo.cxm
11223350	andychung	**********	鍾安迪	andychung@demo.cxm
11223351	jamespeng	**********	詹姆士	jamespeng@demo.cxm
11223352	katychung	**********	鍾凱蒂	katychung@demo.cxm

認證流程二

當下使用的瀏覽器



userid	loginname	session id
11223344	andywu	
11223345	nicklu	
11223346	jeffreyyang	
11223347	kenchen	
11223348	debbyweng	
11223349	dorischung	
11223350	andychung	l5m2vnk3p07fg2nmnrsvslguch
11223351	jamespeng	
11223352	katychung	

Name	Value	Domain	Path	Expires / Max...
pwm_ver	5.8	.pwm-image.trendmic...	/	2023-07-23T...
pwm_topbar_ver	5.8	.pwm-image.trendmic...	/	2023-07-21T...
_ga_D6ZCVXWFJR	GS1.1.1674227155.1.1.1674227167.0.0.0	.trendmicro.com	/	2024-02-24T...
_ga	GA1.2.1710555739.1674227156	.trendmicro.com	/	2024-04-23T...
AMCV_3A9C6D6D567024D27F000101%...	1176715910%7CMCIDTS%7C19378%7CMCMID%...	.trendmicro.com	/	2024-02-24T...
SL_G_WPT_TO	zh	authenticationtest.com	/	Session
1P_JAR	2023-06-05-09	.gstatic.com	/	2023-07-05T...
SL_wptGlobTipTmp	1	authenticationtest.com	/	Session
SL_GWPT_Show_Hide_tmp	1	authenticationtest.com	/	Session
pwm_extensionframe_ver	5.8	.pwm-image.trendmic...	/	2023-07-21T...
PHPSESSID	l5m2vnk3p07fg2nmnrsvslguch	authenticationtest.com	/	Session

cookies

set cookie

l5m2vnk3p07fg2nmnrsvslguch

session id

認證流程三

核對session id

request

response

▼ Request Headers

:Authority:	www.office.com
:Method:	POST
:Path:	/landingv2
:Scheme:	https
Accept:	text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,a
Accept-Encoding:	gzip, deflate, br
Accept-Language:	en-US,en;q=0.9,zh-TW;q=0.8,zh;q=0.7
Cache-Control:	max-age=0
Content-Length:	2181
Content-Type:	application/x-www-form-urlencoded
Cookie:	OH.DCAffinity=OH-ejp; OH.FLID=aeb52e72-89e1-4567-9fca-daa4ff35ffaa; MSFPC=GUID=94c68e9ae1(UserIndex=H4sIAAAAAAAAAChWMSQ7CMAxFQ5kOgOAEbF3sxHHarlixY8cFMiBAFLKgvT%2BJrGc96du% gECkwYg3yEEk%2BbSq3evKxY%2B%2F%2B6bY4dqSlbSOEZmFSTtTph7%2BASmjDwCbAAAA; p.Display(.AspNetCore.OpenIdConnect.Nonce.UJoE5Xsbst7j7XpuzDuvNAzcWxOLaGqxReRWYS6dN5xNOpoGDW 1j8oUkpq4wq0UnkyxW9MU73UmVhTxu0j5bjyBegSuQplS5sM3PlaqO67vE7m1RiU7i6AP296sek-qf79vO .AspNetCore.Correlation.NI8EA9qufvWBkrF68ofYMPmhaMNaRYyCuhi3cBEbxMA=N; OH.SID=84eb426f
Origin:	https://login.live.com
Referer:	https://login.live.com/
Sec-Ch-Ua:	"Not.A/Brand";v="8", "Chromium";v="114", "Google Chrome";v="114"
Sec-Ch-Ua-Mobile:	?0
Sec-Ch-Ua-Platform:	"Windows"

▼ Response Headers

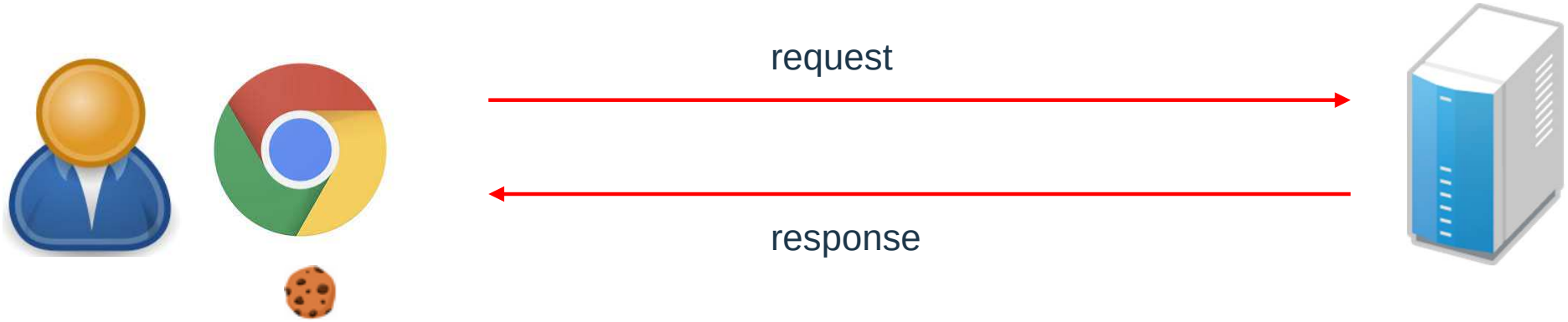
Cache-Control:	no-cache,no-store
Content-Length:	0
Date:	Thu, 29 Jun 2023 02:59:59 GMT
Expires:	Thu, 01 Jan 1970 00:00:00 GMT
Location:	/
Pragma:	no-cache
Referrer-Policy:	strict-origin-when-cross-origin
Request-Context:	appId=
Set-Cookie:	.AspNetCore.Correlation.NI8EA9qufvWBkrF68ofYMPmhaMNaRYyCuhi3cBEbxMA=; expires=Thu, 01 Jan 197
Set-Cookie:	.AspNetCore.OpenIdConnect.Nonce.UJoE5Xsbst7j7XpuzDuvNAzcWxOLaGqxReRWYS6dN5xNOpoGDWvaX aYZ1j8oUkpq4wq0UnkyxW9MU73UmVhTxu0j5bjyBegSuQplS5sM3PlaqO67vE7m1RiU7i6AP296sek-qf79vO 1970 00:00:00 GMT; path=/; secure; samesite=none; httponly
Set-Cookie:	AjaxSessionKey=ivWJvFh03LCESHNngjAFzcOaCy5AlgHISrT2qdmy3S0y4CzgaUyV0gaA6I9MIRwJR5IS0ZemC
Set-Cookie:	MUID=200809698A7E6A492AE51A568B046B57; path=/; secure; expires=Tue, 23-Jul-2024 03:00:00 GMT; d

Cookies

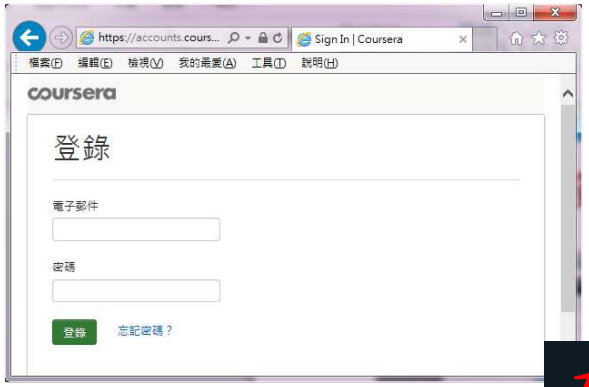
- Cookie是在瀏覽器儲存訊息的一種方式，伺服器可以回應瀏覽器set-cookie標頭，瀏覽器收到這一個標頭與數值後，會將之儲存為電腦上的一個檔案，這個檔案就稱之為Cookie。你可以設定給Cookie一個存活期限，保留一些有用的訊息在客戶端，如果關閉瀏覽器後，再度開啟瀏覽器並連接伺服器，而Cookie仍在有效期限中，瀏覽器會使用cookie標頭自動將Cookie發送給伺服器，伺服器就可以得知一些先前瀏覽器請求的相關訊息。
- Cookie的規範定義在RFC 2109: HTTP State Management Mechanism

Session

- Session是在伺服器端保存的一個數據結構，用來追蹤用戶的狀態，這一個數據可以是任何類別的數。每一個用戶都會有一個獨立的session。當程序需要為某個客戶端的請求創建一個session的時候，服務器首先檢查這個客戶端的請求裡是否已包含了一個session標識 - 稱為 session id，如果已包含一個session id則說明以前已經為此客戶端創建過session，服務器就按照session id把這個 session檢索出來使用。
- 在談論session機制的時候，常常聽到這樣一種誤解「只要關閉瀏覽器，session就消失了」。其實這並不完全正確，當使用者直接關閉瀏覽器時，其實後端伺服器並不知道，所以使用者的session id依舊存在伺服器端，直到session time out才會回收。

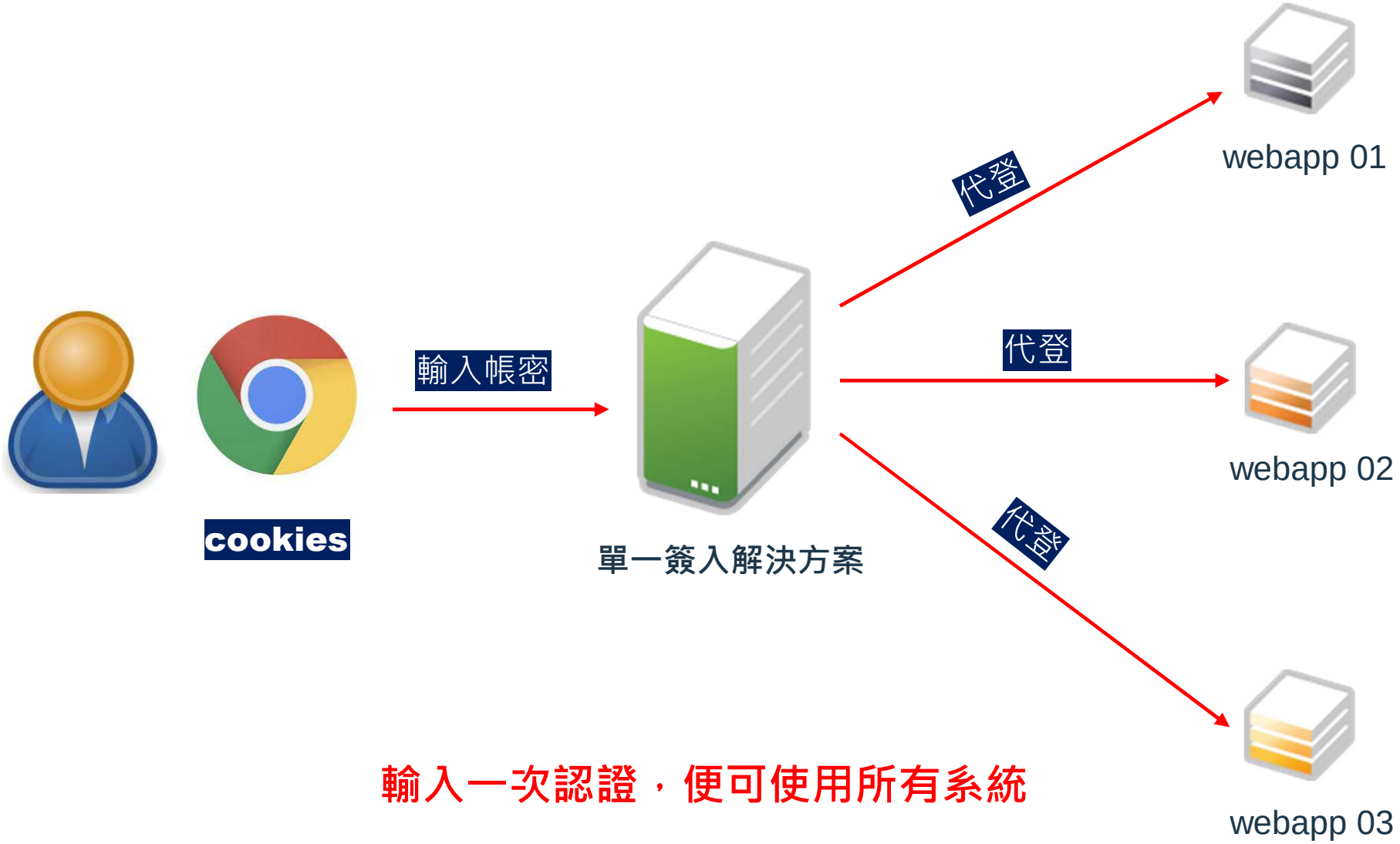


沒有單一簽入機制



登入六個網頁需要敲六次帳號與密碼

單一簽入



輸入一次認證，便可使用所有系統

1997 RFC 2109, HTTP State Management Mechanism

SSO代登技術 - Form Post

盡量不要用

SSO解決方案

透過http post方式將帳號密碼送給form action的url

webapp

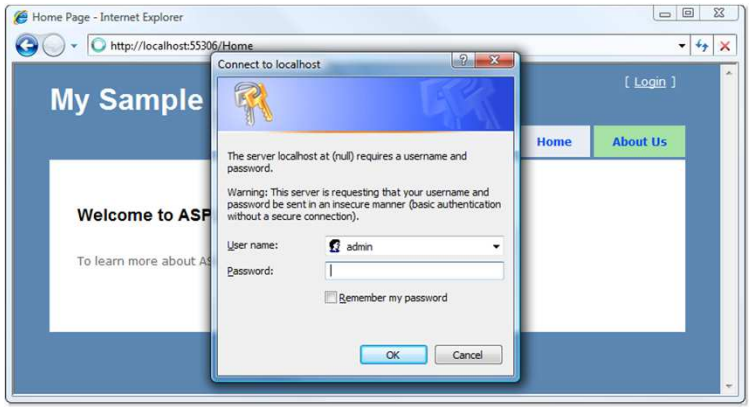
```
<form action="action_page.php" method="post">
  <div class="imgcontainer">
    <img src="img_avatar2.png" alt="Avatar" class="avatar">
  </div>

  <div class="container">
    <label for="uname"><b>Username</b></label>
    <input type="text" placeholder="Enter Username" name="uname" required>

    <label for="psw"><b>Password</b></label>
    <input type="password" placeholder="Enter Password" name="psw" required>

    <button type="submit">Login</button>
    <label>
      <input type="checkbox" checked="checked" name="remember"> Remember me
    </label>
  </div>

  <div class="container" style="background-color:#f1f1f1">
    <button type="button" class="cancelbtn">Cancel</button>
    <span class="psw">Forgot <a href="#">password?</a></span>
  </div>
</form>
```

SSO代登技術 - Basic Authentication

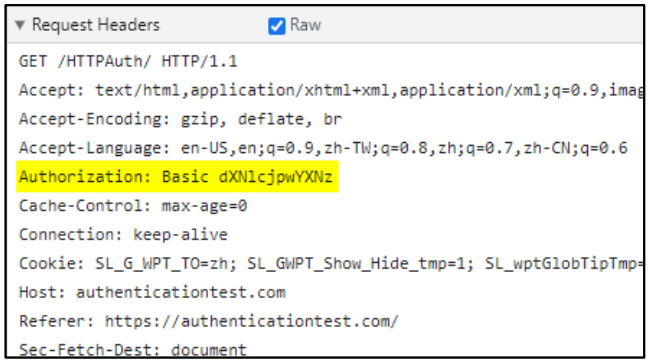
API認證使用

SSO解決方案

以B64編碼將帳號密碼編碼的值寫入到Authentication Header



webapp

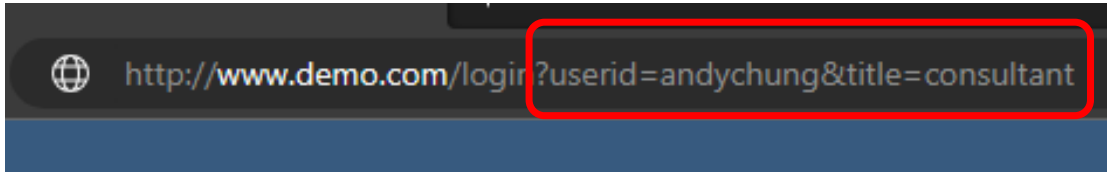


公式:
b64 (id:passwd)

SSO代登技術 - Query String

不建議使用

SSO解決方案



webapp

webapp程式去parsing url取得到userid

SSO代登技術 - Custom Header

HTTPS下使用

SSO解決方案

```
"x-forwarded-proto": "https",
"x-forwarded-port": "443",
"host": "postman-echo.com",
"x-amzn-trace-id": "Root=1-64bcbad0-4df9574968f0a65f465074e9",
1 "x-mysecretvalue": "12345",
2 "htic-mysecretvalue": "12345",
3 "mysecretvalue": "12345",
```

webapp

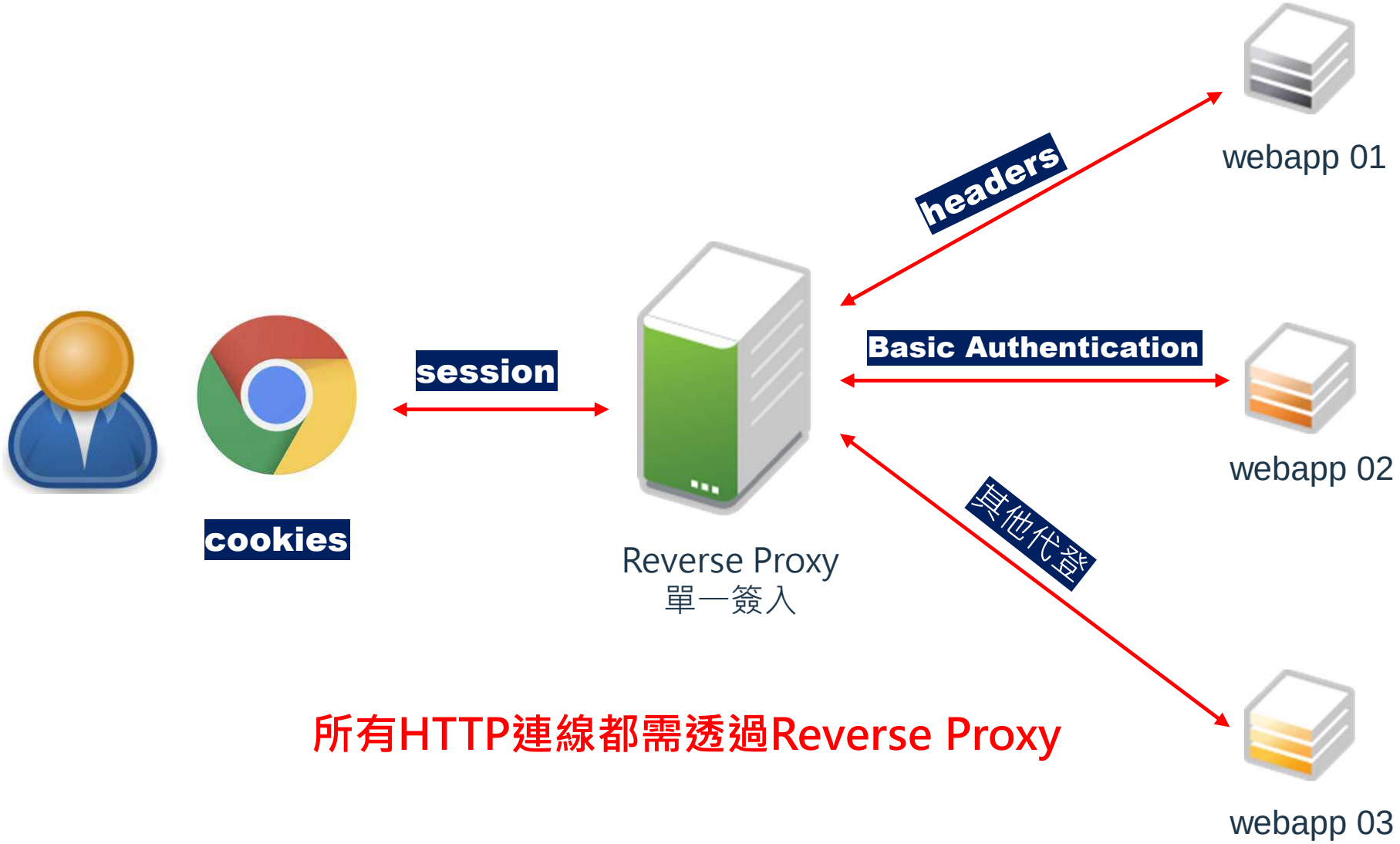
RFC 6648 :「*Creators of new parameters to be used in the context of application protocols SHOULD NOT prefix their parameter names with “X-” or similar constructs.*」

webapp程式去抓取headers資訊

TIP

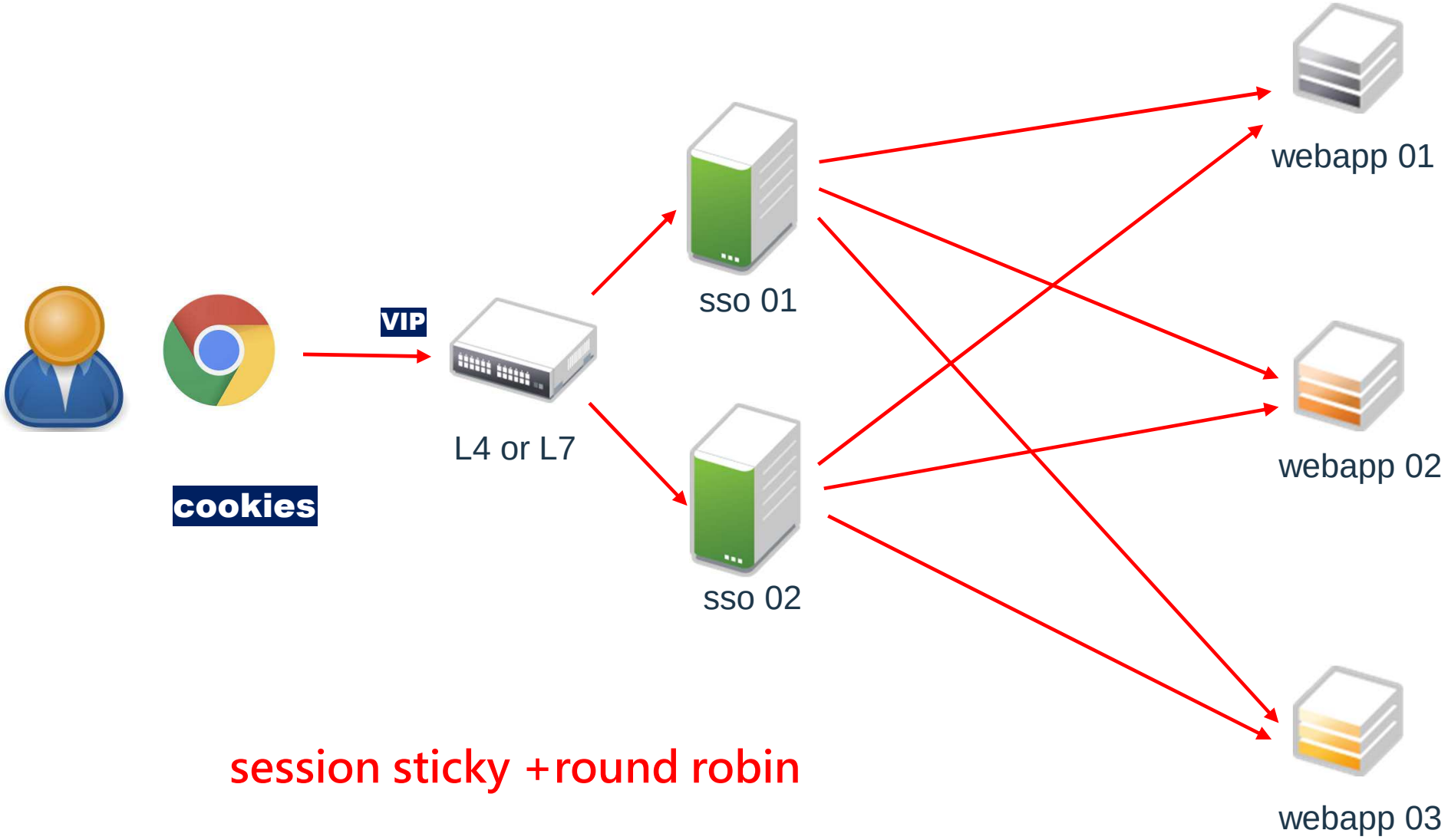
1. 沒有一定要在客製Header前面加X-
2. 客製header使用沒有定義的header名稱
3. 客製header可使用公司名稱+header名稱
4. 不建議移除現在既有的客製header名稱

單一簽入解決方案 – Reverse Proxy



所有HTTP連線都需透過Reverse Proxy

單一簽入 – 高可用性


VIP
L4 or L7
sso 01
sso 02
webapp 01
webapp 02
webapp 03
cookies
session sticky +round robin

使用Reverse Proxy SSO的好處

1. 集中式認證和授權
2. 增強的安全性
3. 合規性和審計
4. 負載平衡和性能優化
5. 未來擴展性
6. 簡化的訪問管理
7. 減少IT負擔
8. 單一入口點

竊取與偽造cookie資訊

- XSS: cross site scripting 跨網站指令碼，通常指的是通過利用網頁開發時留下的漏洞，通過巧妙的方法注入惡意指令代碼到網頁，使用者載入並執行攻擊者惡意製造的網頁程式。這些惡意網頁程式通常是JavaScript，但實際上也可以包括Java，VBScript，ActiveX，Flash或者甚至是普通的HTML。攻擊成功後，攻擊者可能得到更高的權限（如執行一些操作）、私密網頁內容、對談和cookie等各種內容。

- CSRF: cross site request forgery 跨站請求偽造，也被稱為 one-click attack 或者 session riding，通常縮寫為 CSRF 或者 XSRF， 是一種挾制使用者在當前已登入的Web應用程式上執行非本意的操作的攻擊方法。

保護與強化Cookie

- http only: 防止client side JavaScript read cookie
- Secure cookie: cookie資訊必須使用https連線環境
- Same site:只要連上的網站其 Domain 跟 Path 與 Cookie 一致，就會被視為同源。
  - Samesite=strict : 同一個domain網站網頁都可以收到這一個domain cookie，如連到別的domain，之前的cookie是不會送出。
  - Samesite=lax：放鬆一些限制允許在別的domain也可收到之前的cookie
    - 使用者自行在網址列輸入網址
    - 以Get的方式送出表單 如：<form method="GET" …>
    - 點擊連結 如：
    - 使用link 如： <link ref="prerender" href="…" />
  - Samesite=none;secure：允許第三方使用cookie但限定在https連線下。

Cookie防篡改

服務器可以為每個Cookie產生一個簽名，由於用戶篡改Cookie後無法生成對應的簽名， 服務器便可以得知用戶對Cookie進行了篡改。一個簡單的校驗過程可能是這樣的：

1. 服務端提供一個簽名生成算法secret
2. 根據方法生成簽名secret(wall)=34Yult8i
3. 將生成的簽名放入對應的Cookie項username=wall|34Yult8i。其中，內容和簽名用|隔開。
4. 服務端根據接收到的內容和簽名，校驗內容是否被篡改。

現代式單一簽入

Session認證的挑戰

- 單一簽入系統loading重
- Basic Authentication 帳號密碼安全性問題
- 整合上沒有統一規範
- 帳號密碼竊取問題
- Session ID被竊取的問題
- Fail-Over問題

OAuth 2.0

傳統認證的挑戰

在傳統的 Client-Server 架構裡， Client要拿取受保護的資源 (Protected Resource) 的時候，要向 Server 出示使用者 (Resource Owner) 的帳號密碼才行。為了讓第三方應用程式也可以拿到這些 Resources ，則 Resource Owner 要把帳號密碼給這個第三方程式，這樣子就會有以下的問題及限制：

1. 第三方程式可以偷偷儲存使用者的帳號密碼。
2. Server 必須支援密碼認證，即使密碼有天生的資訊安全上的弱點。
3. 第三方程式會得到幾乎完整的權限，沒辦法限制第三方程式可以拿取 Resource 的時效，以及可以存取的範圍。
4. 無法只撤回單一個第三方程式的存取權，而且必須要改密碼才能撤回。
5. 任何第三方程式被破解，就會導致使用該密碼的所有資料被破解。

OAuth 2.0 介紹

- OAuth 1.0 (2009), OAuth 2.0 (2013), OAuth 2.1 draft (2020-2023)

- 開放授權（OAuth）是一個開放標準，允許使用者讓第三方應用存取該使用者在某一網站上儲存的私密的資源（如相片，影片，聯絡人列表），而無需將使用者名稱和密碼提供給第三方應用。

- OAuth允許使用者提供一個權杖，而不是使用者名稱和密碼來存取他們存放在特定服務提供者的資料。每一個權杖授權一個特定的網站（例如，影片編輯網站)在特定的時段（例如，接下來的2小時內）內存取特定的資源（例如僅僅是某一相簿中的影片）。這樣，OAuth讓使用者可以授權第三方網站存取他們儲存在另外服務提供者的某些特定資訊，而非所有內容。

OAuth 2.0 特性

- OAuth 2.0是一個開放式的授權架構
- OAuth 2.0透過HTTP通訊協定，授權給Devices, APIs, Servers與Applications
- OAuth 2.0主要的功能是用Access Token來存取保護資源
- 建議全程使用 HTTPS with TLS 1.2 (或更高)
- User-Agent必須支援HTTP Redirection (HTTP 302 Status Code)
- RFC 6749 是OAuth 2.0 技術規範

OAuth 2.0 角色定義

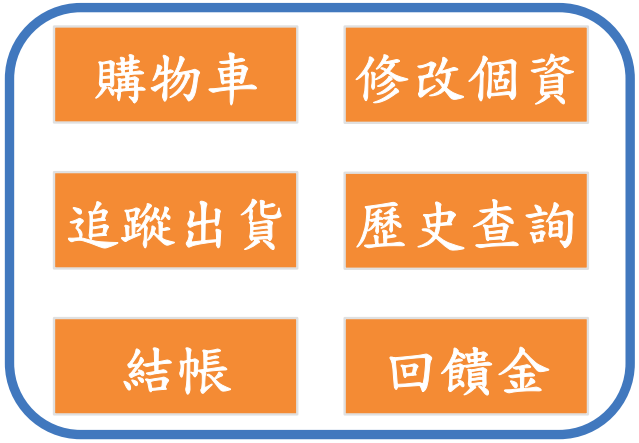
Resource Owner

使用者

USER AGENT

瀏覽器

CLIENT



購物平台系統選單

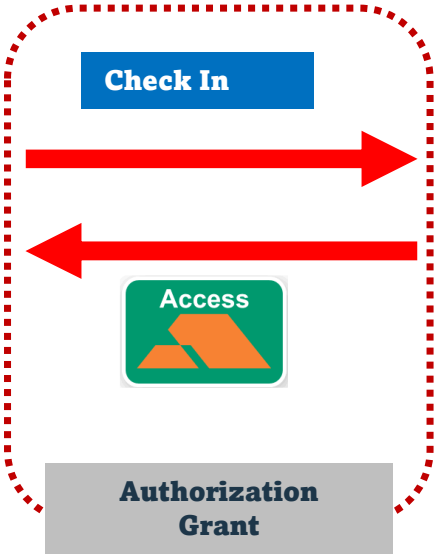
Source Server

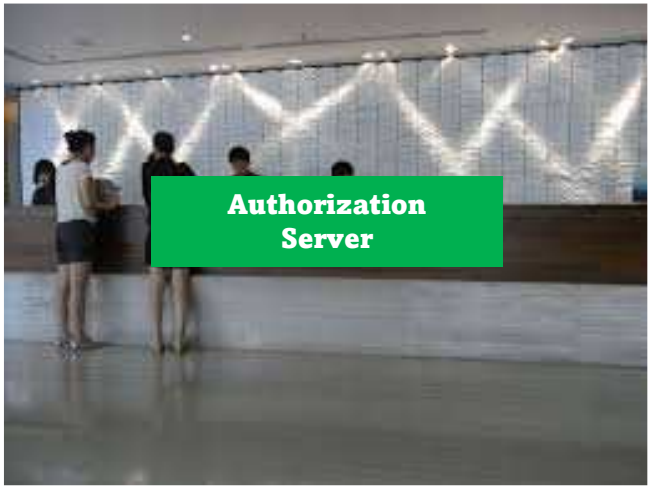
API服務

Authorization Server

認證與授權伺服器

OAuth 2.0 角色範例





SCOPE
開門

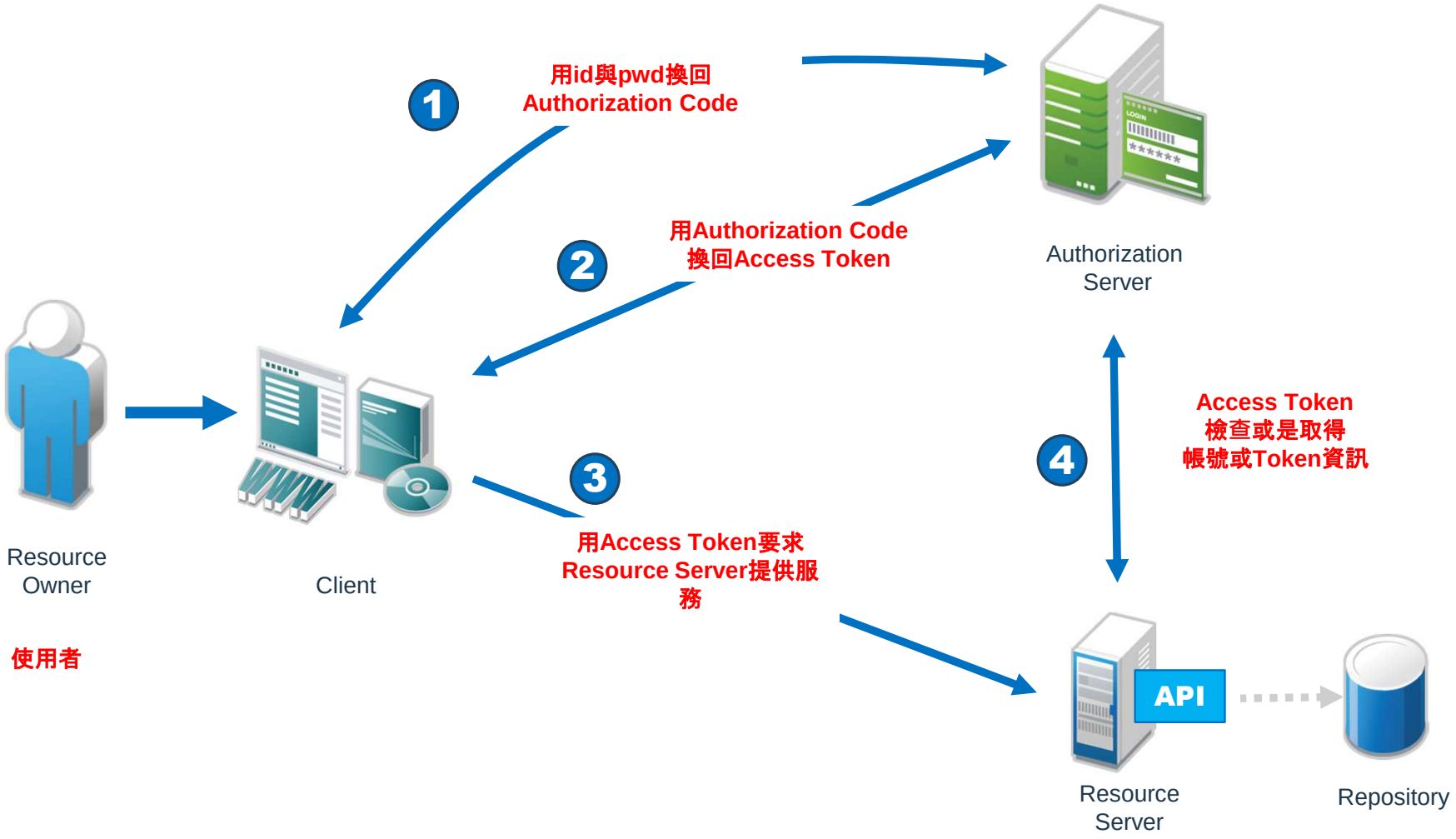
Endpoint

Endpoint name	Definition
Authorization endpoint	the resource owner grants authorization to the OAuth client to access the protected resource
Token endpoint	OAuth client exchanges an authorization grant for an access token and an optional refresh token
Logout endpoint	end a session by revoking an Access Token
Introspect endpoint	It can be used to validate reference tokens
Revocation endpoint	A URL where you can revoke OAuth tokens issued to a client
Metadata endpoint	Final portion of URL is a path parameter that is the name of your API Protection definition
Userinfo Endpoint	an OAuth 2.0 protected resource that returns claims about the authenticated end-user.

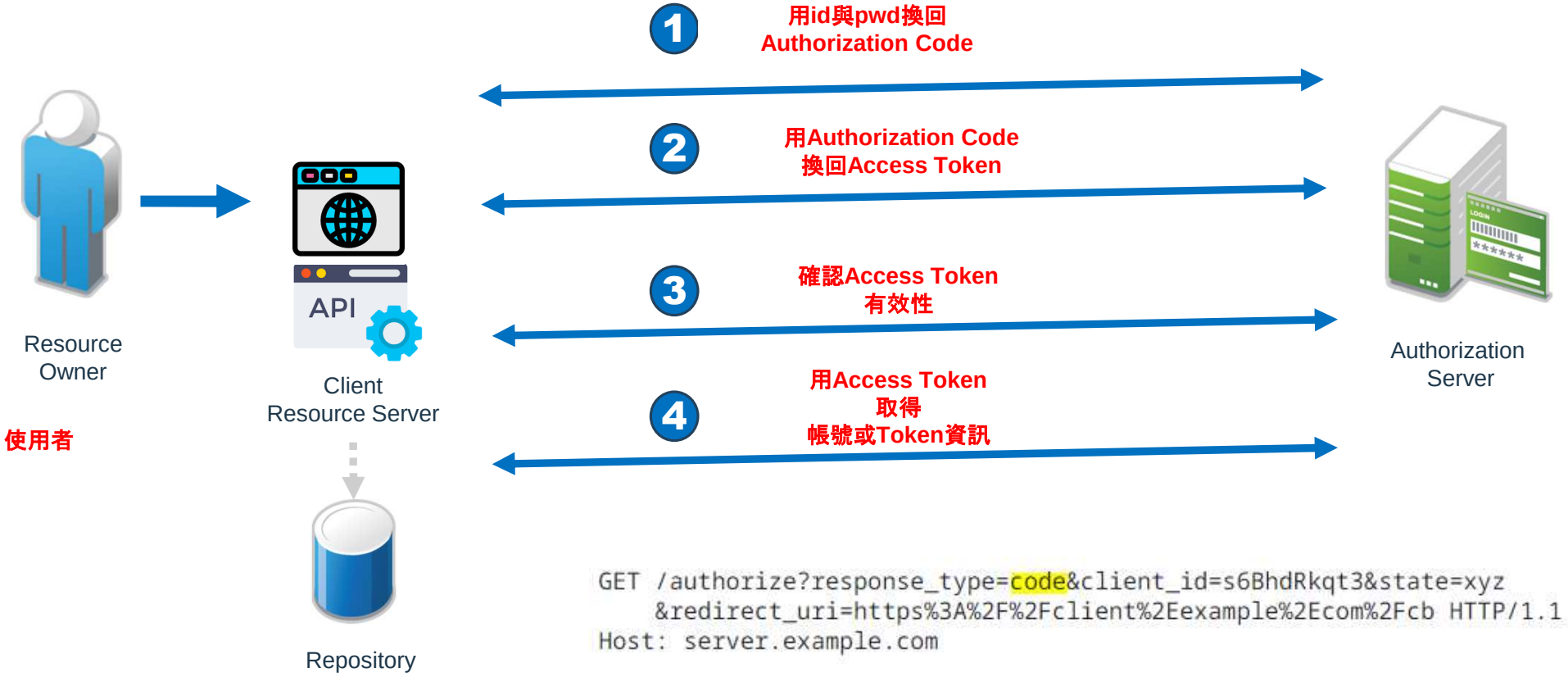
Grant Type for OAuth 2.0

- **Authorization Code**
  - 最複雜但最安全，透過Authorization Code換取Access Token
  - https do post and redirect
- **Implicit**
  - 相對簡單，但Access Token放在URI，容易被取得
  - Single Page Application、JavaScript
- **Resource Owner Password Credentials**
  - 直接將使用者帳號密碼給Client到Authorization Server取得Access Token
- **Client Credentials**
  - API calls 使用

Authorization Code Grant



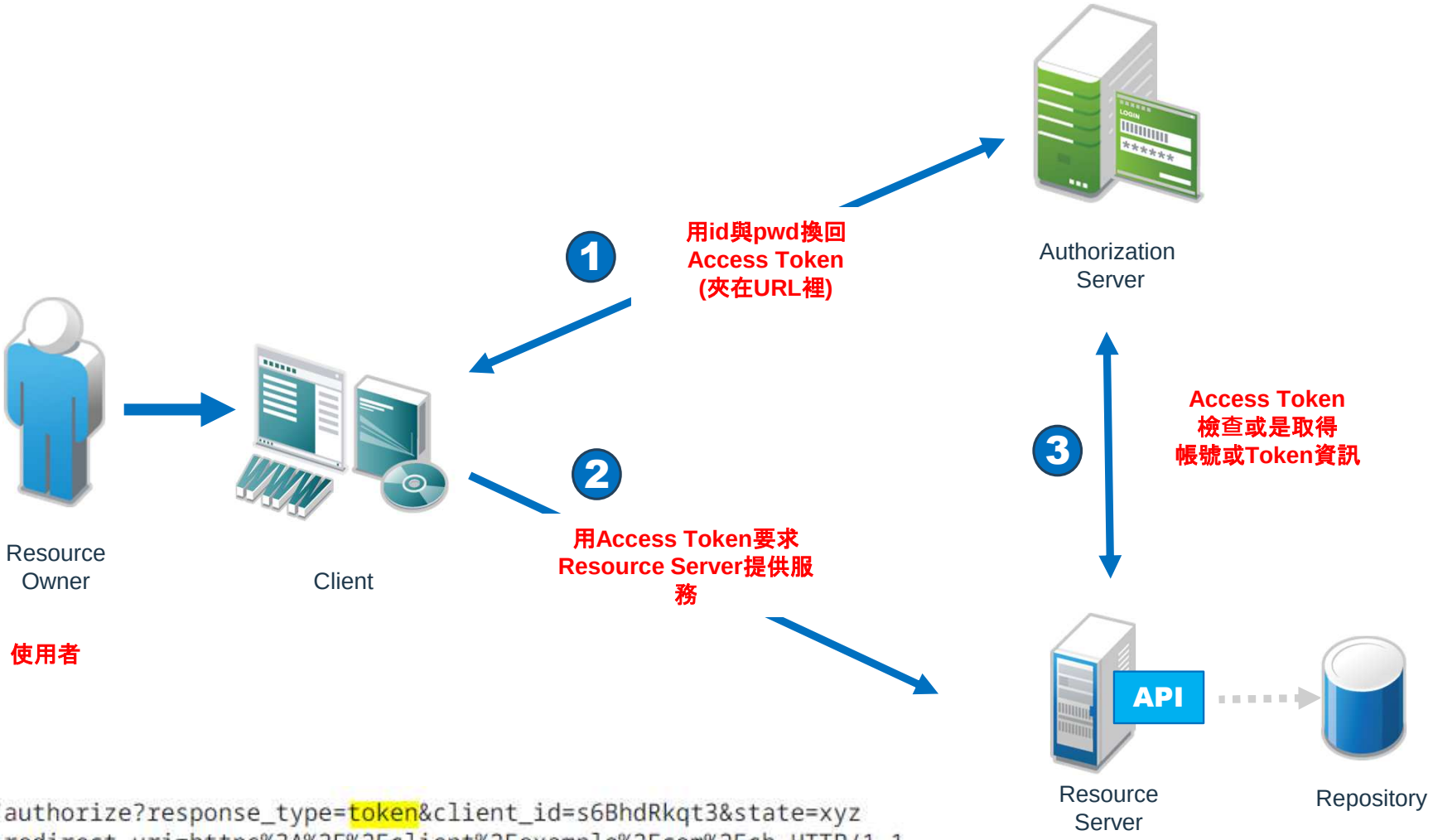
Authorization Code Grant (二)




```
GET /authorize?response_type=code&client_id=s6BhdRkqt3&state=xyz
    &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1
Host: server.example.com
```

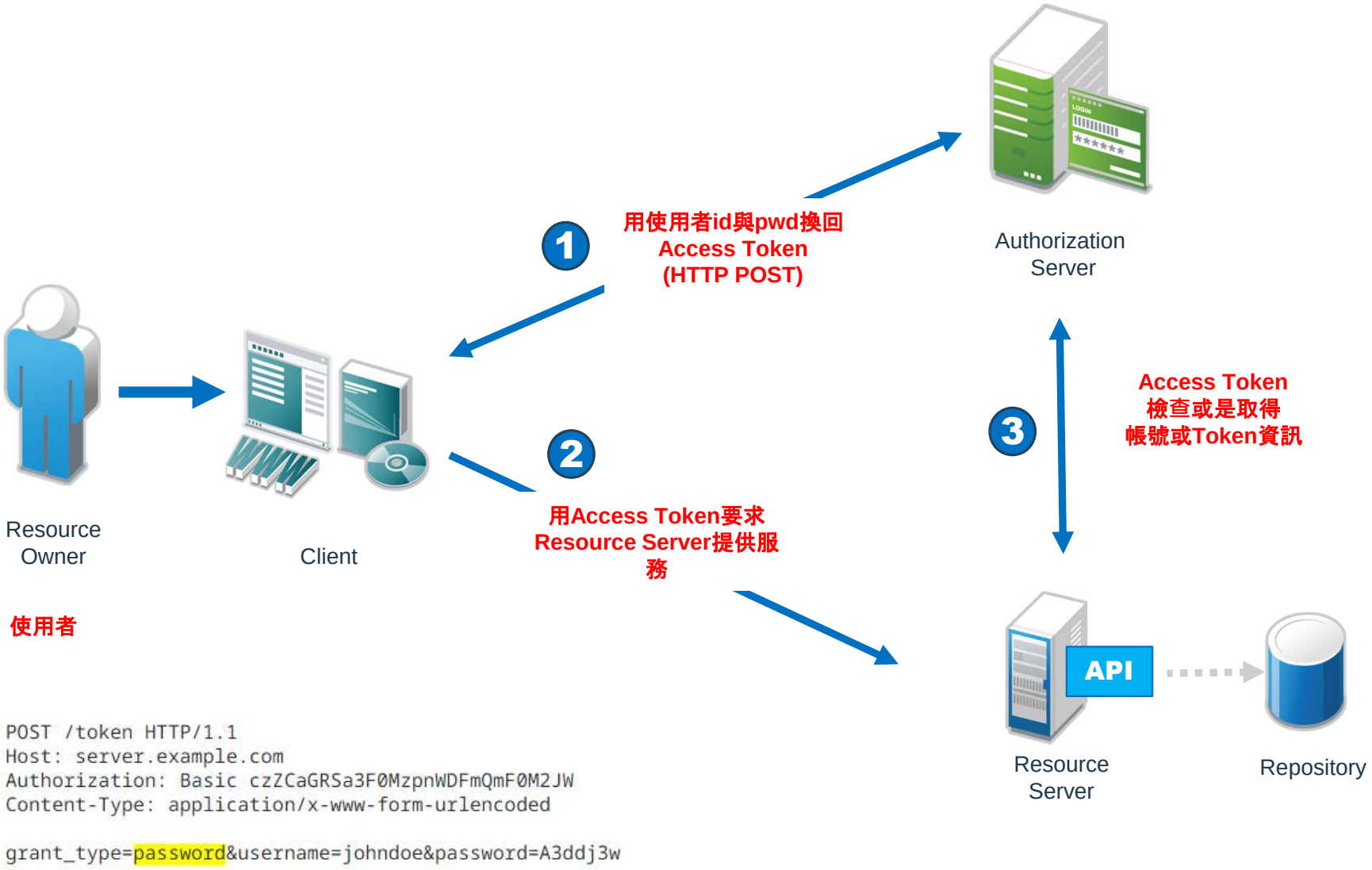
```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSbIA
&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
```

Implicit Grant



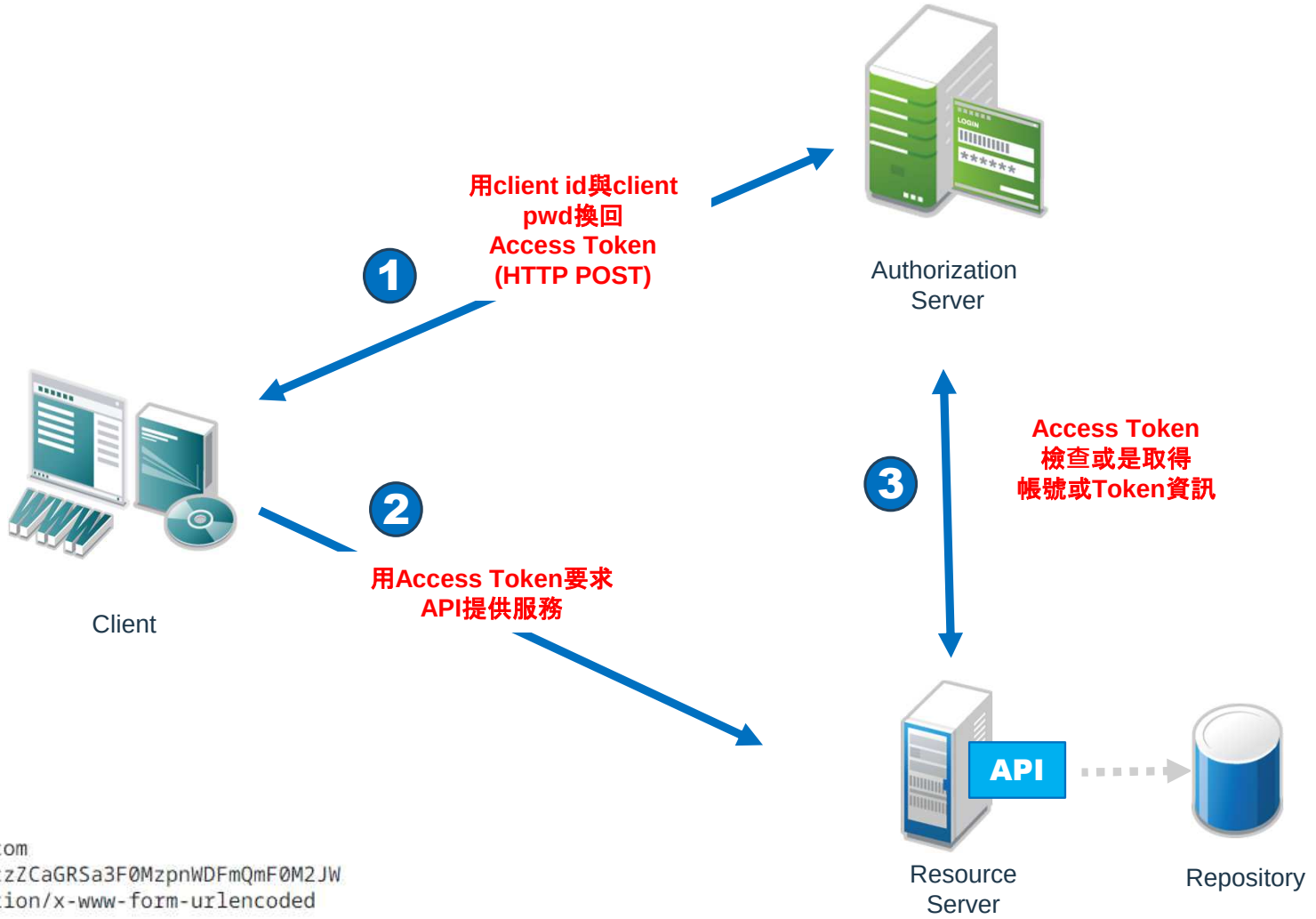

```
GET /authorize?response_type=token&client_id=s6BhdRkqt3&state=xyz
    &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1
Host: server.example.com
```

Resource Owner Password Grant




```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=password&username=johndoe&password=A3ddj3w
```

Client Credential Grant




```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials
```

Grant Type for OAuth 2.1

- **Authorization Code + PKCE**
  - 必須使用Proof Key of Code Exchange (PKCE)
  - Redirect URIs must be compared using exact string matching
- **Implicit 與 Resource Owner Password Credentials**
  - 被移出OAuth 2.1規格
- **Refresh Token**
  - 在public client使用下需要使用者允許或是只能使用一次。

其它Grant Type

- **Device Authorization Grant**
  - RFC 8628
  - 給設備授權使用 (HTTP)
  - 簡易授權介面
  - Smart TVs, Media Consoles, digital picture frames, printers等
- **Security Assertion Markup Language (SAML) 2.0 Bearer Grant**
  - RFC 7522
  - SAML 2.0 Assertion Token converts to OAuth 2.0 Access Token

Token Type

- **Access Token** 用來存取保護資源，是一個具體的字串（string），其代表特定的 scope （存取範圍）與時效。概念上是由 Resource Owner 授予Resource Server ，而Authorization Server 提供access token相關服務。
- **Refresh Token** 用來向 Authorization Server 重新取得一個新的 Access Token。如現有一個 Access Token 過期、無效或是權限不足。在概念上，Refresh Token 代表了 Resource Owner 授權 Client 重新取得新的 Access Token 而不需要再度請求 Resource Owner 的授權。Client 可以自動做這件事，例如 Access Token 過期了，自動拿新的 Token，來讓應用程式的流程更順暢。

使用OAuth 2.0的好處

1. 提高安全性：使用OAuth 2.0 允許用戶在不分享用戶名和密碼的情況下驗證和授權第三方應用。這減少了帳戶遭受破壞的風險，確保用戶的憑證不會暴露給潛在的攻擊者。
2. 用戶控制：OAuth 2.0 允許用戶隨時授予或撤銷第三方應用訪問其數據的權限。這使用戶對其數據擁有更多控制權，確保他們能夠管理誰能訪問他們的數據。
3. 可擴展性：OAuth 2.0 是一個廣泛採用的標準，許多熱門的平台和服務都支持它。這使得開發人員能夠輕鬆地建立與多個服務和平台集成的應用程序。
4. 減少開發時間：OAuth 2.0 提供了一種標準的授權處理方式，可以減少需要從多個來源訪問數據的應用程序的開發時間。

OpenID Connect

OpenID Connect

1. OpenID Connect跟OpenID 1.0/1.1/2.0是完全不一樣的東西。
2. OpenID Connect簡稱OIDC，它和OAuth2.0最大區別: OIDC目的是認證(Authentication)而OAuth2.0目的是授權(Authorization)。
3. OpenID Connect建構於OAuth2.0架構上。用Claims來傳遞End-User資料，並提供ID Token。
4. 啟用OIDC需在Authorization Request階段加上scope = openid。
5. ODIC比OAuth 2.0多增加三個元素：ID Token、UserInfo Endpoint、Standard Set of Scopes
6. 目前OIDC並沒有RFC，但有Financial-grade API (FAPI) Guide

Default Scope

Scope	回傳的Claim
Openid	ID Token claims
Profile	name, family_name, nickname ,given_name, middle_name, nickname, preferred_username, website, picture, profile, gender, zone_info, birthdate, updated_at and locale
Email	email, and email_verified
Address	address
Phone	phone_number, and phone_number_verified

```
{
 "sub": "248289761001",
 "name": "Jane Doe",
 "given_name": "Jane",
 "family_name": "Doe",
 "preferred_username": "j.doe",
 "email": "janedoe@example.com",
 "picture": "http://example.com/janedoe/me.jpg"
}
```

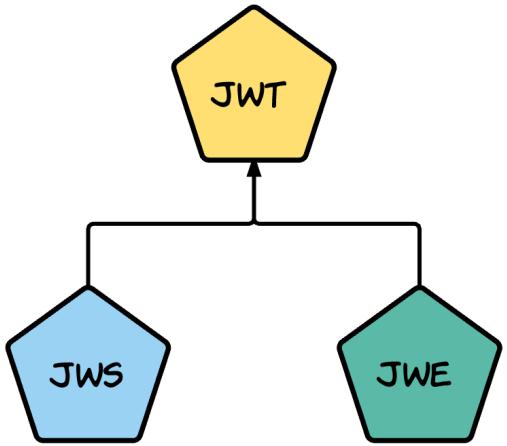
ID Token

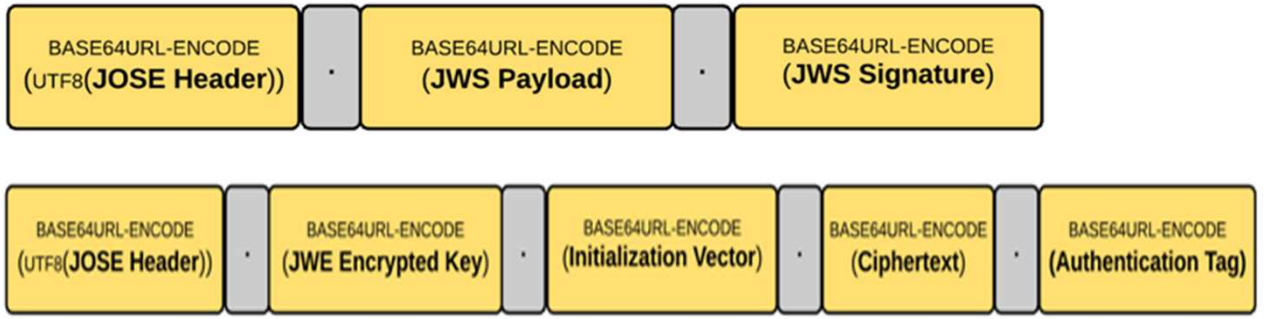
- ID Token內容就是一個Json Web Token (JWT)
- ID Token含有以下資訊：
  - iss (Issuer) Claim **核發者**
  - sub (Subject) Claim **使用者ID**
  - aud (Audience) Claim **接受者**
  - exp (Expiration) Claim **有效期**
  - iat (Issued At time) Claim **核發時間**
  - auth_time (time subject was authenticated) Claim
  - nonce (nonce) Claim : client session
  - acr (Authentication Context Class Reference) Claim
  - amr (Authentication Methods References)
  - azp (Authorization Party)

```
{
 "iss": "https://server.example.com",
 "sub": "24400320",
 "aud": "s6BhdRkqt3",
 "nonce": "n-0S6_WzA2Mj",
 "exp": 1311281970,
 "iat": 1311280970,
 "auth_time": 1311280969,
 "acr": "urn:mace:incommon:iap:silver"
}
```

JWT

- JSON Web Token (JWT)是一個公開的規範(RFC 7519)
- 定義JSON格式的物件並且用於傳遞資訊，用演算法來確定內容的完整性
- 可透過secret(HMAC演算法)或非對稱式金鑰加密(RSA or ECDSA演算法)





- JWT有二種:JWS與JWE
- JWS : JSON Web Signature
- JWE : JSON Web Encryption

JWS Token

Algorithm HS256

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.e
yJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikp
vaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.Sf
lKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQss
w5c
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

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

PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

Signature Verified

SHARE JWT

JWE Token


```
eyJhbGciOiJSU0EtT0FFUCIsImVuYyI6IkEyNTZDQkMtSFM1MTIiLCJraWQiOiIxOGIxY2Y3NThjMWQ0ZWM2YmRhNjU4OT
M1N2FiZGQ4NSIsInR5cCI6IkpXVCIsImN0eSI6IkpXVCJ9.gCbxP78o3DgpDTUQbuHniuGgYpATqgGkRGy7paC6hRrz7N7
eIa6sAOWDO9Fhnj-c8ocMl4cF4Jb_mv5qRPCh9r57PBqx7jOhMIMPTwJGpjcyBaqtHlZlu1vupY5tQ3Y2jGz1Ti4Bnywae
EHPyIPQJtN7F7hIAORzj7IY4sIKkVXtQJZgaKW8pEHq_GCqj8i5aaiM0uJnRG3GOh3livp9Npjv9doqp3gyPa1zjrg2H1R
sOGn0j2QMGvtuVfkuNwF-SoPKFECyHOq0ZK1oH2sTO8-JwvHflbIZQr5xWTpS8q7MbUXEuqURtrg0Tj-2z6tdaOLT4b3Ue
DufK2ar3bBfRD4-nRALtoY0ekcMyGFOS7o1Mxl3hy5sIG-EySyWeuBVy68aDWDpi9qZoQuY1TbxxakjncCOGu_Gh1l1m_m
K2l_IdyXCT_GCfzFq4ZTkPZ5eydNBAPZuxBLUb4BrMb5iDdZjT7AgGOlRre_wIRHmmKm8W9nDeQQRmbIXO23JuOw9.BDCa
rfq2r_Uk8DHNfsNwSQ.4DuQx1cfJXadHnudrVaBss45zxyd6iouuSzZUyOeM4ikF_7hDOgwmaCma-Z97_QZBJ5DzVn9SJh
KUTAqpVR3BRGAxJ_HAXU5jaTjXqbvUaxsh7Z5TgZ9eck0FIoe1lkwv51xEvYqqQ_Xojr4MAEmLuME_9ArCK9mNaMADIzOj
4VoQtaDP1l26ytocc-oENifBRYGu28LbJLkyQKzyQy6FuAOtWjLM0WCXV7-o_dvj6qfeYHNBD7YB5xyqdgD8dcxMBNd2sK
73YsZPHEa0V1-8zz7hm3bH3tZelpwPWScqLLW_SUH586c0FVeI6ggvqzjfLZ_Y6eQibVSdXfOtJBk22QrLsuCXbRK8G1w9
t23Pwu8ukUAw4v0l7HeaW_0SJyKSPQANRP83MyFbK7fmzTYaW9TYN2JrKN-PLpd2dIFSm2Ga_EfaCwNJBm4RDMzDNrf-O0
AissvYyHb0WaALiCiFCogliYqLzRB6xDb-b4964M.J7WDOFLRRPJ7lLpTfN2mOiXLDg5xtaF-sLQ4mOeN5oc
```

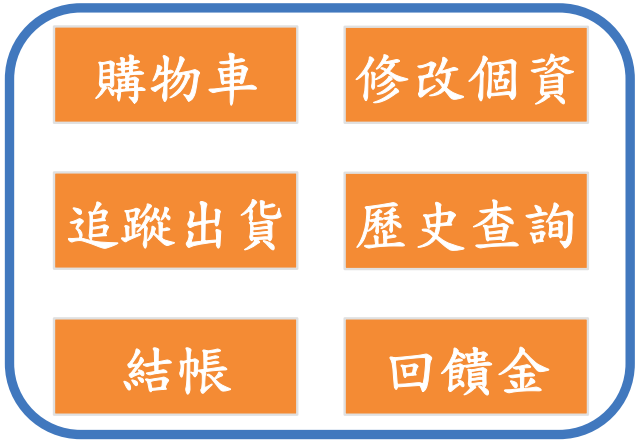
OAuth2.0 vs OIDC 角色定義

OAUTH 2.0

Resource Owner | **USER AGENT** | **CLIENT** | **Source Server** | **Authorization Server**

使用者

瀏覽器



購物平台系統選單

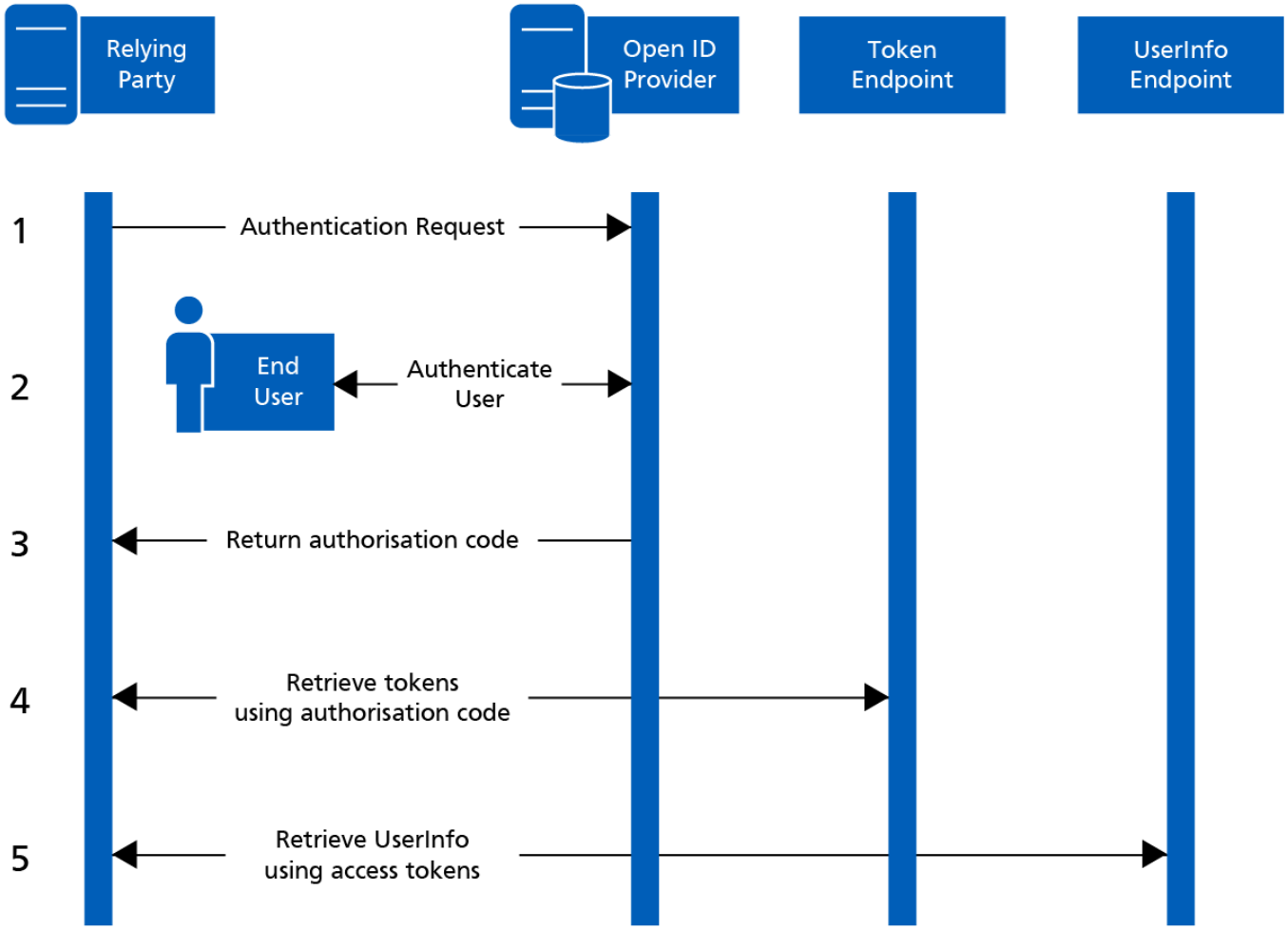
API服務

認證與授權伺服器

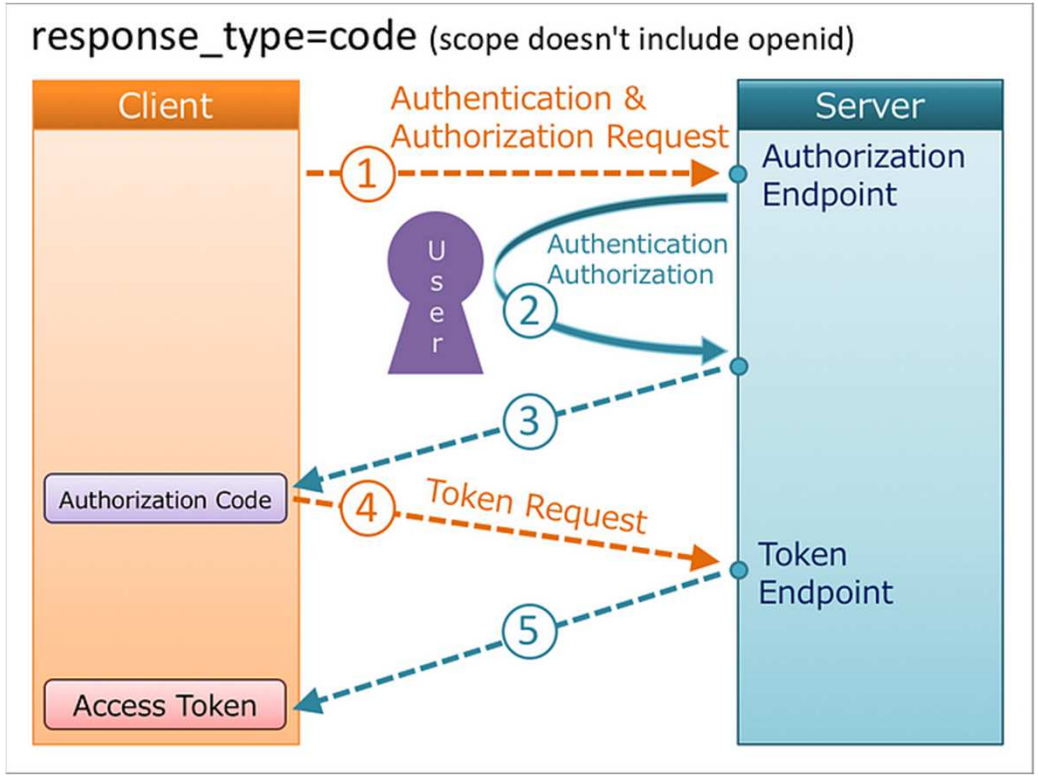
End-User | **USER AGENT** | **Relying Party** | **Source Server** | **OpenID Provider**

OpenID Connect

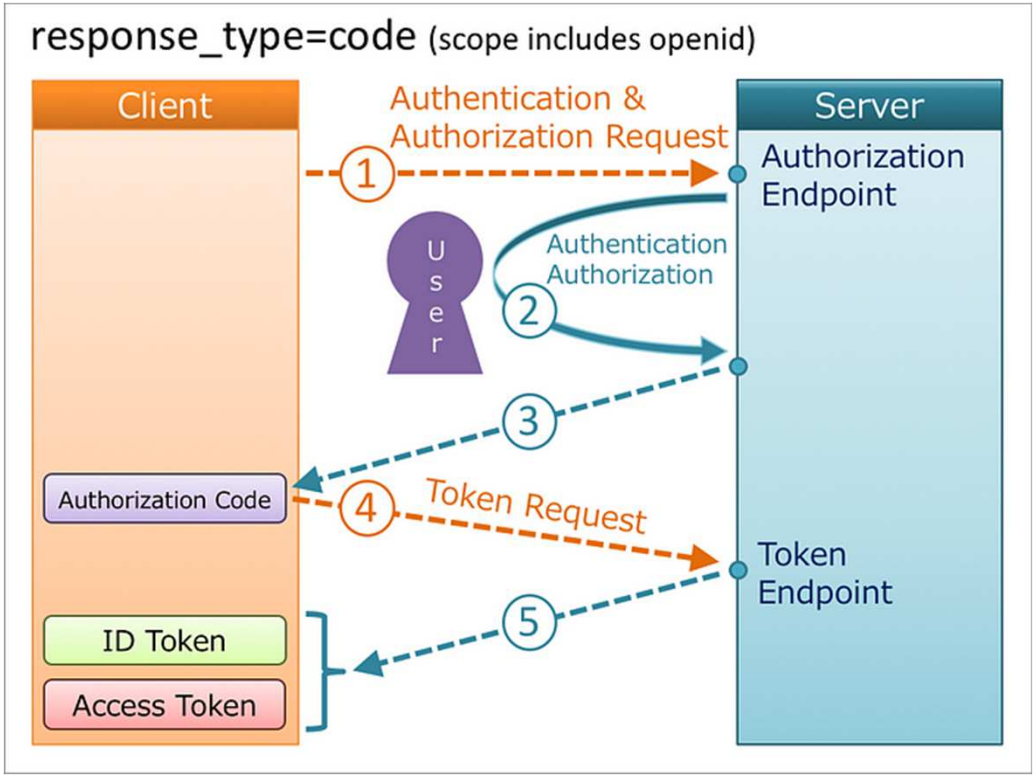
Authorization Code Grant



OAuth2.0 vs OIDC

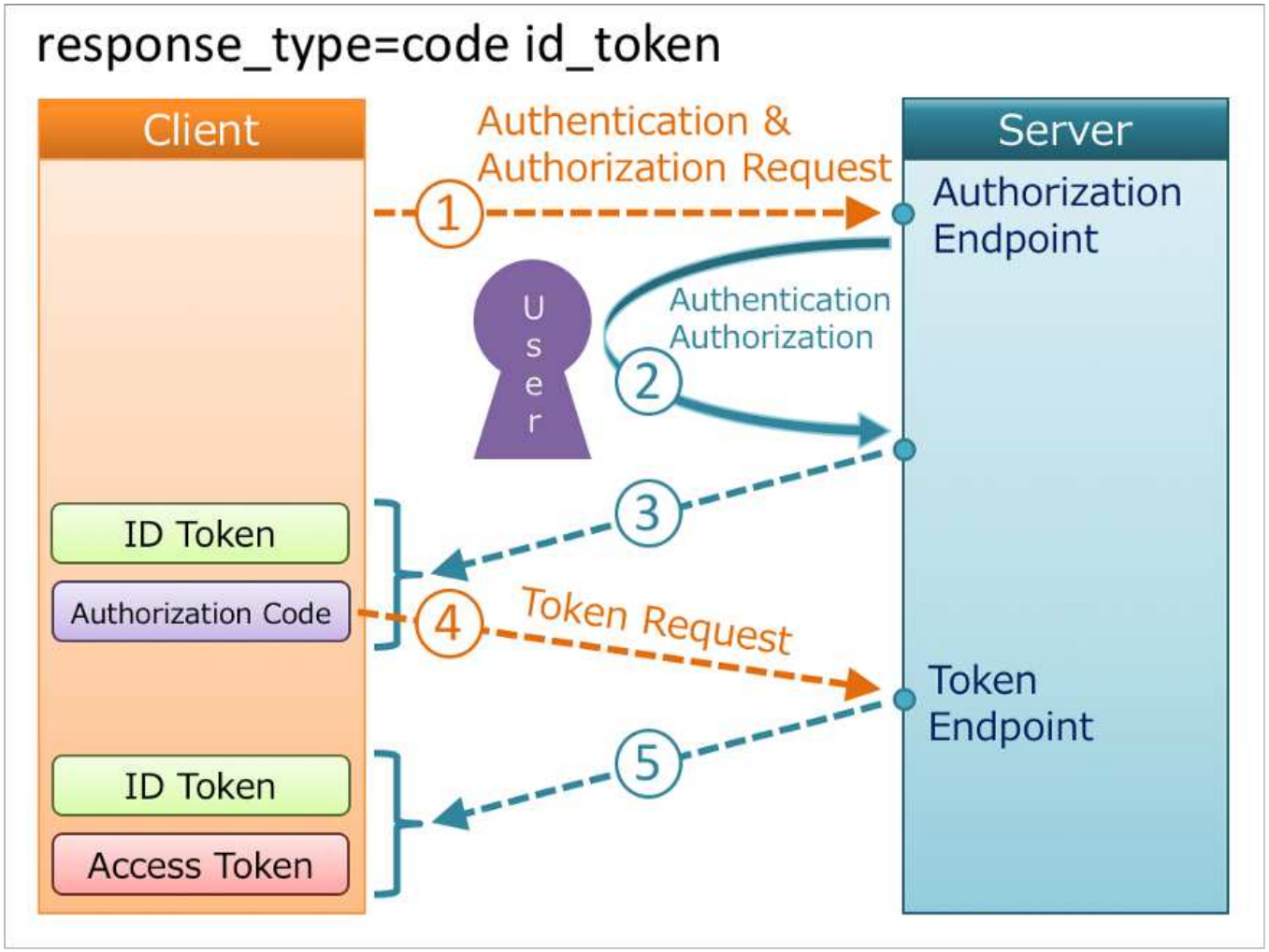


OAuth 2.0



OpenID Connect

Hybrid Flow



OpenID Connect Hybrid Flow適用於需要更高安全性、更多控制、刷新令牌支援、額外聲明和同意步驟以及跨設備和應用程式類型的情況。

RFC

- RFC 6749 — The OAuth 2.0 Authorization Framework
- RFC 6750 — The OAuth 2.0 Authorization Framework: Bearer Token Usage
- RFC 7515 — JSON Web Signature (JWS)
- RFC 7516 — JSON Web Encryption (JWE)
- RFC 7517 — JSON Web Key (JWK)
- RFC 7518 — JSON Web Algorithms (JWA)
- RFC 7519 — JSON Web Token (JWT)
- RFC 7523 — JSON Web Token (JWT) Profile for OAuth 2.0 Client

OAuth 2.0 與 OIDC 差別

	OAuth 2.0	OpenID Connect
主要目的	在不暴露用戶的資料下，訪問社交媒體資料等	在OAuth 2.0基礎上建立身份驗證層
身份驗證與授權	授予客戶端訪問特定資源的權限，但不關心用戶的身份	使用 ID 令牌來傳遞用戶的身份信息，以確保用戶是合法的，並使用 Access Token 來授予對資源的訪問權限
協議擴展	允許實現者根據其需求進行自定義擴展	OAuth 2.0 的擴展進行了規範化，OIDC 需要遵循特定的流程和訊息格式。
令牌	使用 Access Token 來表示授權，用於訪問受保護的資源	同時使用 ID 令牌（包含身份信息）和 Access Token（用於訪問資源），從而實現身份驗證和授權

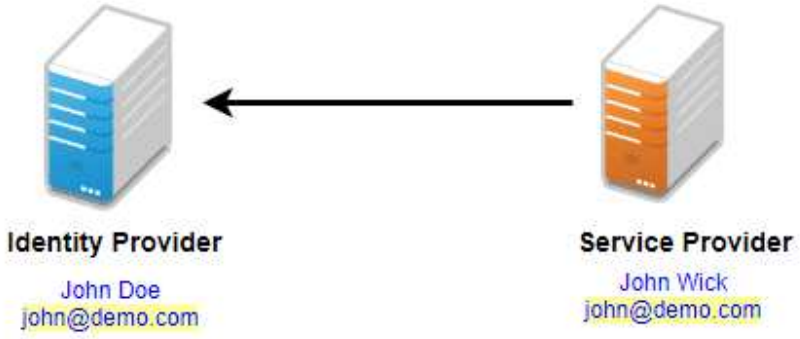
SAML 2.0

Identity Federation

Federation主要是規範如何將同一個人的帳號與自己在不同網頁應用系統上的帳號關連起來。

SAML 2.0是一種Identity Federation的機制。

SAML 2.0（安全斷言標記語言 2.0）是一種基於XML的標準，用於在各方之間交換身份驗證和授權信息，主要用於Web的單一簽入（SSO）場景。它使用戶可以通過一次登錄使用其令牌，然後訪問多個應用程序或服務，而無需為每個應用程序單獨進行身份驗證。



SAML 2.0 術語

SAML術語

- Assertions
  - XML 格式的令牌，用於在消息中傳輸用戶身份信息，例如身份驗證、屬性和授權信息。
- Protocols
  - 用於獲取身份驗證數據和管理身份的請求消息和響應消息的類型。
- Bindings
  - 用於傳輸消息的通信方法。
- Profiles
  - protocols、assertions和bindings的組合，它們一起用於創建聯合併啟用聯合單點登錄。

Three Entities for SAML

USER AGENT

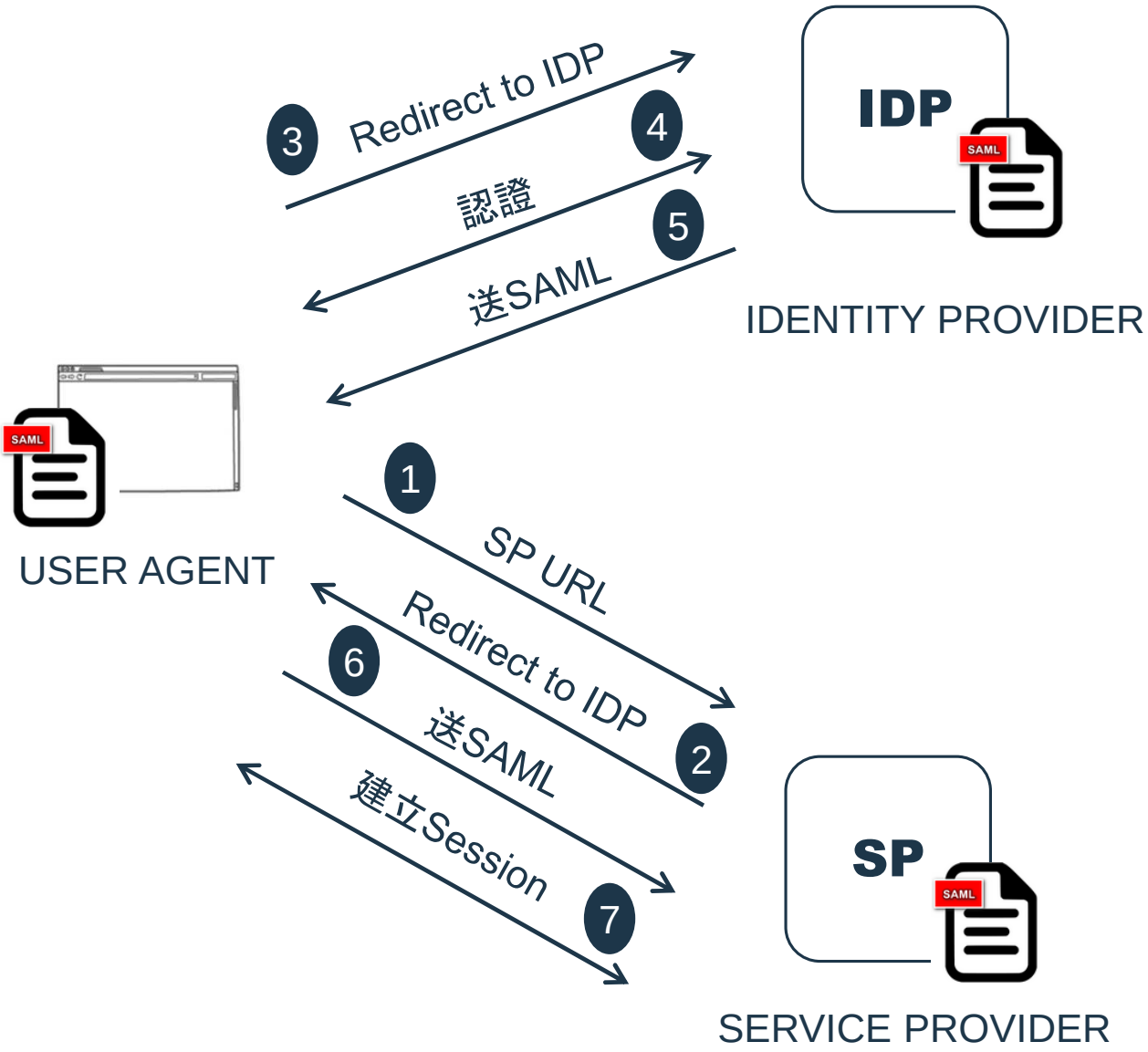
IDP

IDENTITY PROVIDER

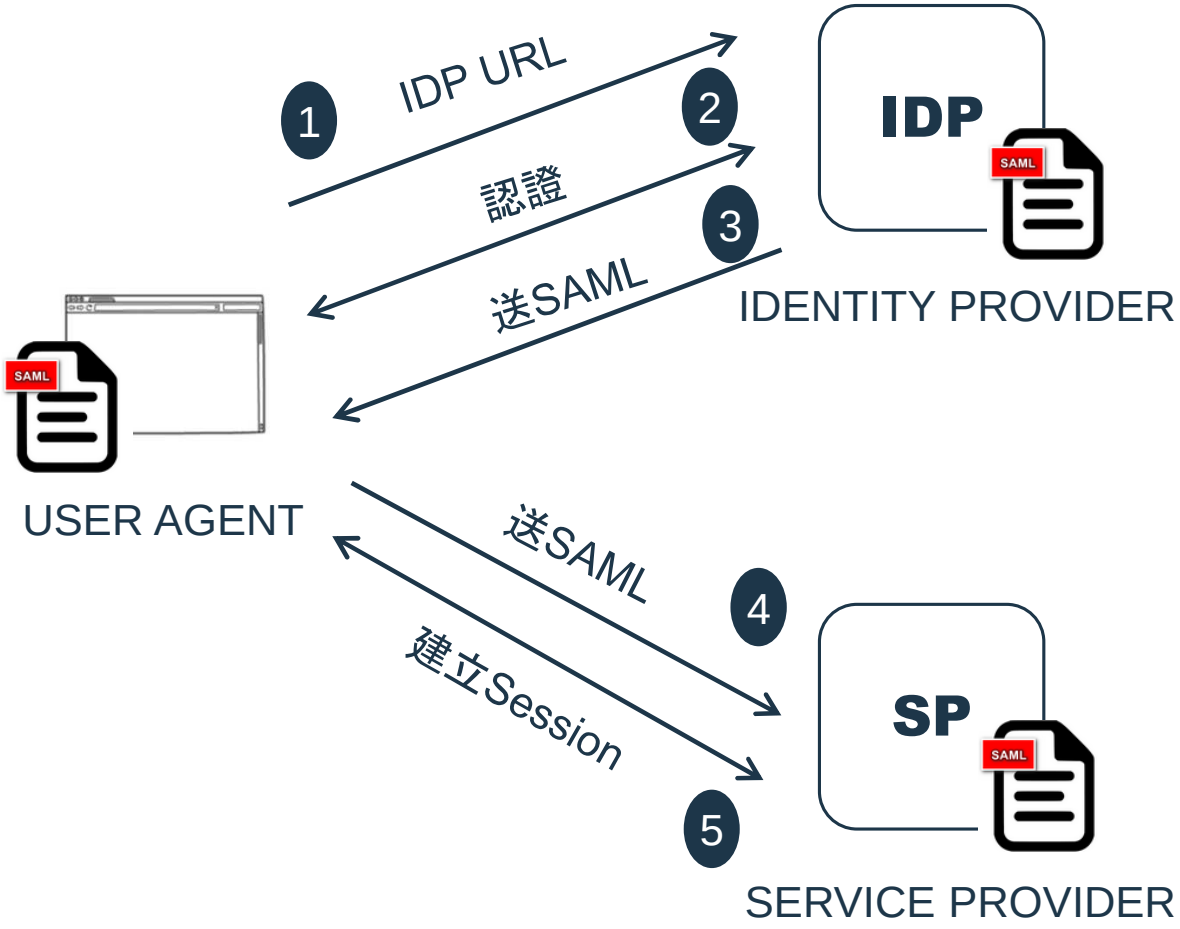
SP

SERVICE PROVIDER

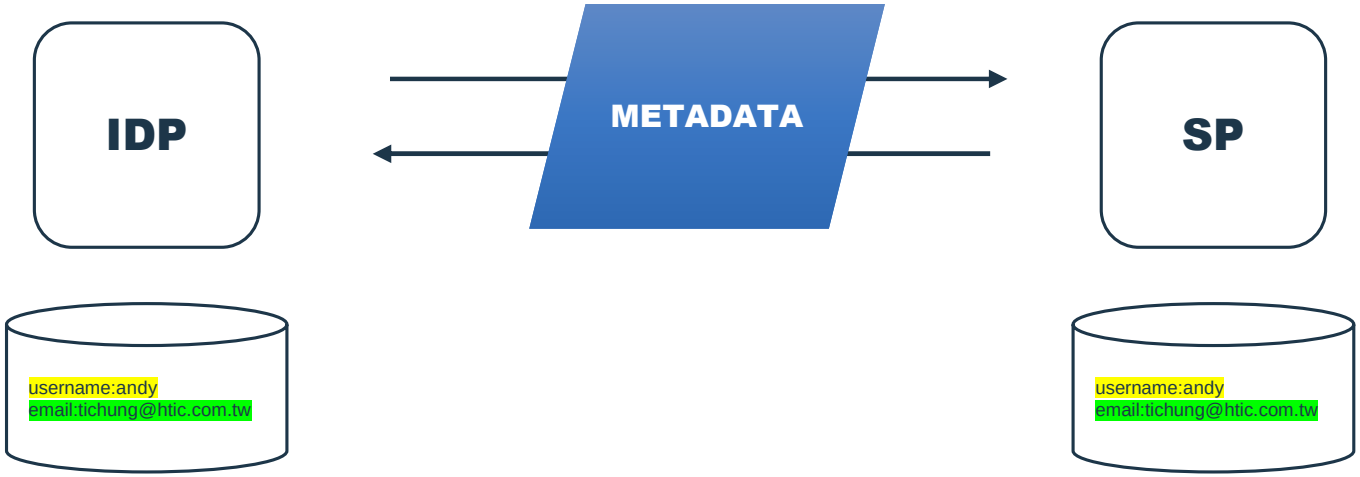
SP Initial


IDP
SAML
IDENTITY PROVIDER
3
Redirect to IDP
4
認證
5
送SAML
SAML
USER AGENT
1
SP URL
Redirect to IDP
2
6
送SAML
建立Session
7
SP
SAML
SERVICE PROVIDER

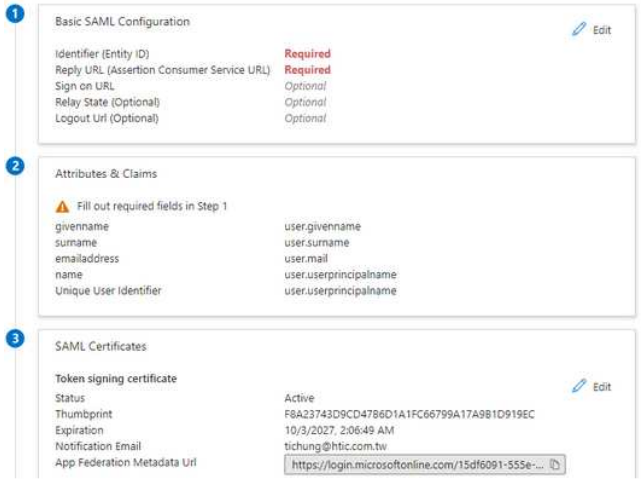
IDP Initial


1
IDP URL
2
認證
3
送SAML
IDP
SAML
IDENTITY PROVIDER
SAML
USER AGENT
送SAML
4
建立Session
5
SP
SAML
SERVICE PROVIDER

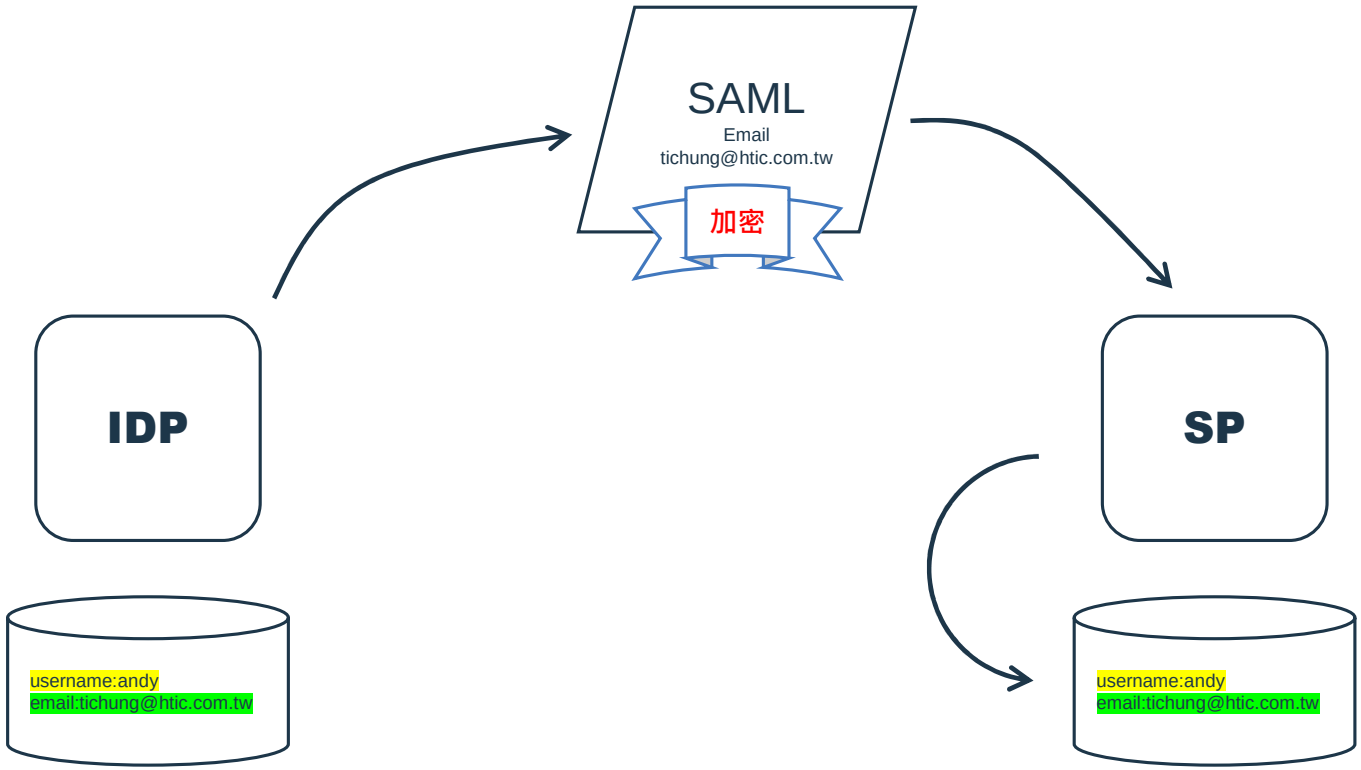
SAML Federation – 設定（一）



IDP與SP各自產生METADATA



SAML Federation – 設定（二）

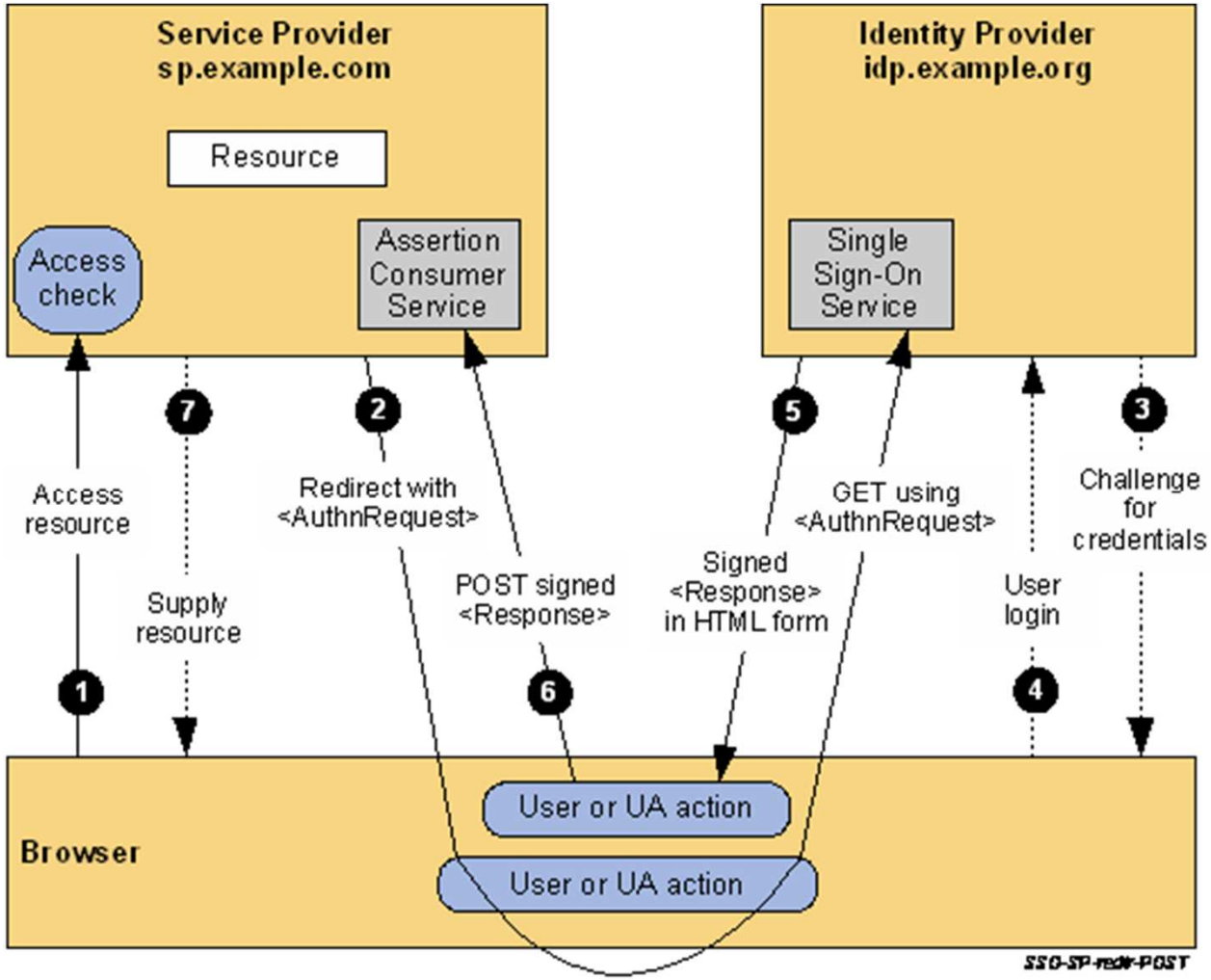


交換雙方的憑證 + 指定對應的屬性

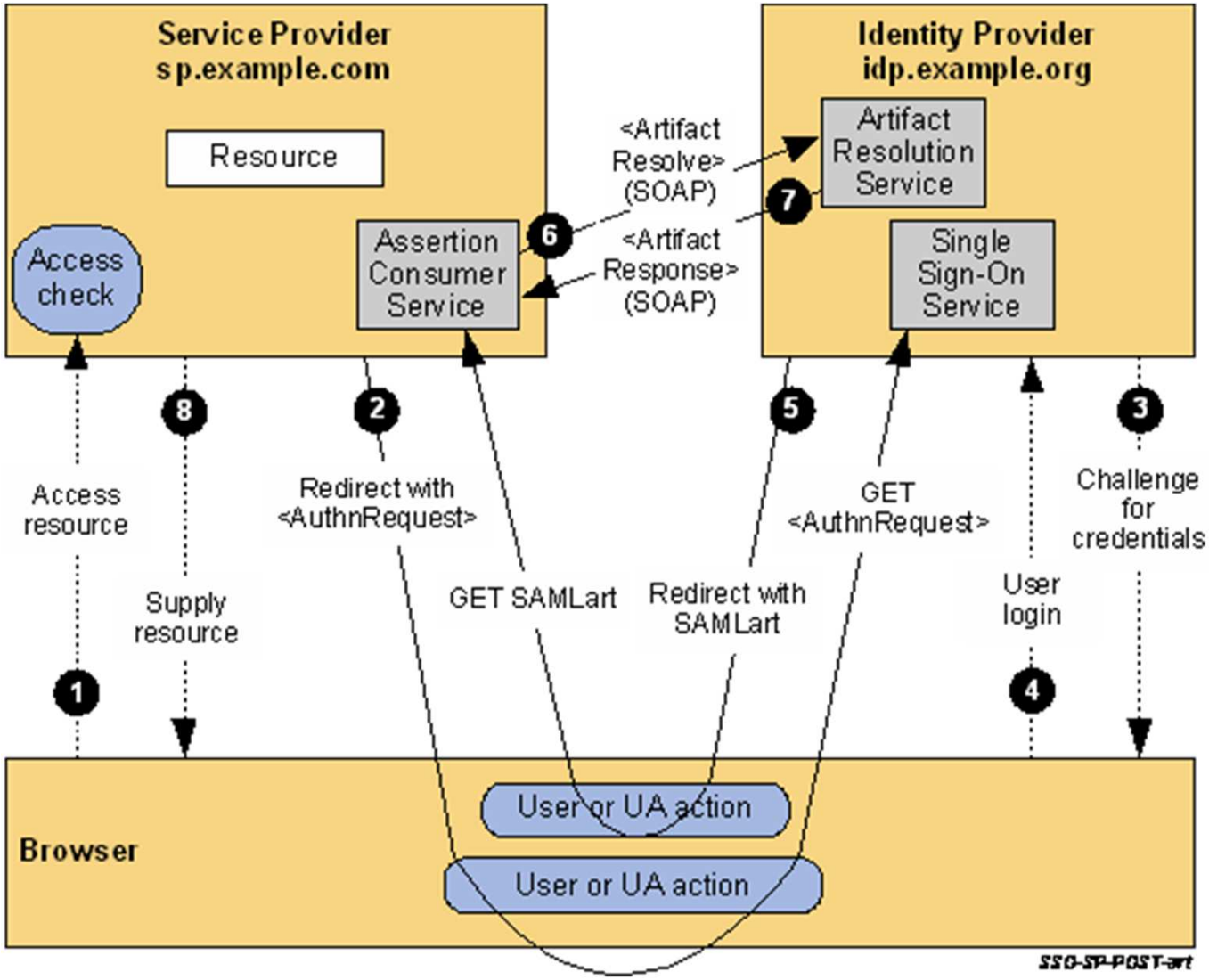
SAML 2.0 BINDINGS

- SAML SOAP BINDING (BASED ON SOAP 1.1)
- REVERSE SOAP (PAOS) BINDING
- HTTP REDIRECT BINDING
- **HTTP POST BINDING**
- **HTTP ARTIFACT BINDING**
- SAML URI BINDING

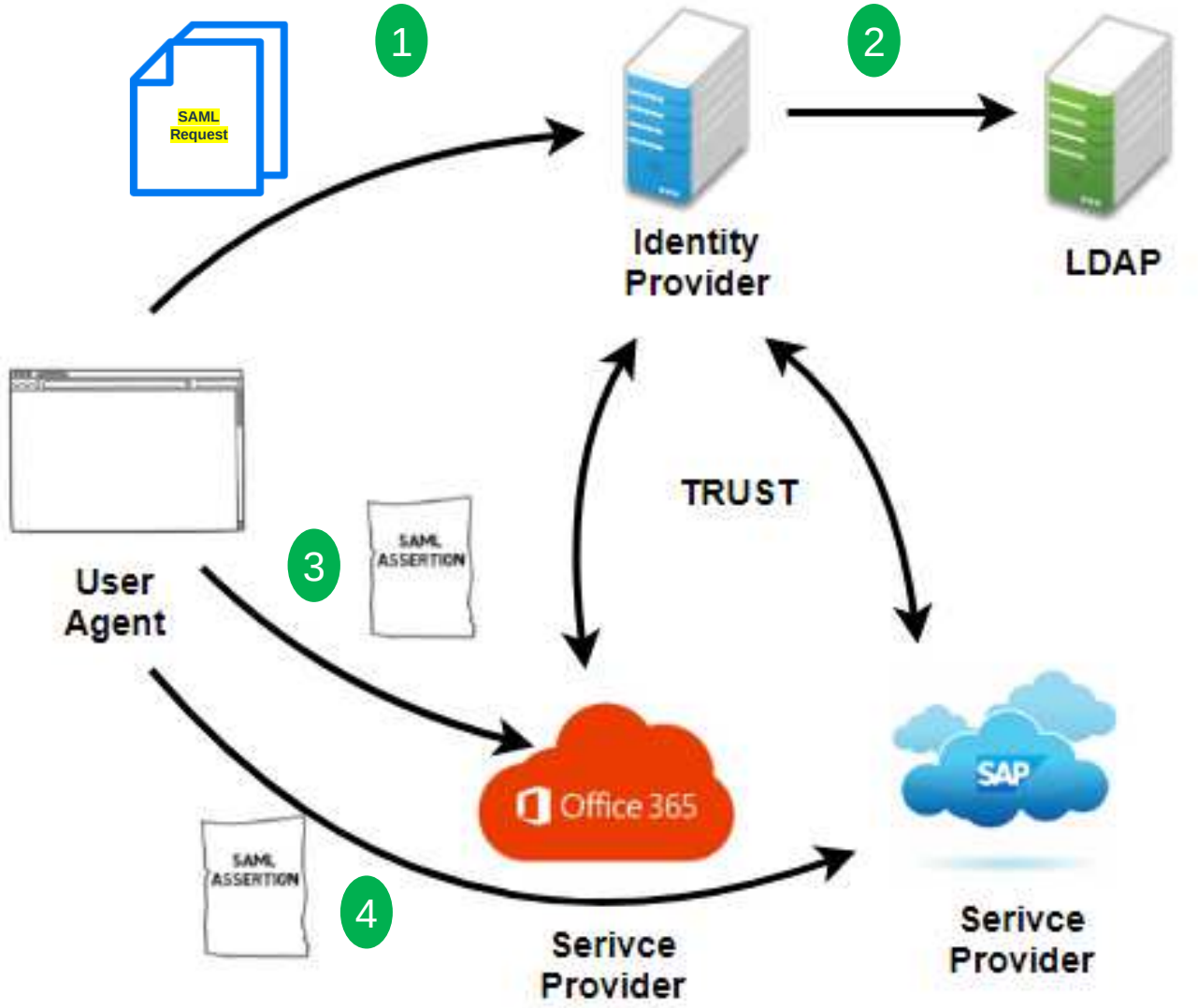
SAML 2.0 HTTP POST BINDING



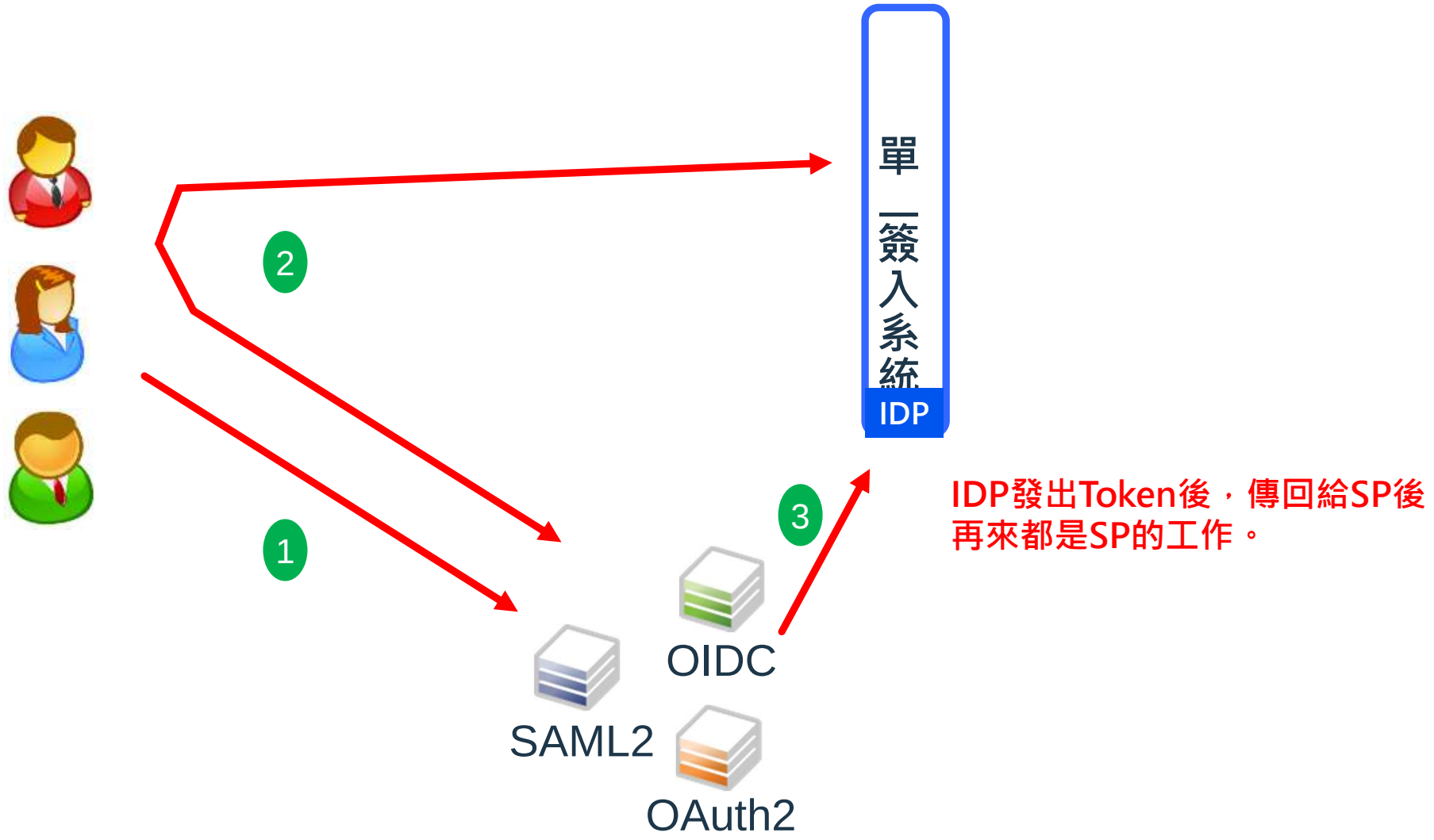
SAML 2.0 HTTP ARTIFACT BINDING



SAML 2.0 SSO


SAML Request
1
2
Identity Provider
LDAP
TRUST
User Agent
3
SAML ASSERTION
Office 365
SAP
SAML ASSERTION
4
Serivce Provider
Serivce Provider

單一簽入系統-Token認證

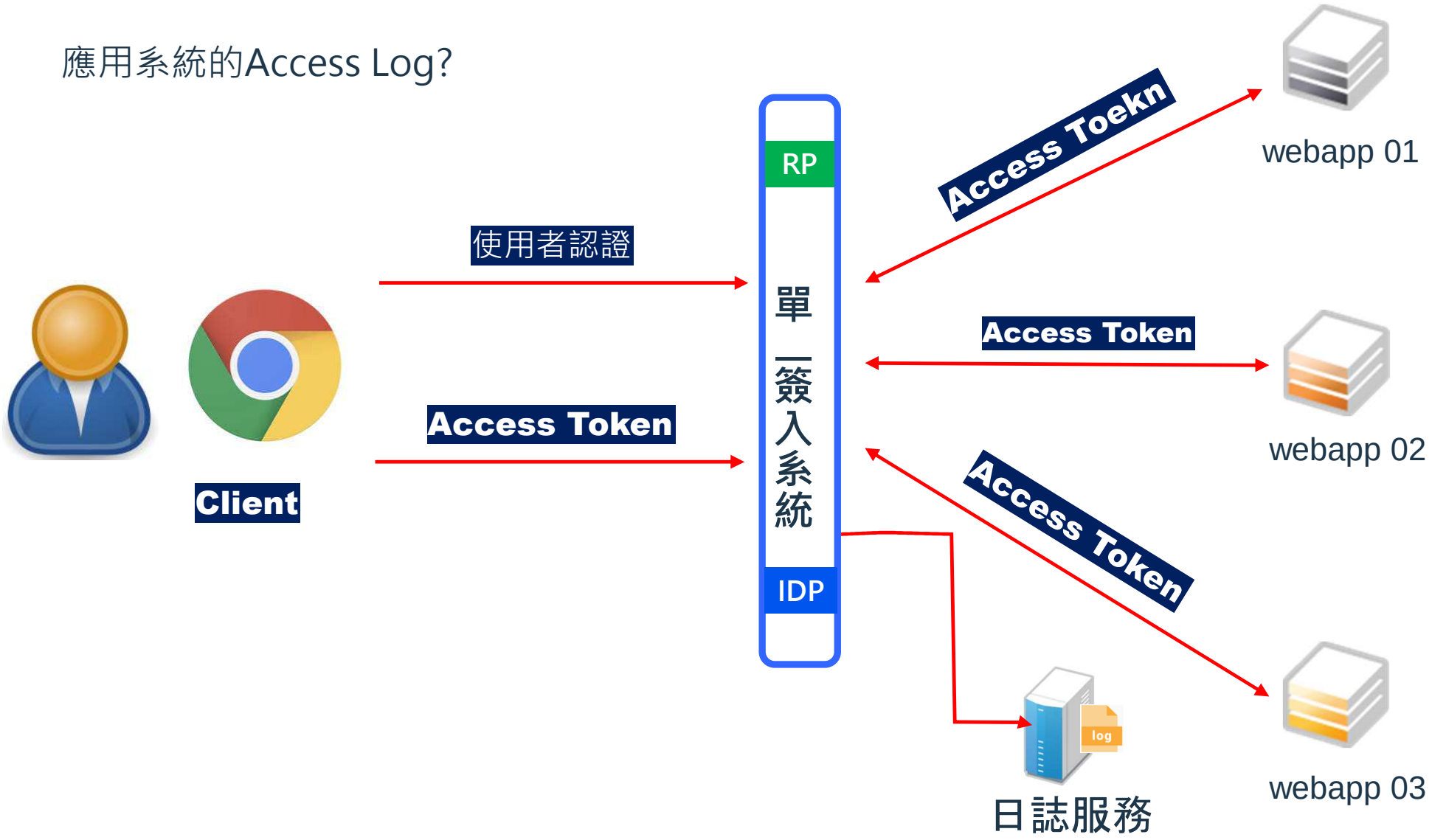


現代式認證的問題

1. JWT Token使用與保護?
   - 每一個Client與應用系統要保護好Token，並且不要分享給其他應用系統。
2. 該由誰來確認Access Token or JWT Token的有效性?
   - 由Service Provider或 Resource Server。
3. Access Token與Refresh Token有效時間要設定多少?
   - 基本原則Access Token較短，Refresh Token較長一點。
   - 依據環境需求而決定。
4. IDP and SP的使用者帳號同步?
   - SP and IDP是透過Federation機制，因此雙方都需要有帳號。可透過SAML JIT機制完成，或是scim機制做帳號整合同步。

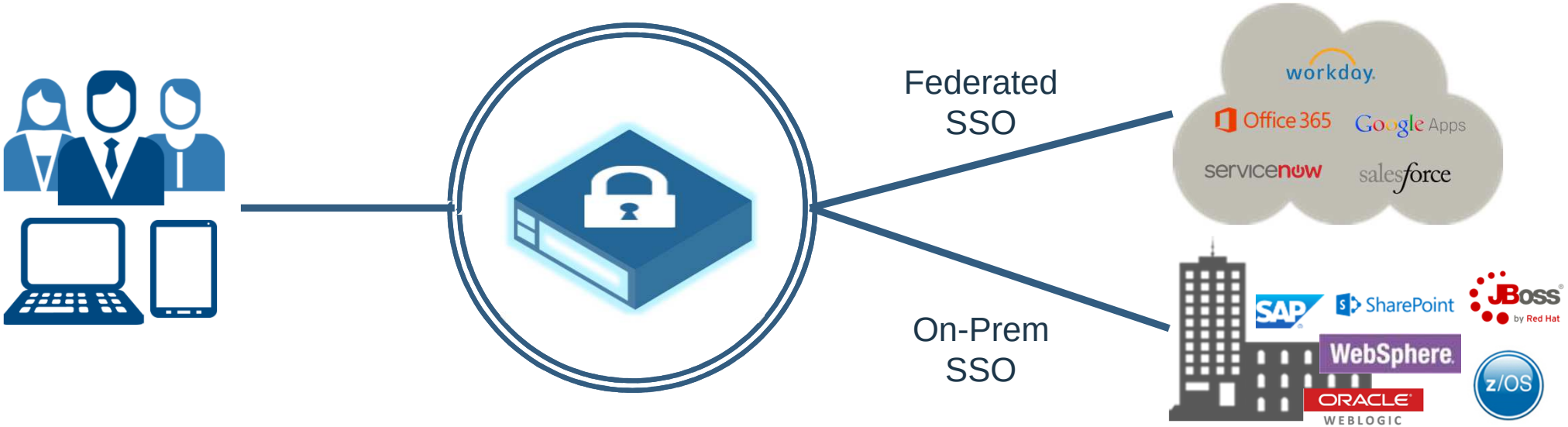
現代式認證的問題（二）

應用系統的Access Log?



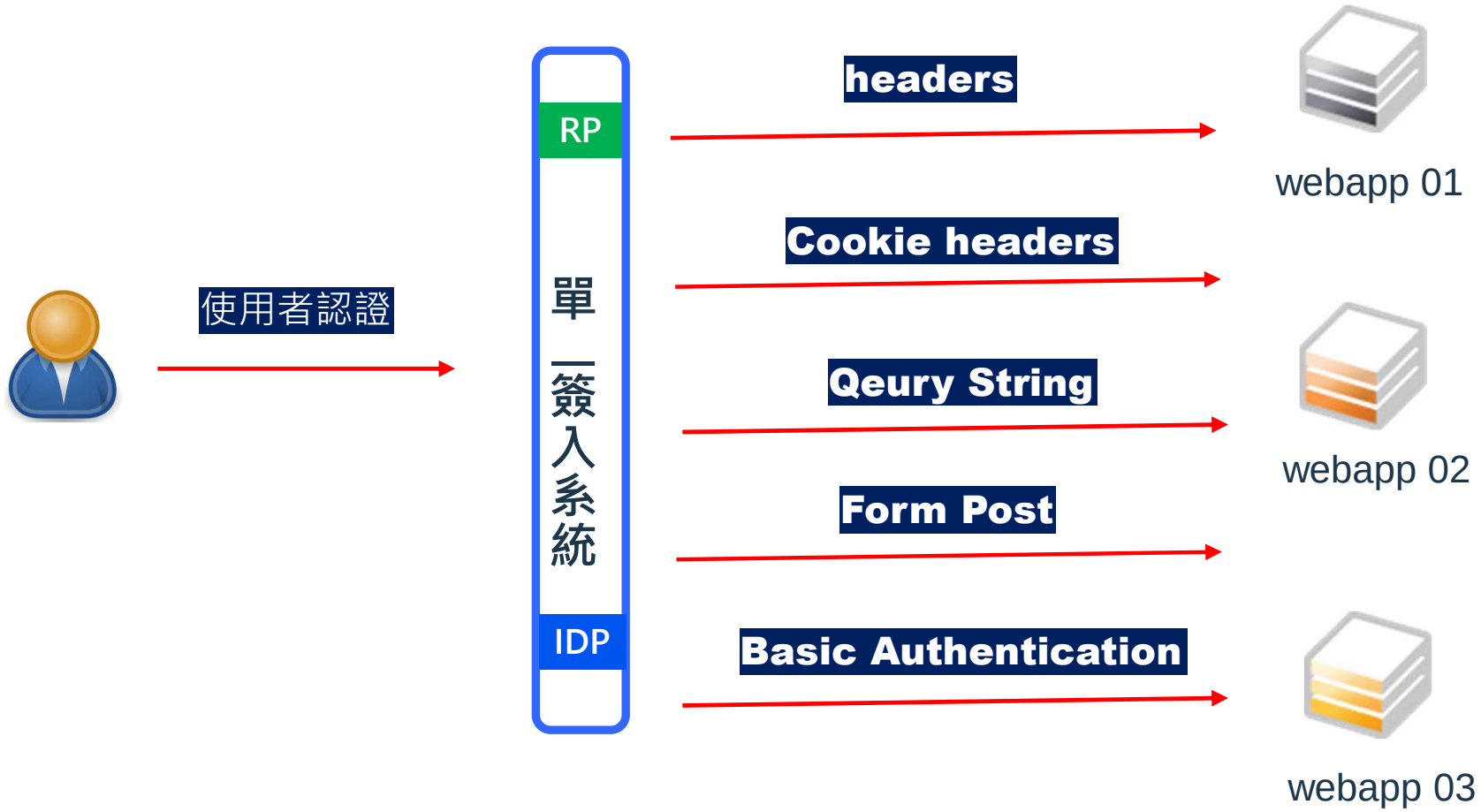
混和雲單一簽入

混和環境單一簽入

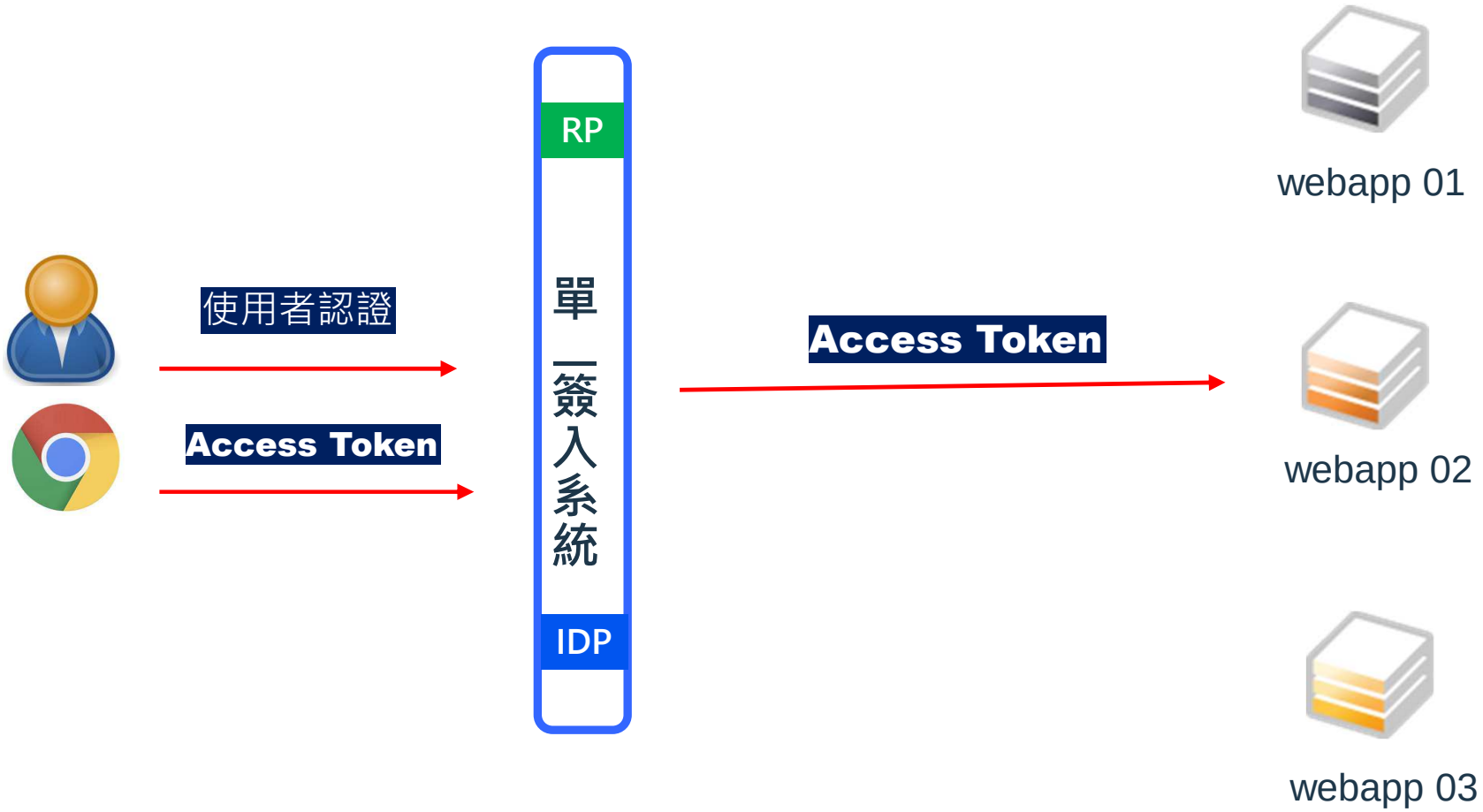


- 透過統一平台整合地端與雲端單一簽入
- 使用者只需輸入一次認證，既可登入到地端與雲端服務

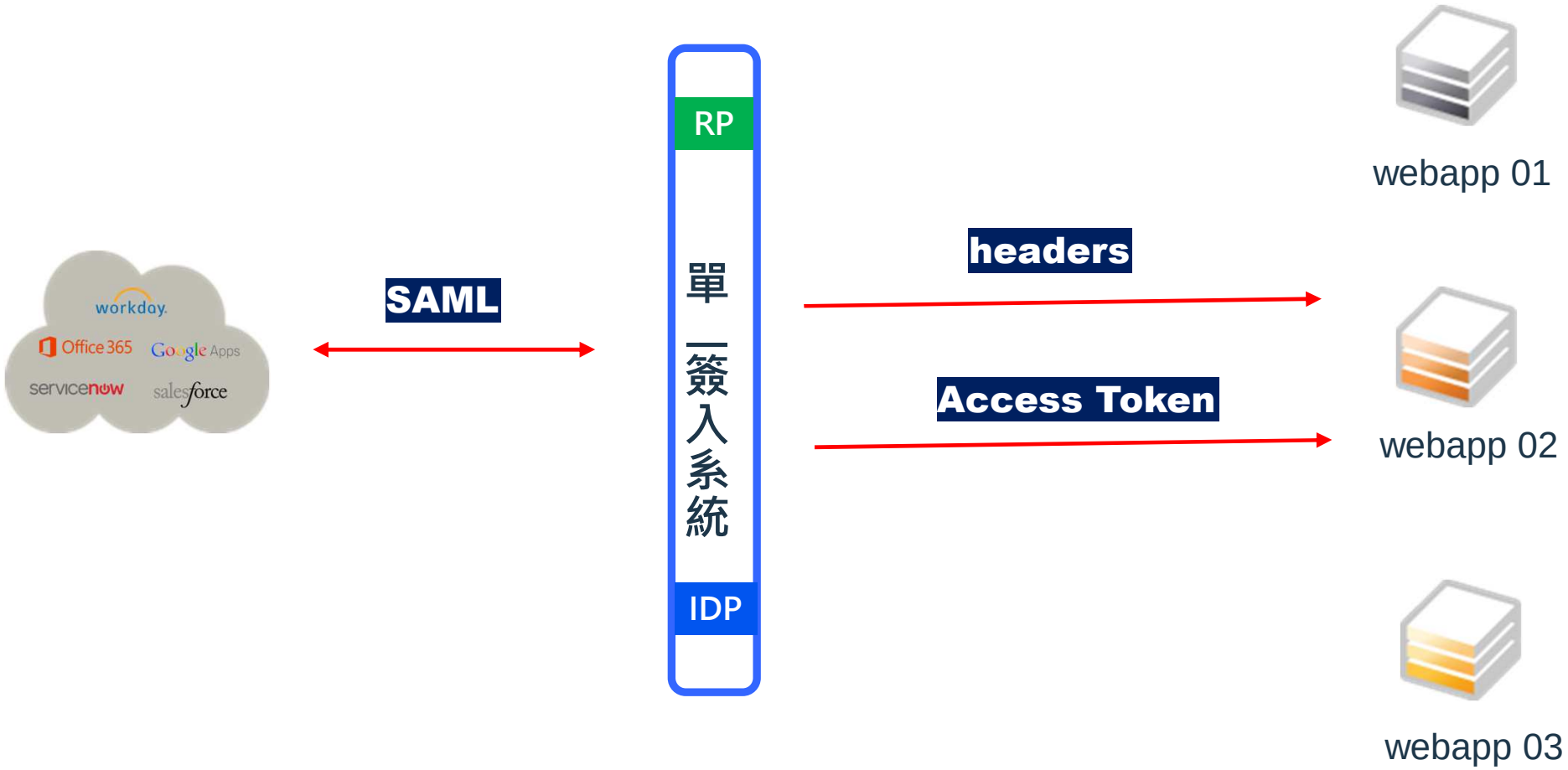
單一登入系統可做到 (一)


使用者認證
RP
單一簽入系統
IDP
headers
Cookie headers
Qeury String
Form Post
Basic Authentication
webapp 01
webapp 02
webapp 03

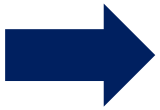
單一登入系統可做到 (二)


RP
單一簽入系統
IDP
使用者認證
Access Token
Access Token
webapp 01
webapp 02
webapp 03

單一登入系統可做到 (三)


workday.
Office 365
Google Apps
servicenow
salesforce
SAML
RP
單一簽入系統
IDP
headers
Access Token
webapp 01
webapp 02
webapp 03

傳統單一簽入規劃



DMZ

RP

單一簽入系統

reCAPTCHA

Headers, Form Post

多因子服務

webap01

webap03

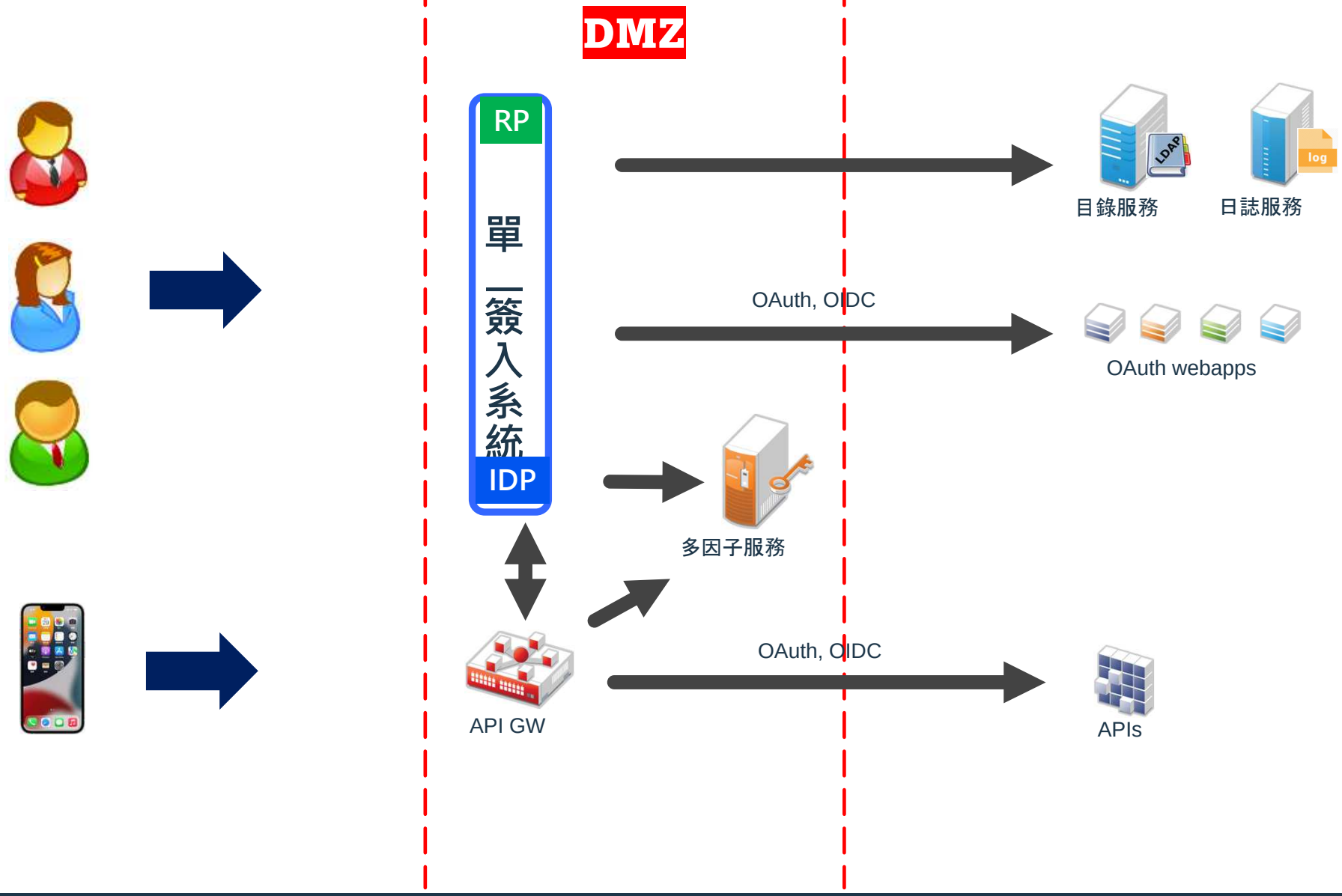
webap02

webap04

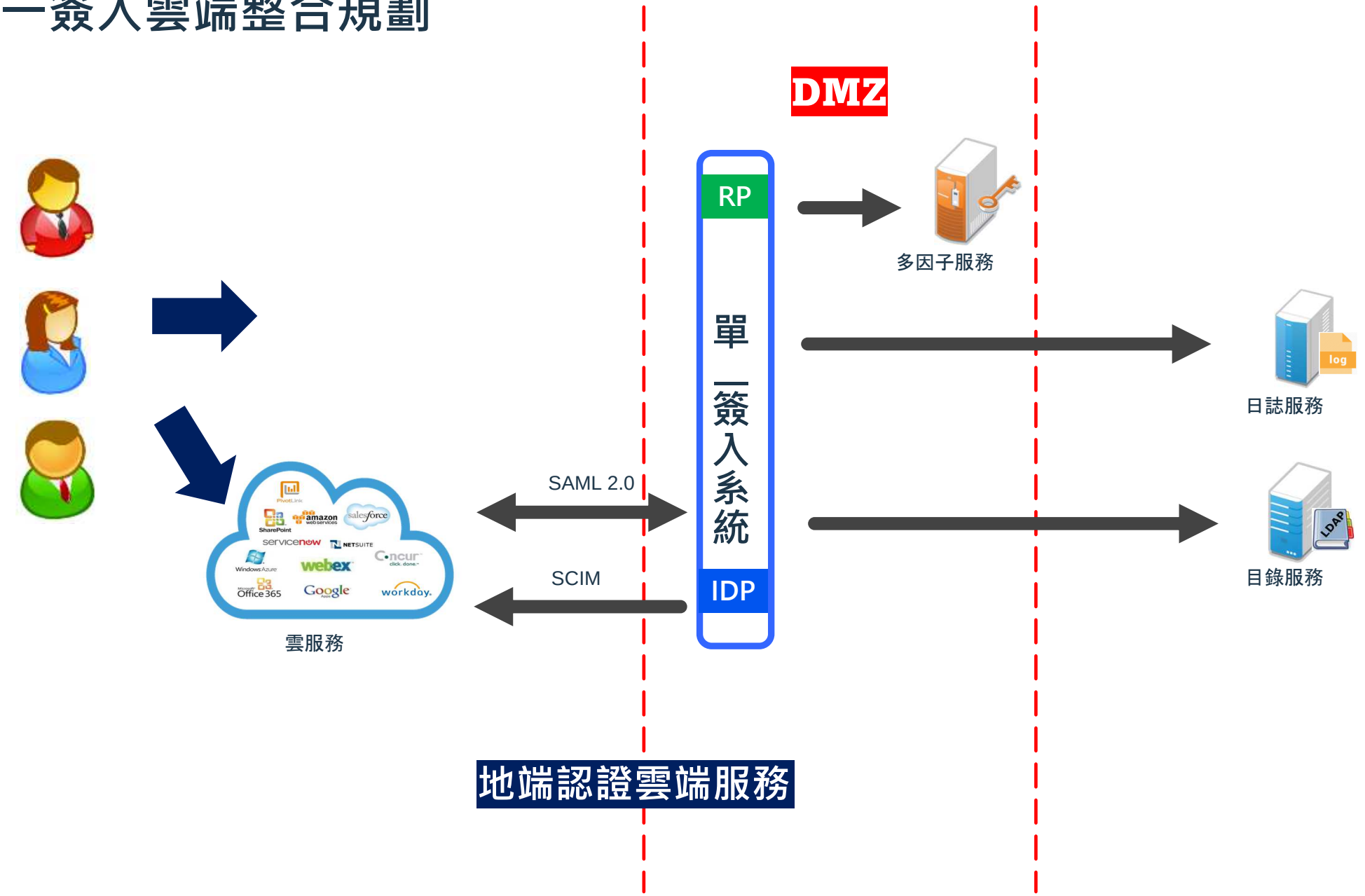
目錄服務

日誌服務

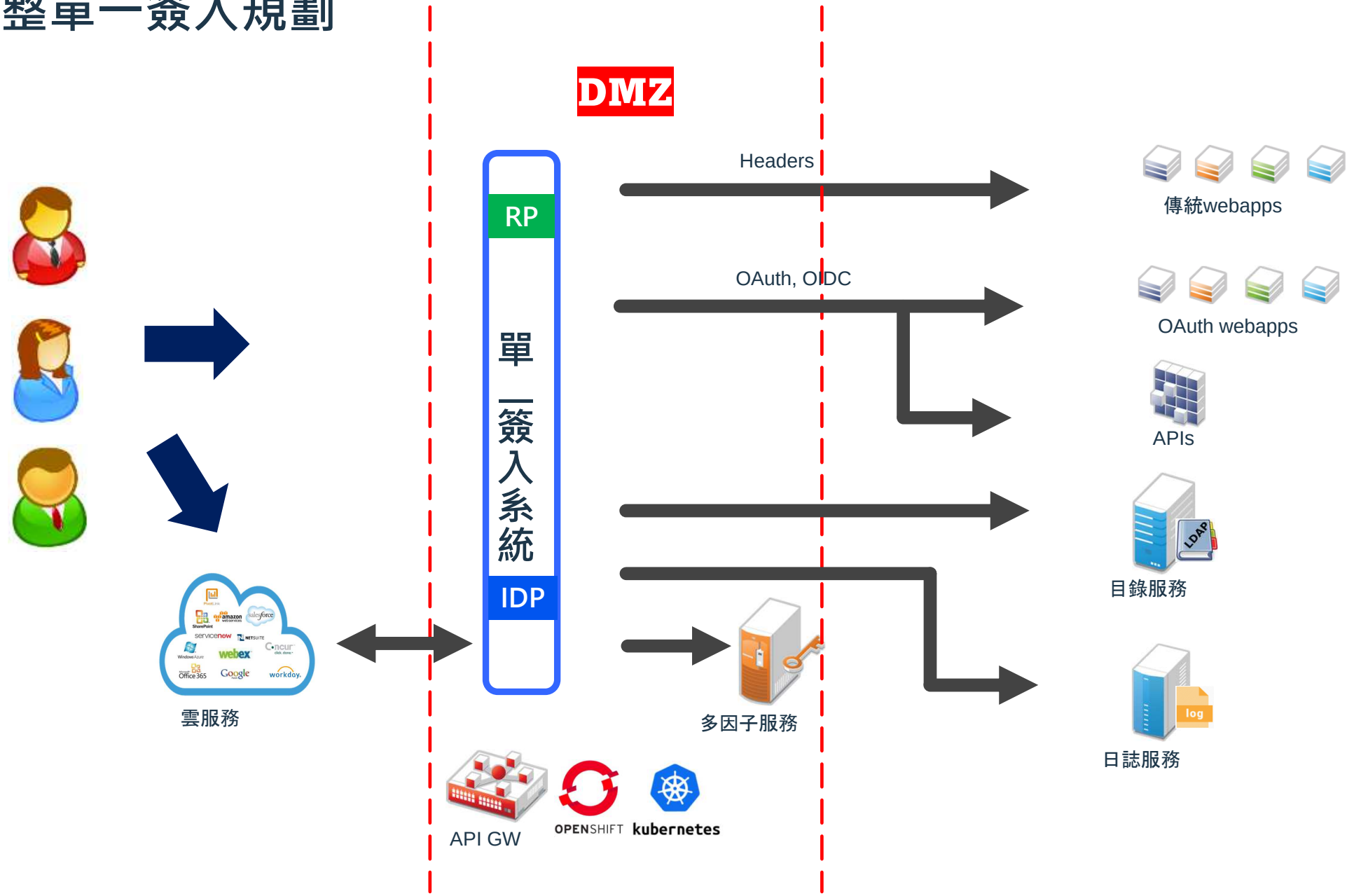
單一簽入Token認證規劃



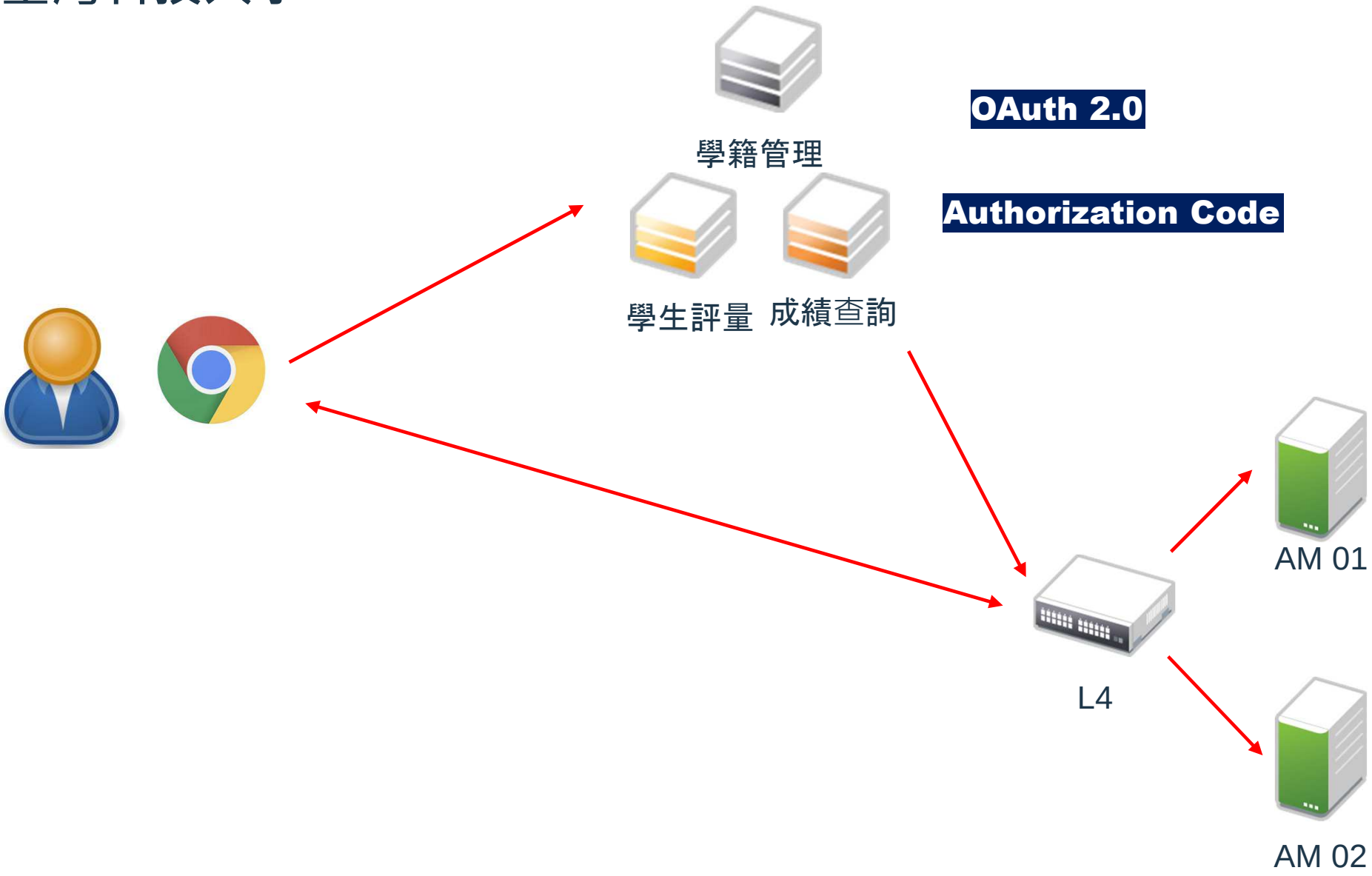
單一簽入雲端整合規劃



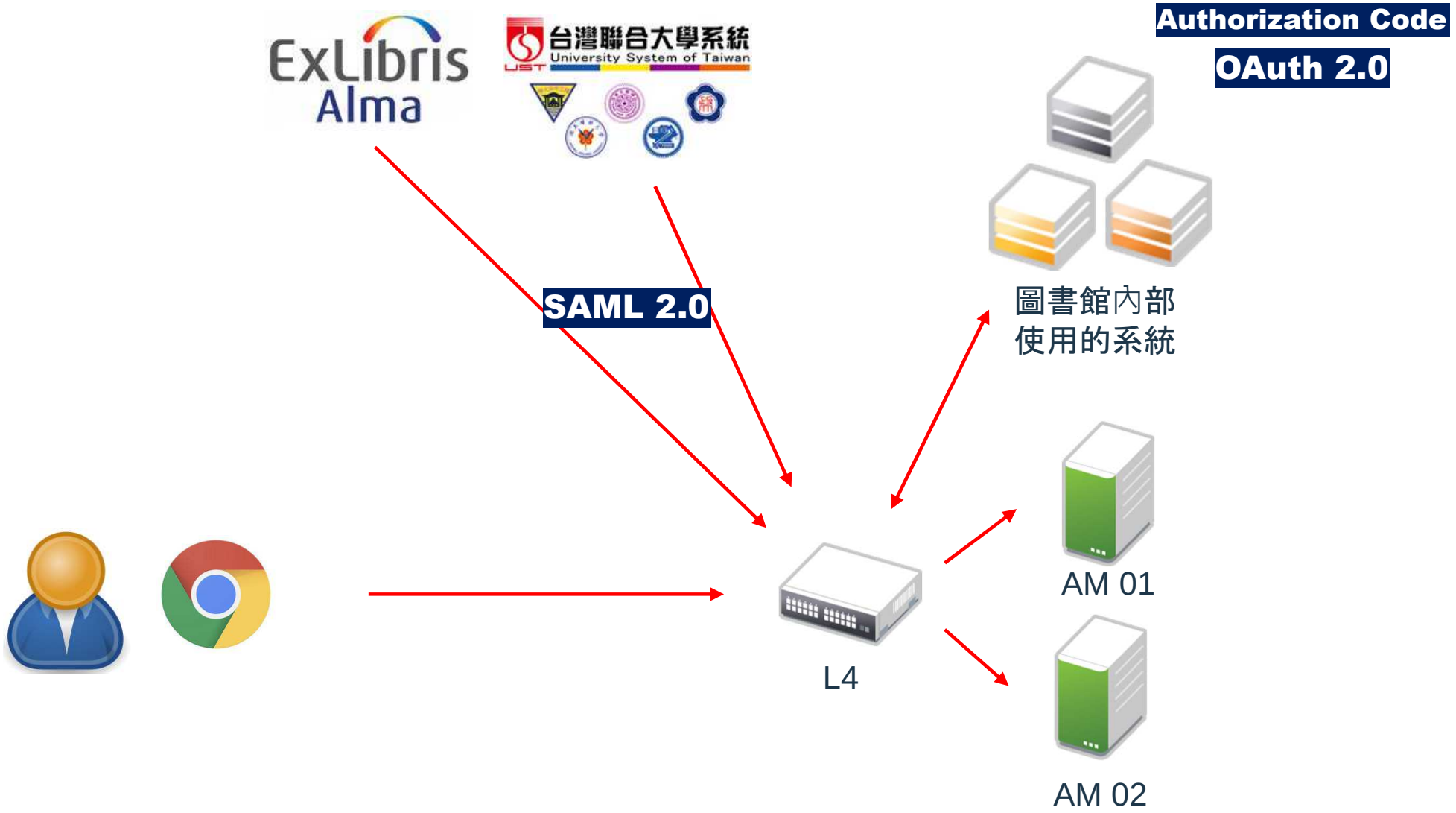
完整單一簽入規劃


DMZ
RP
單一簽入系統
IDP
Headers
OAuth, OIDC
傳統webapps
OAuth webapps
APIs
目錄服務
日誌服務
雲服務
多因子服務
API GW
OPENSHIFT
kubernetes

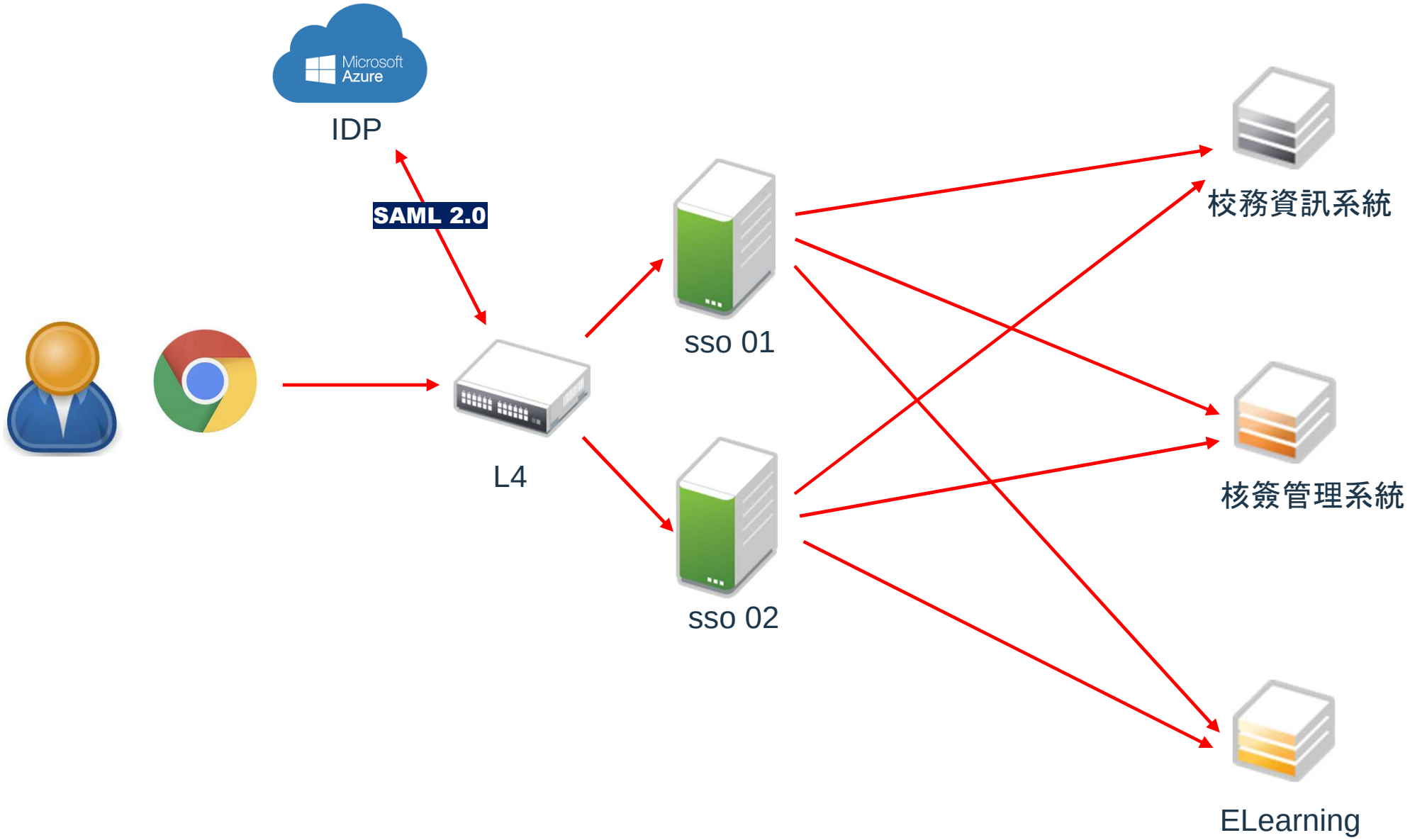
臺灣科技大學



政治大學圖書館


Authorization Code
OAuth 2.0
SAML 2.0
圖書館內部
使用的系統
AM 01
AM 02
L4

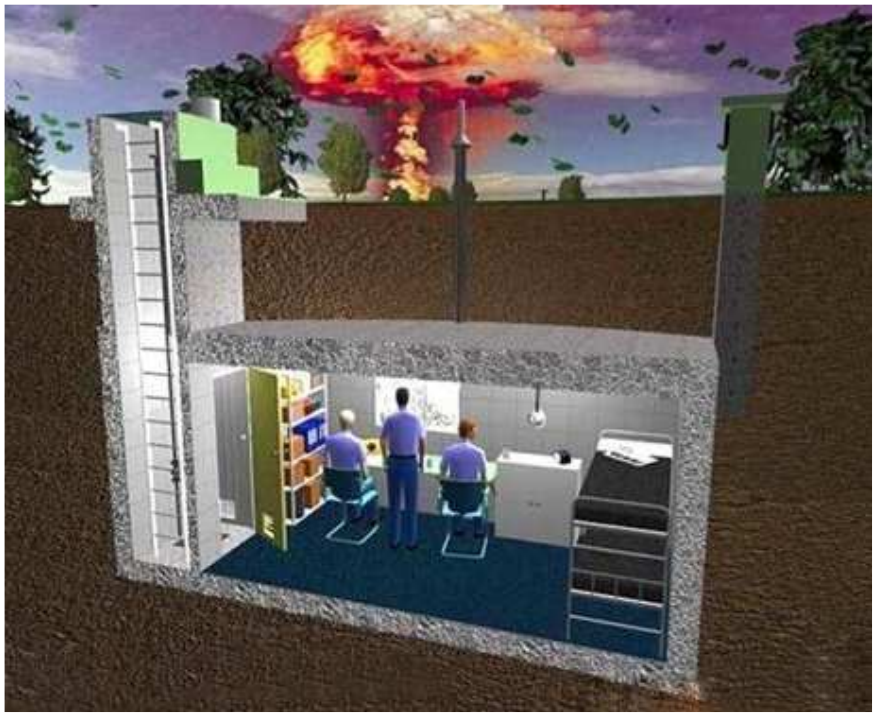
長庚大學


Microsoft Azure
IDP
SAML 2.0
L4
sso 01
sso 02
校務資訊系統
核簽管理系統
ELearning

單一簽入資安架構規劃

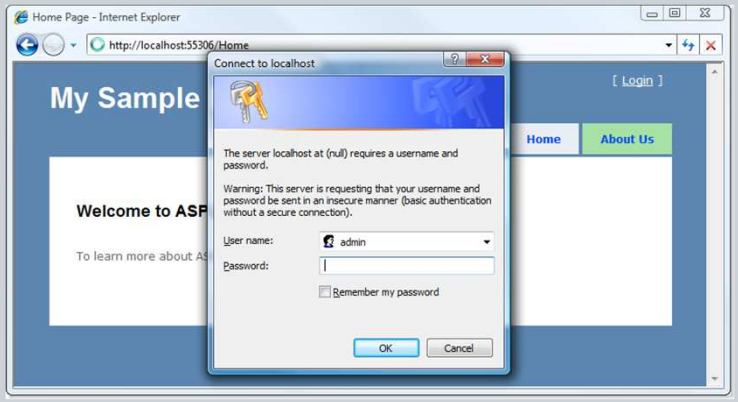
監視器要裝哪裡

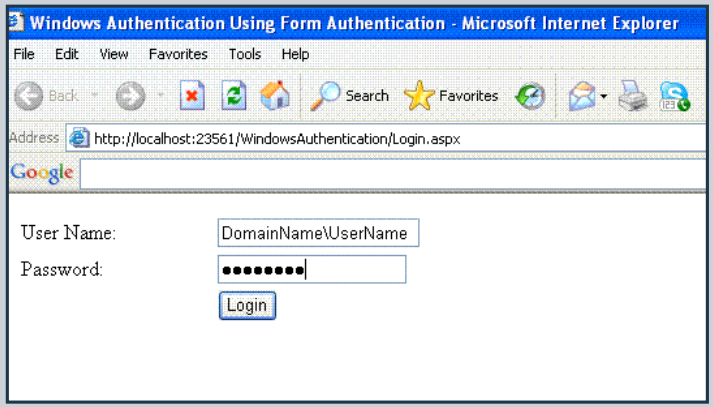
保護範圍與防護措施



網頁應用系統資源

傳統web應用系統





雲服務web應用系統

零信任架構 ZTA

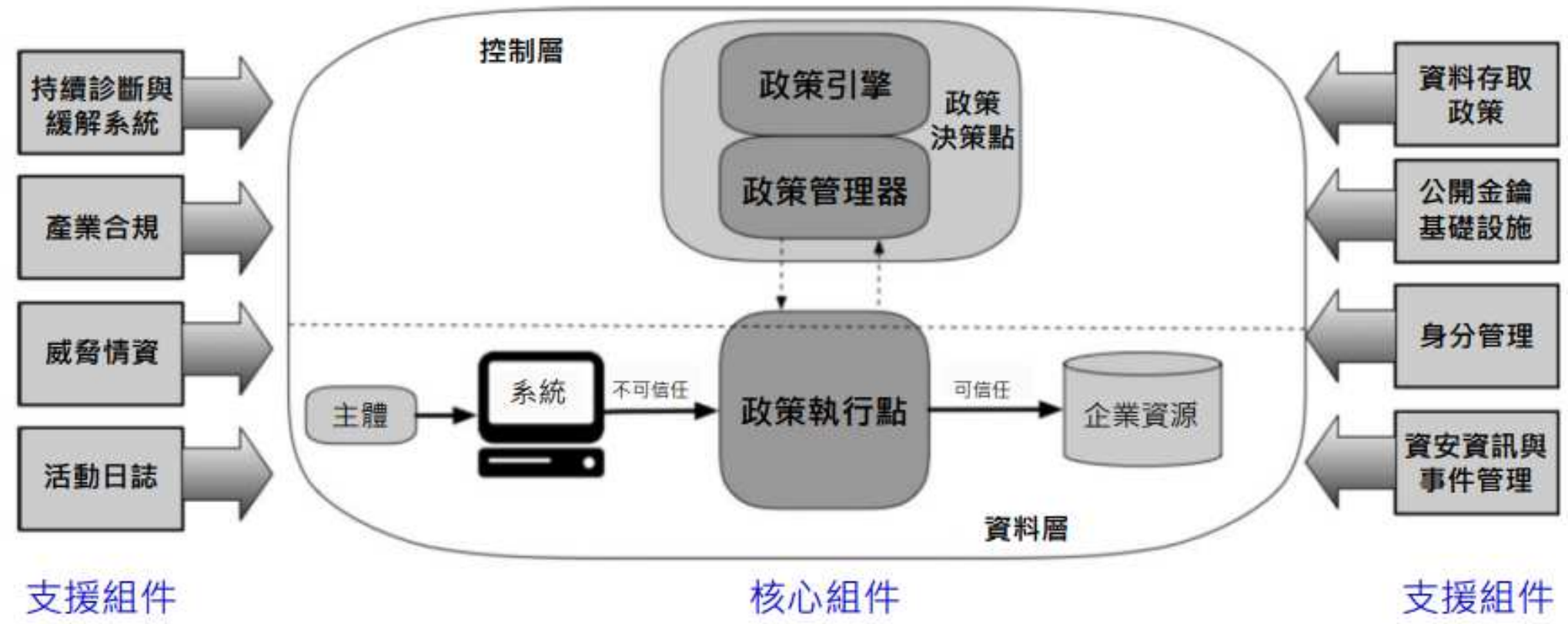
零信任安全模型（英語：Zero trust security model），也稱零信任架構、零信任網路架構、ZTA 、ZTNA等，還有時稱為無邊界安全（perimeterless security），此概念描述了一種IT系統設計與實施的方法。零信任安全模型的主要概念是「**從不信任，總是驗證**」，即不應預設信任裝置，即使裝置已經連接到經許可的網路（例如公司區域網路）並且之前已通過驗證。大多數現代企業網路結構複雜，包含眾多相互連接的區域、雲服務以及基礎設施，以及與遠端和移動環境的連接、非常規IT連接（例如物聯網裝置）。零信任原則是因傳統的方法（如在名義上的「企業邊界」內信任裝置，或者裝置通過VPN進行連接）不切合企業網路的環境複雜性。零信任提倡相互認證，包括在不考慮位置的前提下檢查裝置身分和完整性，以及基於裝置身分和裝置狀況的置信度來結合使用者身分驗證，提供對應用程式和服務的訪問許可。

Implementing a Zero Trust Architecture

- NIST SP 800-207零信任架構：Zero Trust Architecture

1) NIST SP 1800-35A：執行摘要（ Executive Summary ）
2) NIST SP 1800-35B：方法、架構和安全特性（ Approach, Architecture, and Security Characteristics ）
3) NIST SP 1800-35C：如何操作指引（ How-To Guides ）
4) NIST SP 1800-35D：功能演示（ Functional Demonstrations ）
5) NIST SP 1800-35E：風險與合規管理 （Risk and Compliance Management ）

國家資通安全研究院 -政府零信任架構說明文件



政府零信任架構說明文件 – 第6頁

ZTA Supporting Component

1. ICAM: Identity, credential, and access management
2. Endpoint Security
3. Data Security
4. Security Analytics
5. Resource Protection

NIST SP 1800- 35B : Implementing a Zero Trust Architecture – 第55頁

ZTA Component for SSO

- ICAM:
  - Access and credential management
  - Federation identity
  - Multi-Factor Authentication
  - *Identity management*
  - *Identity governance*

- Resource Protection:
  - Application connector
  - Cloud workload protection
  - Cloud security posture management

NIST SP 800-207 ZERO TRUST ARCHITECTURE

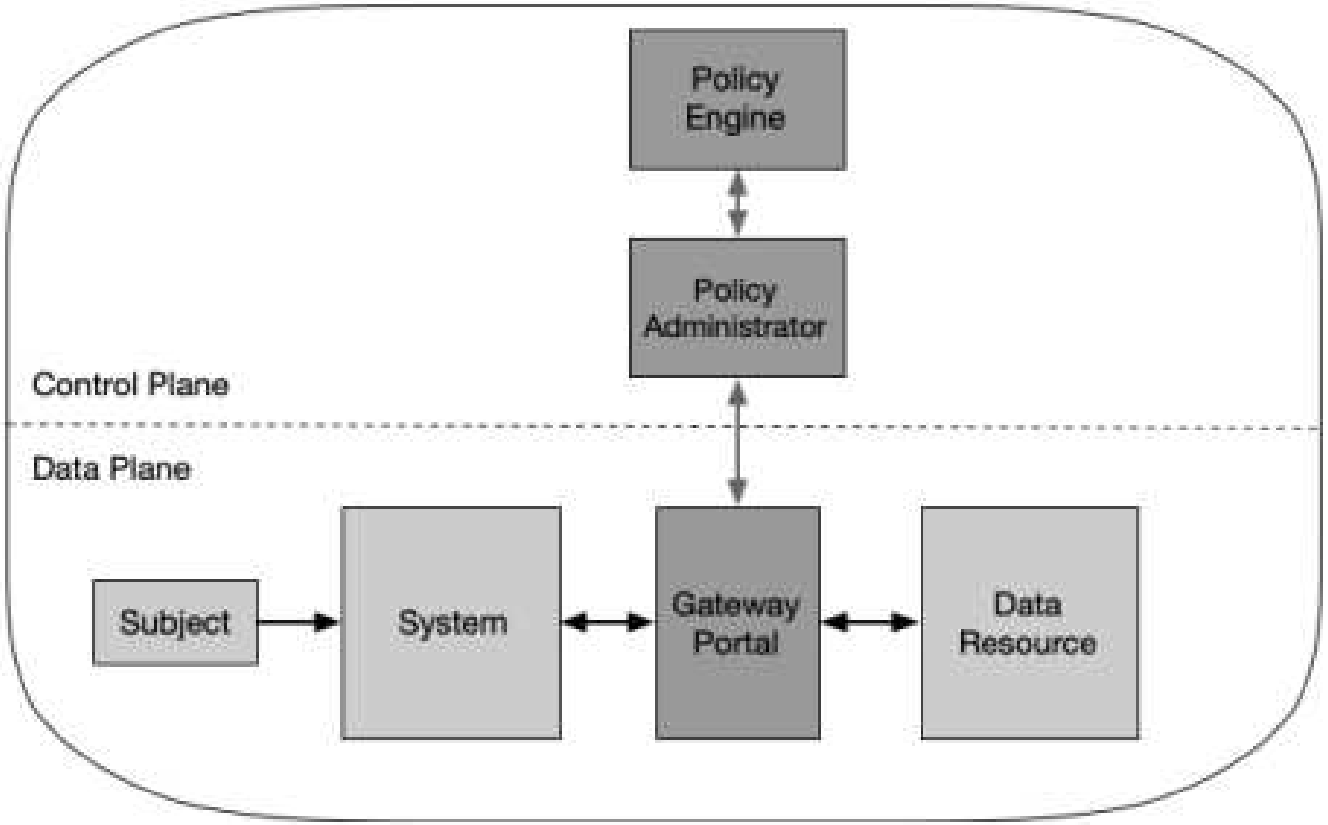
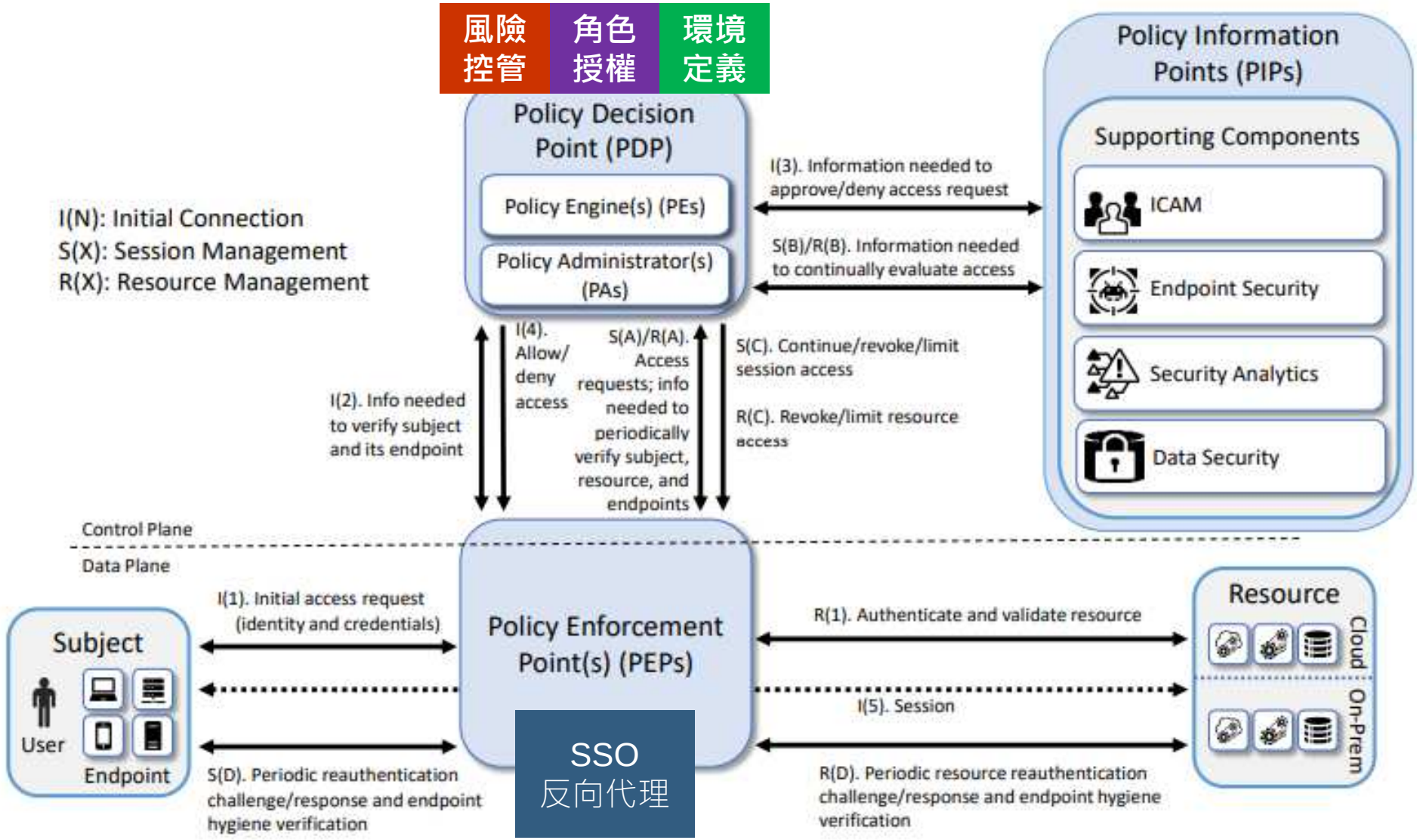


Figure 5: Resource Portal Model

NIST SP 800-207 ZERO TRUST ARCHITECTURE – 第16頁

NIST SP 1800-35B: Implementing a Zero Trust Architecture



General ZTA Reference Architecture – NIST SP 1800-35B 第54頁

零信任架構 – SSO建議的功能

Web Single Sign-On

Identity Federation

User Authentication

Access Control

API Protection

Basic Identity Administration

NIST Cybersecurity Framework CSF

辨識 IDENTIFY	保護 PROTECT	偵測 DETECT	回覆 RESPONSE	復原 RECOVER
● 資產管理 ● 營運環境 ● 治理 ● 風險評估 ● 風控管理	● 存取控制 ● 教育訓練 ● 資料安全 ● 資料保護程序 ● 維護 ● 防護技術	● 異常與事件 ● 安全監控 ● 檢測流程	● 回應計畫 ● 溝通 ● 分析 ● 緩解 ● 改善	● 復原計畫 ● 改善 ● 溝通

NIST CSF 五大功能

評估單一簽入架構設計

- 確認有哪一些應用系統
- 網路架構
- 評估應用系統的風險

- 使用人員
- 情境式認證
- 角色控管

- 稽核記錄
- 異常事件

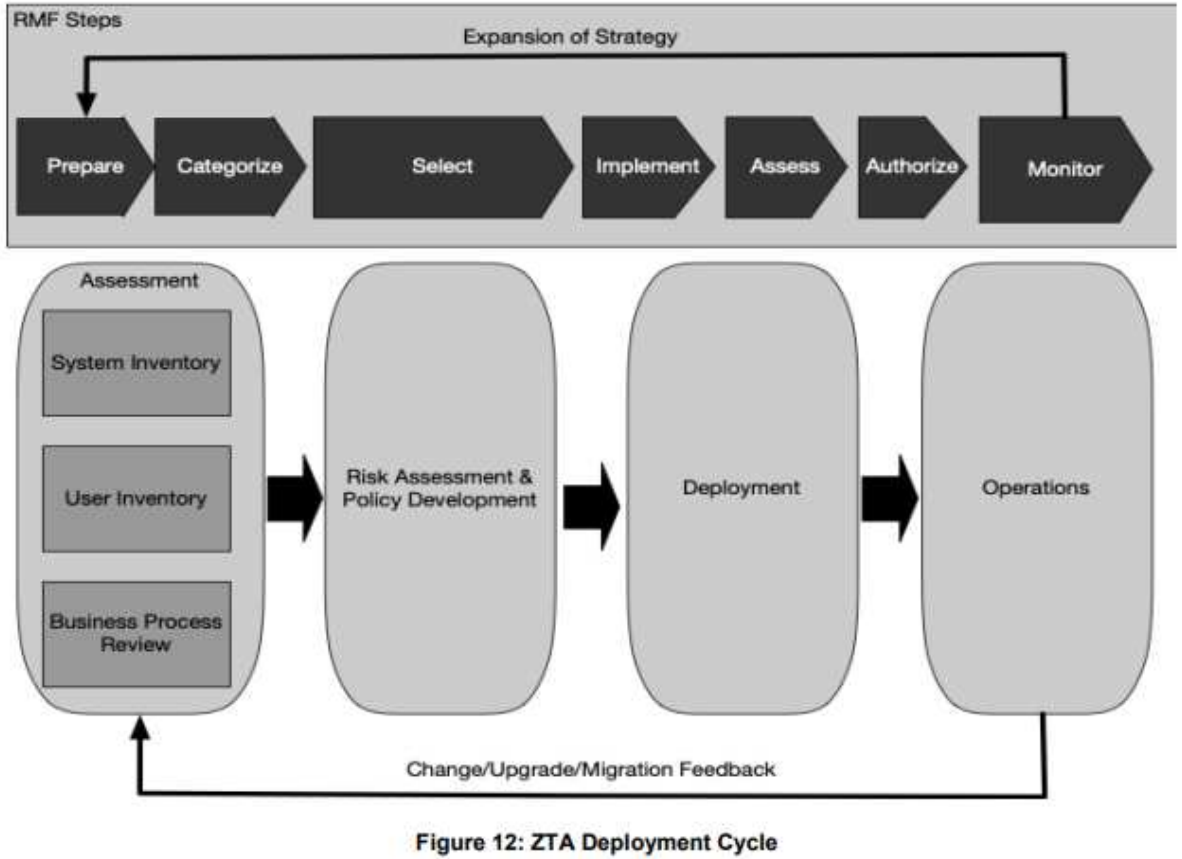
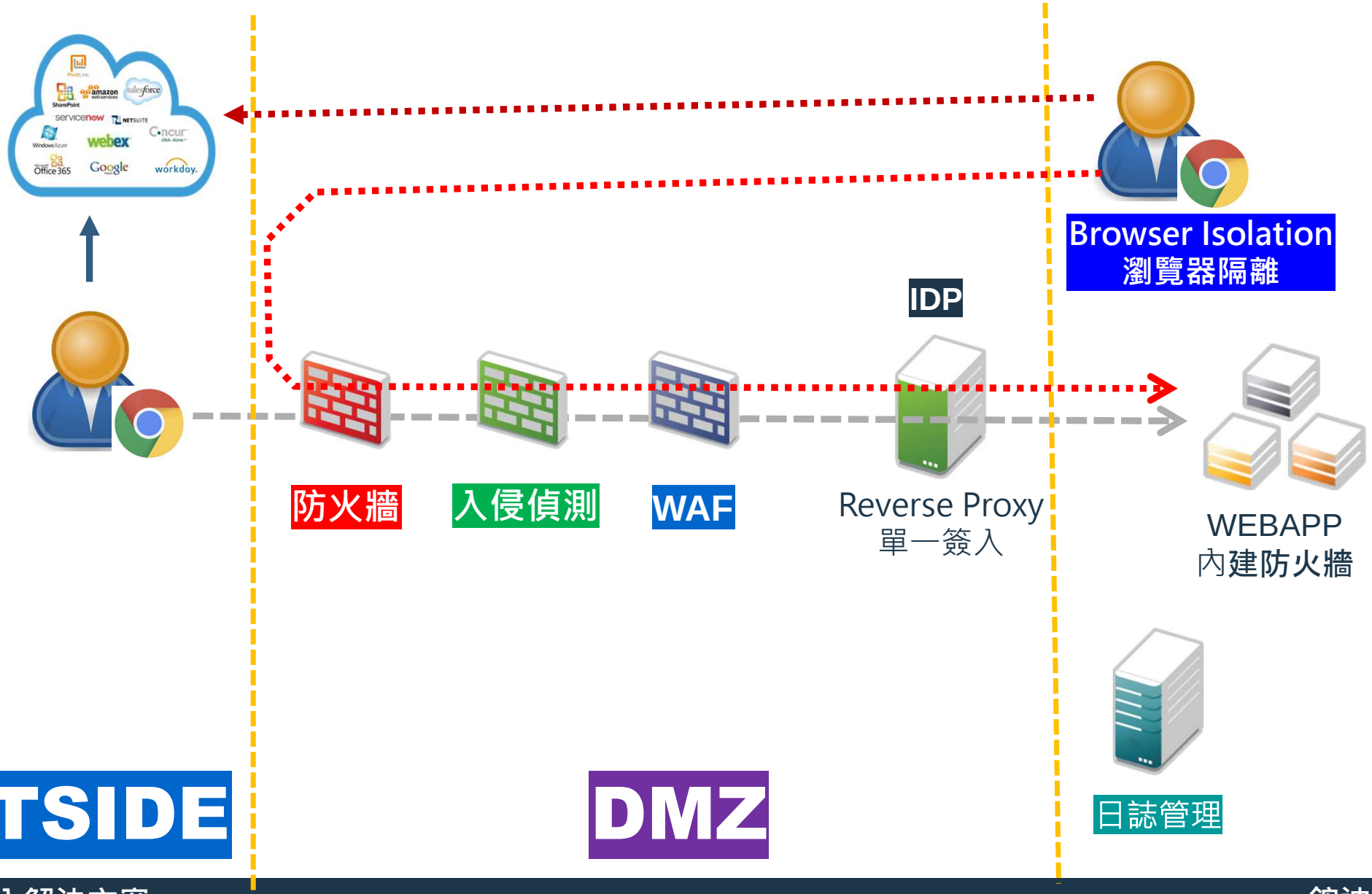


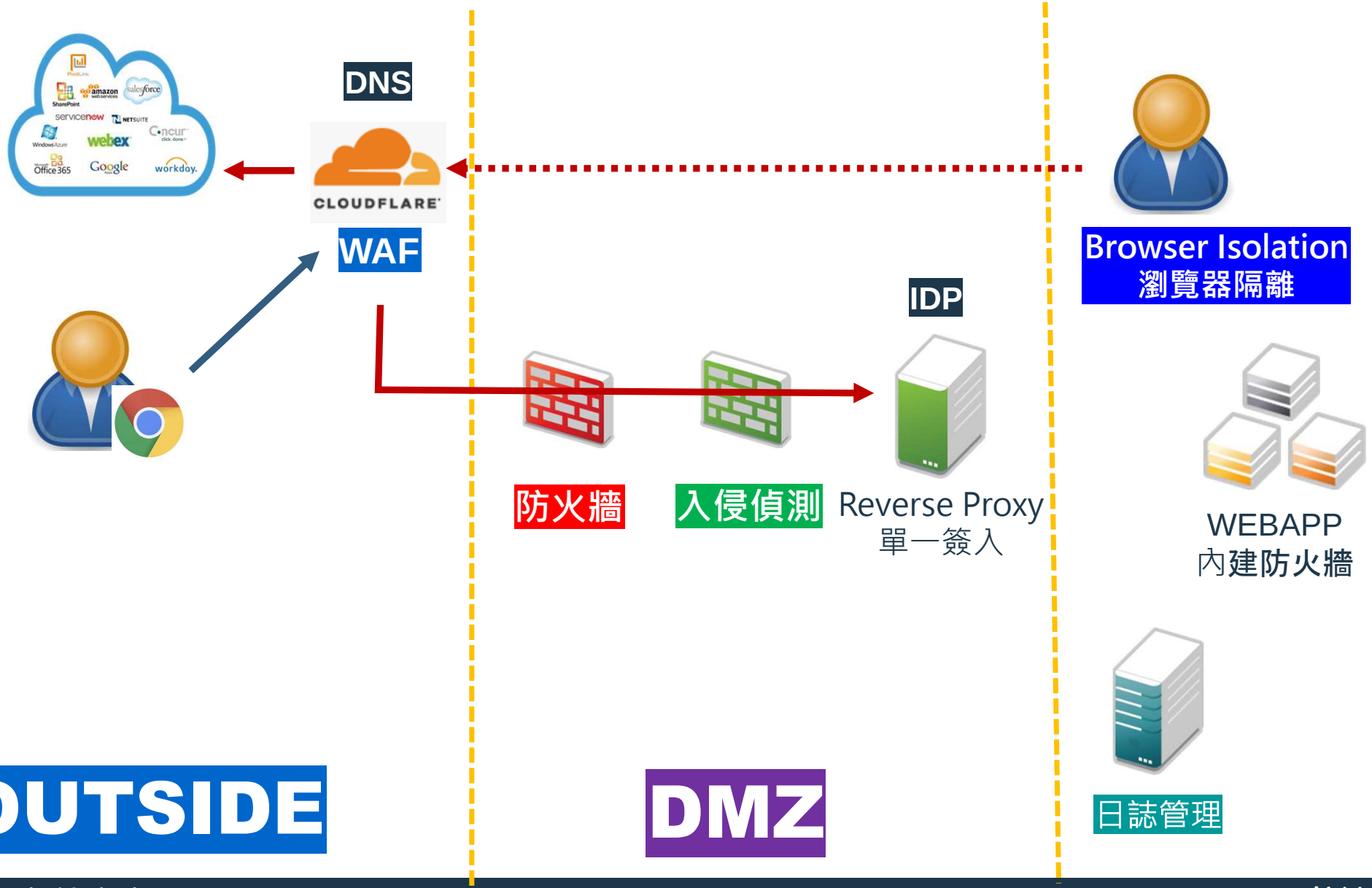
Figure 12: ZTA Deployment Cycle

NIST SP 1800-207 第34頁

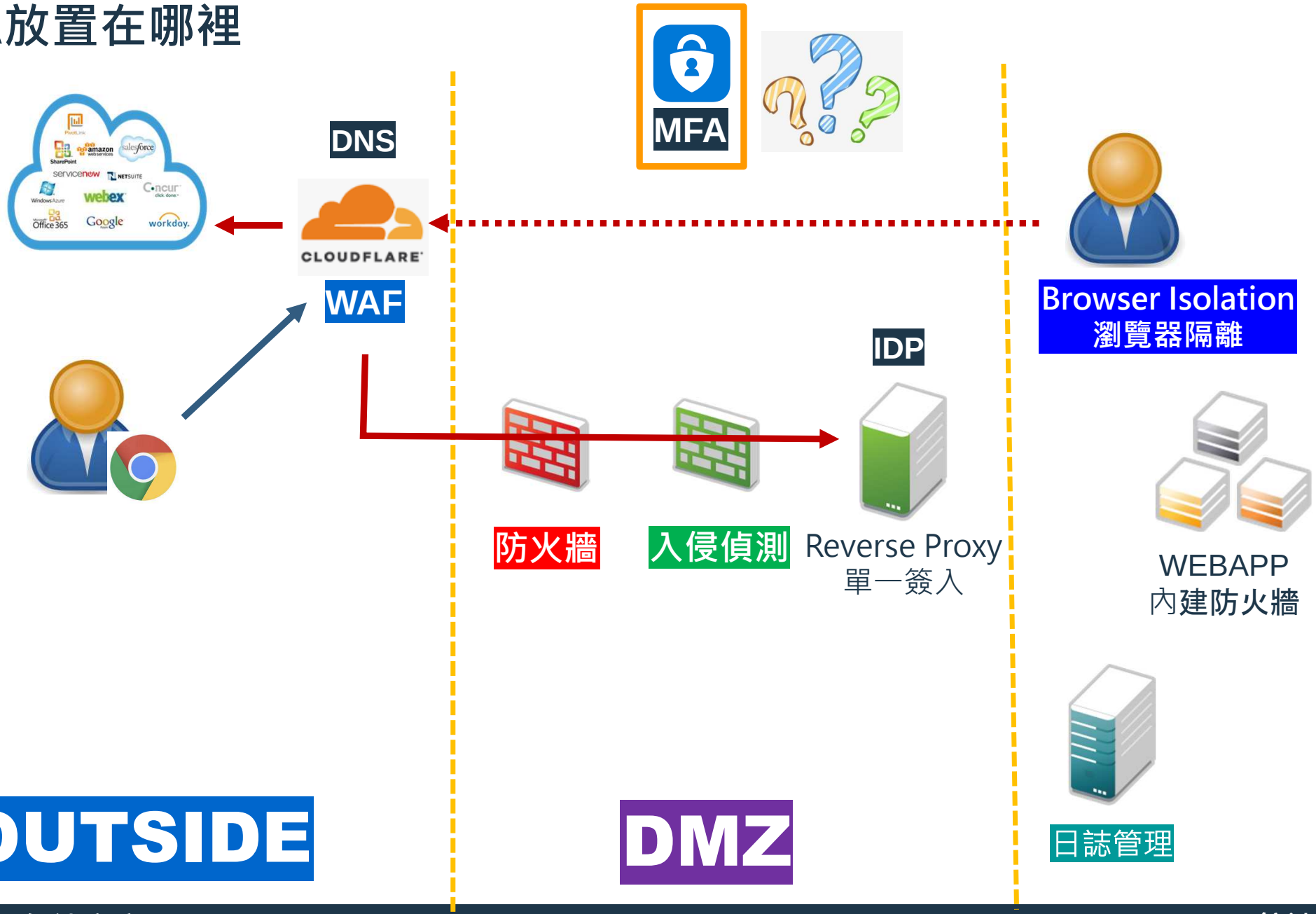
企業單一簽入設計架構


Browser Isolation
瀏覽器隔離
IDP
防火牆
入侵偵測
WAF
Reverse Proxy
單一簽入
WEBAPP
內建防火牆
日誌管理
OUTSIDE
DMZ

企業單一簽入設計架構


DNS
CLOUDFLARE
WAF
IDP
Browser Isolation
瀏覽器隔離
防火牆
入侵偵測
Reverse Proxy
單一簽入
WEBAPP
內建防火牆
日誌管理
OUTSIDE
DMZ

MFA放置在哪裡


MFA
DNS
CLOUDFLARE
WAF
IDP
Browser Isolation
瀏覽器隔離
防火牆
入侵偵測
Reverse Proxy
單一簽入
WEBAPP
內建防火牆
日誌管理
OUTSIDE
DMZ

企業SSO架構規劃

1. 決定實施範圍與優先順序
2. 確認組織目標與方向
3. 建立現況設定檔
4. 進行風險評估
5. 建立目標設定檔
6. 判定、分析和考量各種差異的優先順序
7. 實施行動計畫

永不信任，一律驗證

單一簽入解決方案

THANK YOU

如果給我6個小時砍下一顆樹，我會用前面4個小時把斧頭磨利。

Give me six hours to chop down a tree and I will spend the first four sharpening the axe.

Abraham Lincoln

鋐迪資訊