LABORATORY ON CLOUD

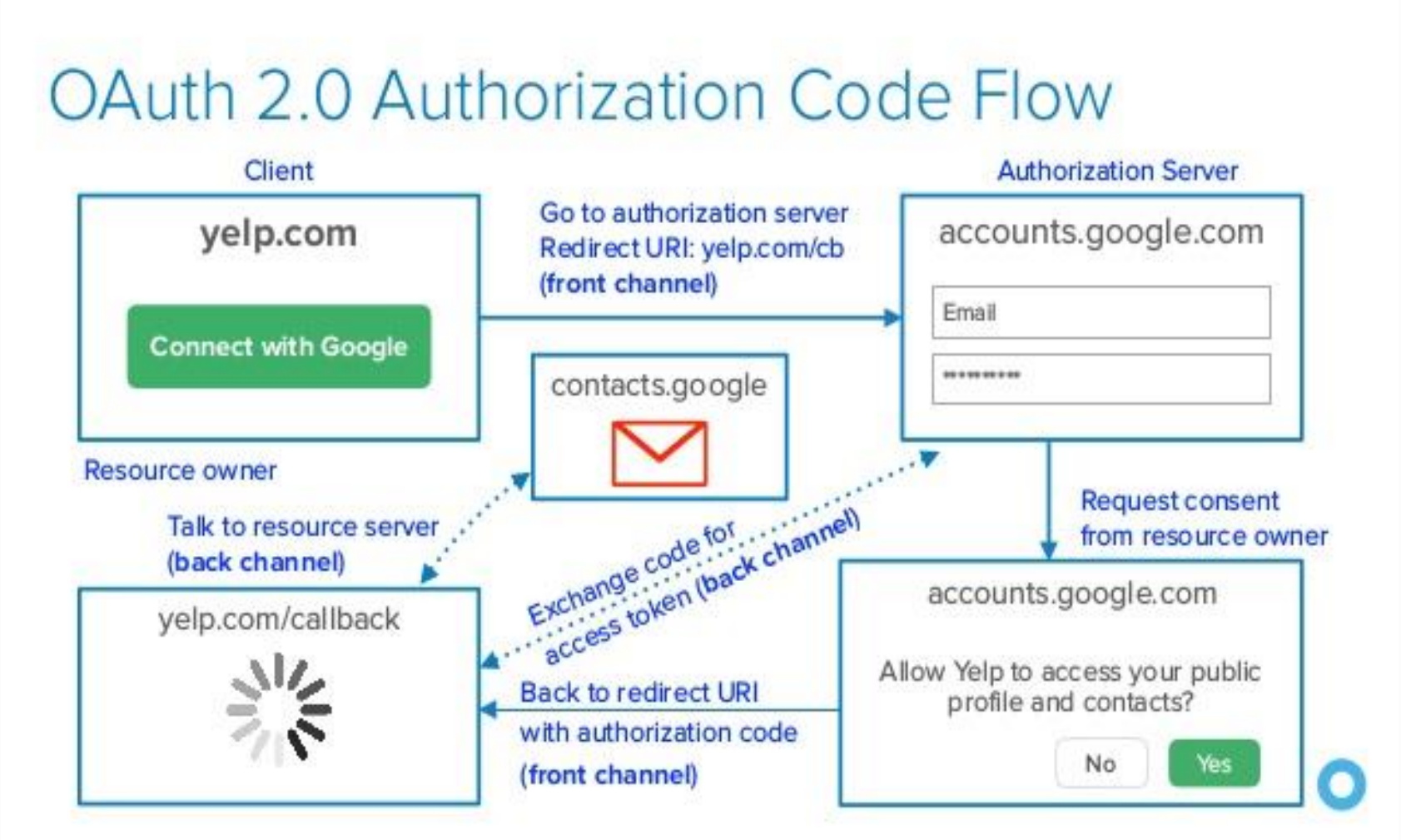
CONTENTS

การนำเสนอความคืบหน้าการดำเนินงาน

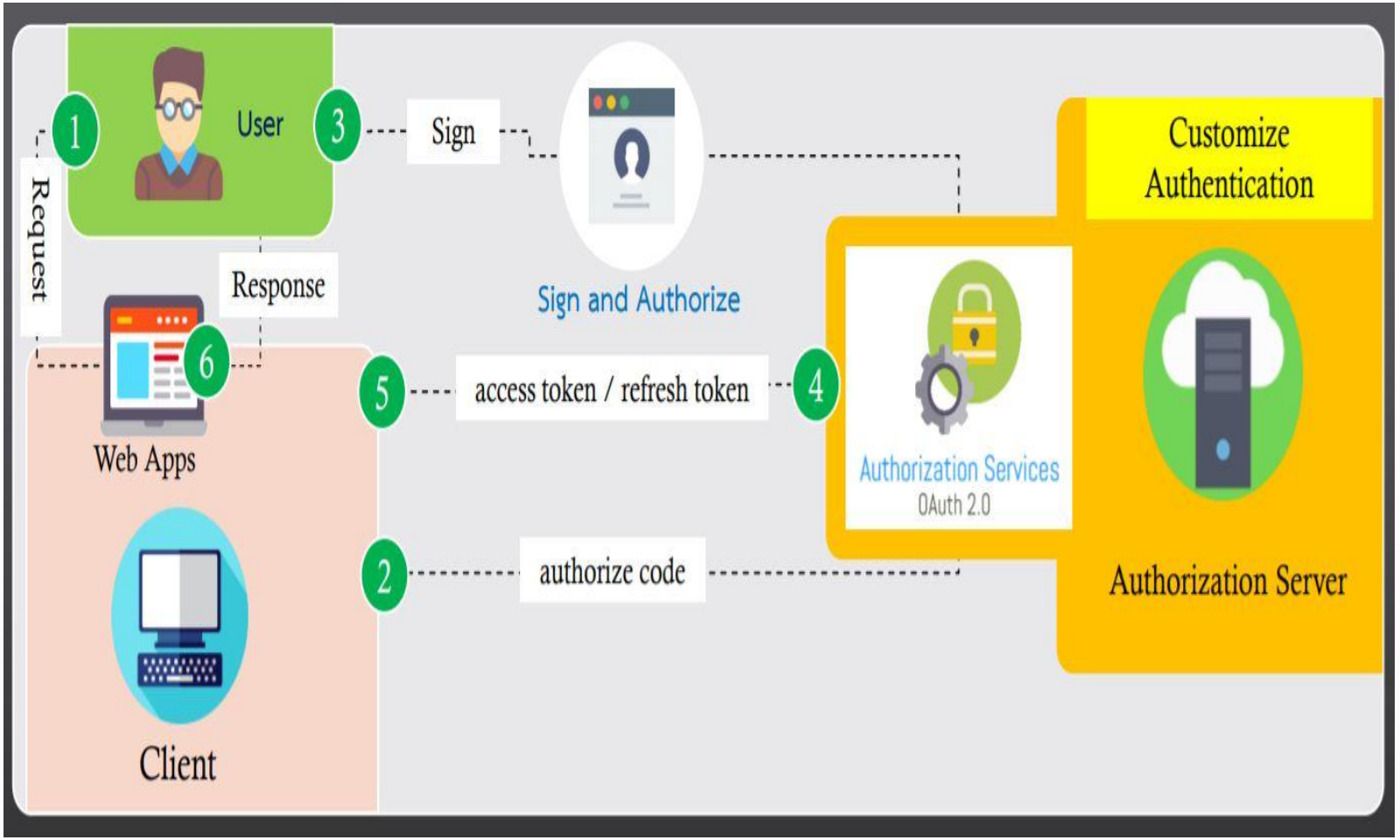
ระบบ LAB On Cloud

OAuth : Open Authentication

ปัจจุบันเป็นเวอร์ชั่น 2 หรือ OAuth 2.0 เป็นมาตรฐานที่ใช้สำหรับการกำหนดสิทธิ์ให้แอพพลิเคชั่นหนึ่งสามารถร้องขอทรัพยากรของผู้ใช้จาก แอพพลิเคชั่นหนึ่งได้ โดยที่แอพพลิเคชั่นนั้นไม่จำเป็นต้องทราบรหัสผ่านของผู้ใช้ แต่จะยืนยันว่าตัวเองมีสิทธ์หรือได้รับอนุญาตให้ใช้งานโดยใช้แอคเซสโทคเค่น (access token) แทนรหัสผ่านของผู้ใช้ ซึ่งแอคเซสโทคเค่นจะมีอายุการใช้งานเพียงช่วงเวลาหนึ่งเท่านั้น โดยแสดงภาพคร่าว ๆ เป็น Protocol Flow ดังรูป


OAuth 2.0 Authorization Code Flow
Client
yelp.com
Connect with Google
Resource owner
Go to authorization server
Redirect URI: yelp.com/cb
(front channel)
Authorization Server
accounts.google.com
Email
contacts.google
Talk to resource server
(back channel)
yelp.com/callback
Exchange code for
access token (back channel)
Back to redirect URI
with authorization code
(front channel)
Request consent
from resource owner
accounts.google.com
Allow Yelp to access your public profile and contacts?
No
Yes

Picture 1


1
User
3
Sign
Request
Response
Sign and Authorize
Customize Authentication
6
Web Apps
5
access token / refresh token
4
Authorization Services
OAuth 2.0
2
authorize code
Client
Authorization Server

Picture 2

OAuth 2.0 มีการทำงาน 4 แบบ เรียกว่า Grant_type ขึ้นอยู่กับว่าเราจะเอาไปทำ service แบบไหน อาจจะใช้แต่ละแบบร่วมกันก็ได้ ขึ้นอยู่กับบริบทของการให้บริการ (service) โดยแต่ละแบบมีรายละเอียด ดังนี้

1. Authorization Code มักจะเอาไว้ใช้กับเว็บเซิฟเวอร์ ที่ใช้ Code ด้านหลังในการเชื่อมต่อกับ OAuth Server โดยไม่ได้เปิดเผยให้สาธารณะเห็น เช่น การทำเว็บแอพพลิเคชั่น แล้วต้องการให้ผู้ใช้ล๊อกอินด้วย Gmail หรือ Facebook (ในกรณีองค์กรอาจใช้ล๊อกอินด้วยอีเมล์องค์กร) โดยโค้ดที่เป็น back-end จะใช้ grant_type

2. Implicit มักเอาไว้ใช้กับฝั่ง client ที่ไม่มี component ของ server ซึ่งไม่จำเป็นต้องมี Web Server เป็นเหมือนการคุยระหว่าง Web Browser Client กับ OAuth Server ตรง ๆ เช่น ใช้งานบน mobile application หรือ web application ที่ทำงานในลักษณะ stateless โดยในการติดต่อกับกับ server นั้น client จะไม่ต้องเก็บข้อมูลลับ (secret key) หรือเราสามารถให้ผู้ใช้ ใช้งาน web application ของเราได้โดยที่ไม่จำเป็นต้องรู้ว่า client_secret คืออะไร

3. Password Credentials ใช้สำหรับ Application ที่เป็นทางการของผู้ให้บริการเอง (ในกรณีที่เป็นผู้ให้บริการ OAuth service เอง) มีการจัดการสิทธิเอง แต่ต้องการยืนยันตัวตนเท่านั้น ซึ่งวิธีนี้ไม่ต้อง Redirect ไปที่ผู้ให้บริการอื่น วิธีนี้เหมาะกับการใช้งานที่เป็นบริการของตัวเอง เพราะ username password จะปรากฏในเครื่องที่ส่งขอ token ถ้าไปรันวิธีนี้บน Server อื่นที่ไม่ได้เป็นเจ้าของ แสดงว่าอาจจะมีการดักจับเอา username และ password ไปใช้ก็ได้

4. Client Credentials เหมาะสำหรับทำบางอย่างในแอพพลิเคชั่นโดยที่ไม่มีส่วนเกี่ยวข้องกับผู้ใช้งาน ใช้ใน code ฝั่ง clientเช่น ต้องการแก้ไขชื่อของแอพพลิเคชั่นที่ถูกลงทะเบียนไว้ หรือแก้ไขข้อมูล metadata อื่นๆ ในแอพพลิเคชั่นที่ไม่เกี่ยวข้องใดๆกับผู้ใช้เลย

อ้างอิง :

WUNCA 34 [1]การพัฒนาระบบ Authorization services provider ตามแบบวิธี OAuth 2.0 Protocol สำหรับให้บริการพิสูจน์ตัวตนเพื่อเข้าใช้งานแอพพลิเคชั่น

สิ่งที่ต้องการสนับสนุนจากผู้บริหาร

- มีหนังสือแจ้งจากเขต ขอความร่วมมือจากโรงพยาบาลต่างๆ ประสานงานกับ LIS ให้ส่งข้อมูลเข้าระบบ LAB On Cloud

Health

Thank You

Mr.Akkarapol Utbuawong