สถาบันวิจัยแสงซินโครตรอน (องค์การมหาชน)
Synchrotron Light Research Institute (Public Organization)
เอกสารความรู้ (knowledge documents)

ประเภทเอกสาร

- ☑ TR: รายงานเชิงเทคนิค (TECHNICAL REPORT)
- ☐ TN: รายงานเชิงเทคนิค (ฉบับย่อ) (TECHNICAL NOTE)
- ☐ MN: คู่มือการดำเนินงาน (Operation Manual) / คู่มือการใช้งาน (Instruction Manual) / แผนปฏิบัติการ (Operation Plan)

หมายเลขเอกสาร(For QDS) KM Document No.	SLRI-TR-2025-068
ชื่อเรื่อง Title	การติดตั้งและใช้งาน JupyterHub ซอฟต์แวร์สำหรับการวิเคราะห์ข้อมูลและควบคุมเครื่องกำเนิดแสงซินโครตรอน
ชื่อฝ่าย Department	ฝ่ายปฏิบัติการเครื่องกำเนิดแสงสยาม 1
วันที่เผยแพร่ Release date	18 สิงหาคม 2568
ระดับการเปิดเผยข้อมูล Level of Disclosure	☐ ข้อมูลในรายงานเป็นความลับ (Undisclosed)
	☐ เปิดเผยข้อมูลเฉพาะภายในฝ่ายหรือส่วนงาน (Information can be disclosed within department/section)
	☑ เปิดเผยข้อมูลได้สำหรับพนักงานของสถาบันฯ และอนุญาตให้บันทึกข้อมูลเข้าเป็นส่วนหนึ่งของระบบ Knowledge Management ภายในสถาบันฯ (Information can be disclosed for SLRI staffs and can be part of SLRI's Knowledge Management System)
	☑ เปิดเผยข้อมูลได้เพื่อเป็นองค์ความรู้สาธารณะ เช่นเว็บไซต์ของสถาบันฯ (Information is available for public)
คำสำคัญ Keyword	jupytehub, jupyterlab, jupyter, notebook, python

รายชื่อผู้จัดทำรายงานหรือผู้ดำเนินโครงการ (Name)	**ส่วนร่วมในการปฏิบัติงานในโครงการ Responsible tasks in the project**
นายณัฐวุฒิ สุรเดช	ติดตั้งและตั้งค่าระบบ, ทดสอบระบบ, จัดทำรายงาน
ดร. วรรณิสา พรมดี	ทดสอบระบบ
นายสุรกวินท์ สืบค้า	ทดสอบระบบ
ดร. ฐาปกรณ์ ภู่ลำพงษ์	ที่ปรึกษา

บทคัดย่อ

การพัฒนาและปรับใช้งานระบบ JupyterHub และ JupyterLab บนสภาพแวดล้อมเซิร์ฟเวอร์แบบหลายผู้ใช้ (Multi-User) มีความสำคัญอย่างยิ่งต่อองค์กรและสถาบันที่ต้องการรองรับการใช้งานเชิงวิชาการ การวิจัย และการประมวลผลข้อมูลขนาดใหญ่ (Data Analysis) บทความนี้มีวัตถุประสงค์เพื่ออธิบายแนวทางการติดตั้งและปรับแต่ง JupyterHub ให้สามารถบริหารจัดการผู้ใช้งานหลายคนได้อย่างปลอดภัยและมีประสิทธิภาพ โดยครอบคลุมตั้งแต่การกำหนดค่าระบบ การจัดการผู้ใช้ การเชื่อมต่อกับระบบ Authentication และการตั้งค่า Workflow ที่เหมาะสม

ในกระบวนการดำเนินงาน ได้มีการเลือกใช้แนวคิดการออกแบบระบบแบบรวมศูนย์ (Centralized System Management) เพื่อให้ผู้ดูแลสามารถควบคุมการเข้าถึงทรัพยากรได้อย่างง่ายดาย พร้อมทั้งใช้ JupyterHub เป็นแกนกลางในการจัดสรร Session ของผู้ใช้แต่ละราย โดยรองรับการทำงานร่วมกับ JupyterLab ซึ่งเป็นสภาพแวดล้อมการเขียนโค้ดเชิงโต้ตอบ (Interactive Computing Environment) ที่สามารถใช้งานได้หลากหลายภาษา เช่น Python, R เป็นต้น ทั้งนี้ยังมีการวิเคราะห์ผลการทำงานจากงานวิจัยและการปรับใช้จริง เพื่อพัฒนาวิธีการที่เหมาะสมสำหรับการใช้งานในองค์กร

ผลการศึกษาแสดงให้เห็นว่าการปรับใช้งาน JupyterHub บนระบบ Linux พร้อมการกำหนดสิทธิ์ผู้ใช้และกลไก Authentication ที่รัดกุม สามารถเพิ่มความปลอดภัยและความยืดหยุ่นในการใช้งานได้อย่างมีประสิทธิภาพ อีกทั้งยังช่วยเพิ่มความสะดวกในการทำงานร่วมกันระหว่างผู้ใช้งานหลายคน โดยไม่ต้องติดตั้งซอฟต์แวร์ซ้ำในแต่ละเครื่อง การใช้งาน JupyterLab บน JupyterHub ยังช่วยให้การทำงานด้าน Data Science, Machine Learning และการพัฒนาโปรแกรมสามารถดำเนินไปได้อย่างรวดเร็วและมีมาตรฐานสูง เหมาะสำหรับการนำไปปรับใช้ในสถาบันการศึกษา หน่วยงานวิจัย และองค์กรที่ต้องการระบบคำนวณแบบรวมศูนย์ในสภาพแวดล้อมที่ปลอดภัยและปรับขยายได้ง่าย การประยุกต์ใช้ JupyterHub ในครั้งนี้จึงเป็นก้าวสำคัญในการนำเทคโนโลยีที่ทันสมัยมาเพิ่มประสิทธิภาพให้กับงานวิจัยและงานควบคุมเครื่องกำเนิดแสงซินโครตรอนได้อย่างแท้จริง

คำสำคัญ Keyword: jupyterhub, jupyterlab, jupyter, notebook, python

1. บทนำ

บทความนี้จะอธิบายวิธีการติดตั้งและตั้งค่า JupyterHub เพื่อใช้งานร่วมกับระบบควบคุมเครื่องกำเนิดแสงซินโครตรอน โดยมีวัตถุประสงค์หลักคือการสร้างสภาพแวดล้อมการทำงานแบบรวมศูนย์สำหรับผู้ใช้งานหลายคน (Multi-user) เพื่อใช้ในการควบคุม, แสดงค่า, วิเคราะห์ข้อมูล, และสร้างแบบจำลองจากข้อมูลของเครื่องกำเนิดแสงซินโครตรอน โดยผู้ใช้สามารถรันโค้ด Python และภาษาอื่น ๆ ผ่านเบราว์เซอร์ โดยไม่ต้องติดตั้งซอฟต์แวร์เพิ่มเติมในเครื่องตนเอง ซึ่งจะช่วยเพิ่มประสิทธิภาพและความสะดวกในการทำงานร่วมกันของทีมวิจัยได้อย่างมาก วิธีการดำเนินงาน

JupyterHub เป็นแพลตฟอร์มที่นำความสามารถของ Jupyter Notebook มาให้กลุ่มผู้ใช้ใช้งานร่วมกัน โดยเปิดโอกาสให้ผู้ใช้เข้าถึงสภาพแวดล้อมการประมวลผลและทรัพยากรคอมพิวเตอร์ โดยไม่ต้องกังวลเกี่ยวกับการติดตั้งและการดูแลระบบ ผู้ใช้งานซึ่งอาจเป็นนักศึกษา นักวิจัย หรือ Data Scientist สามารถทำงานในพื้นที่ของตนเองบนทรัพยากรที่ใช้ร่วมกัน ซึ่งผู้ดูแลระบบสามารถจัดการได้อย่างมีประสิทธิภาพ

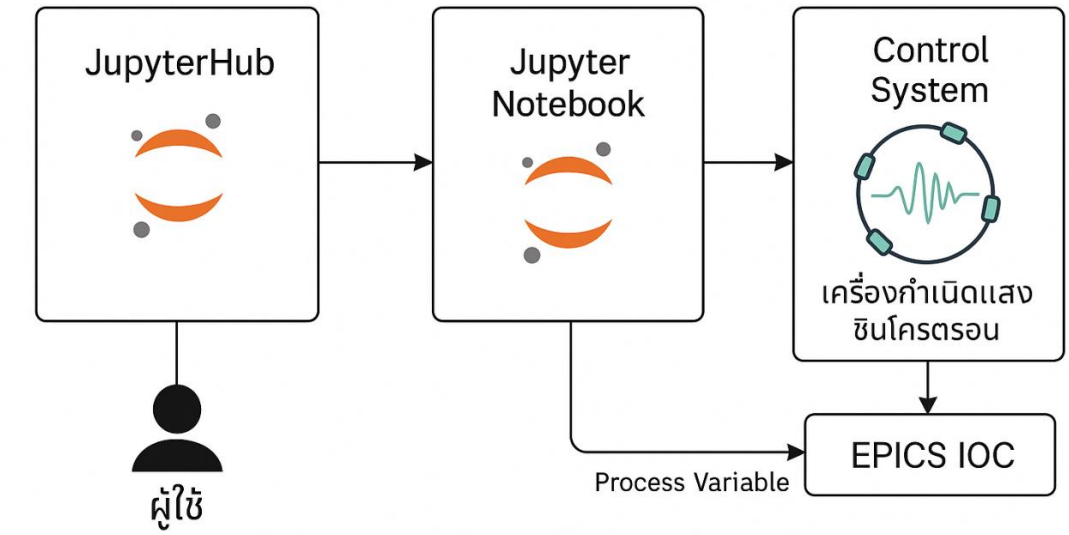
JupyterHub ช่วยให้ผู้ใช้หลายกลุ่มเข้าถึงสภาพแวดล้อมการคำนวณและทรัพยากรต่างๆ ได้โดยไม่ต้องยุ่งยากกับการติดตั้งและบำรุงรักษา ผู้ใช้งานไม่ว่าจะเป็นนักเรียน นักวิจัย หรือนักวิทยาศาสตร์ข้อมูล สามารถทำงานในพื้นที่ของตนเองบนทรัพยากรที่ใช้ร่วมกัน ซึ่งผู้ดูแลระบบสามารถจัดการได้อย่างมีประสิทธิภาพ JupyterHub สามารถรันได้ทั้งบนคลาวด์หรือฮาร์ดแวร์ของคุณเอง ทำให้สามารถให้บริการสภาพแวดล้อมทางวิทยาศาสตร์ข้อมูลที่กำหนดค่าไว้ล่วงหน้าแก่ผู้ใช้ใดๆ ก็ได้ทั่วโลก นอกจากนี้ JupyterHub ยังปรับแต่งและปรับขนาดได้ เหมาะสำหรับทีมขนาดเล็กและขนาดใหญ่, การเรียนการสอน, รวมถึงโครงสร้างพื้นฐานขนาดใหญ่

คุณสมบัติหลักของ JupyterHub:

- **ปรับแต่งได้ (Customizable):** JupyterHub สามารถใช้เพื่อให้บริการสภาพแวดล้อมที่หลากหลายได้ โดยรองรับการใช้งานเคอร์เนลต่างๆ มากมายผ่าน Jupyter server และสามารถใช้กับ User Interface ที่หลากหลาย เช่น Jupyter Notebook, JupyterLab, RStudio, nteract และอื่นๆ
- **ยืดหยุ่น (Flexible):** สามารถตั้งค่า JupyterHub ด้วยระบบยืนยันตัวตน (Authentication) เพื่อจำกัดการเข้าถึงให้กับผู้ใช้บางกลุ่ม โดยระบบยืนยันตัวตนนี้สามารถปรับเปลี่ยนได้และรองรับโปรโตคอลต่างๆ เช่น OAuth และ GitHub
- **ปรับขนาดได้ (Scalable):** JupyterHub รองรับการทำงานแบบคอนเทนเนอร์ (Container-friendly) และสามารถติดตั้งใช้งานร่วมกับเทคโนโลยีคอนเทนเนอร์ที่ทันสมัยได้ นอกจากนี้ยังสามารถรันบน Kubernetes และรองรับผู้ใช้งานได้สูงสุดถึงหลายหมื่นคน

- **สะดวกใช้งาน (Portable):** JupyterHub เป็นโอเพนซอร์สทั้งหมดและถูกออกแบบมาให้สามารถรันบนโครงสร้างพื้นฐานที่หลากหลาย ไม่ว่าจะเป็นผู้ให้บริการคลาวด์เชิงพาณิชย์, เครื่องเสมือน (Virtual Machines), หรือแม้แต่บนเครื่องแล็ปท็อปของคุณเอง

เพื่อให้ผู้อ่านเห็นภาพรวมการทำงานและองค์ประกอบของระบบบทความนี้ได้นำเสนอ แผนผังระบบและ Workflow ของ JupyterHub ซึ่งแสดงการเชื่อมต่อระหว่างผู้ใช้งาน, และการทำงานร่วมกับระบบควบคุมเครื่องกำเนิดแสงซินโครตรอน ดังแสดงใน ภาพที่ 1



ภาพที่ 1 แผนผังระบบและ Workflow การทำงานของ JupyterHub และการเชื่อมต่อกับระบบควบคุมเครื่องกำเนิดแสงซินโครตรอน

2. วัตถุประสงค์

บทความนี้มีวัตถุประสงค์หลักเพื่อนำเสนอวิธีการติดตั้งและใช้งาน JupyterHub บนระบบปฏิบัติการ Ubuntu Server 24.04 เพื่อสร้างสภาพแวดล้อมการทำงานแบบรวมศูนย์ (Centralized Multi-user Environment) สำหรับการวิเคราะห์ข้อมูลและการควบคุมระบบที่ซับซ้อน โดยเฉพาะอย่างยิ่งในบริบทของการเชื่อมต่อกับระบบควบคุมเครื่องกำเนิดแสงซินโครตรอน เพื่อให้บรรลุเป้าหมายดังกล่าว วัตถุประสงค์ย่อยที่สำคัญมีดังนี้

2.1 สร้างสภาพแวดล้อมการทำงานแบบรวมศูนย์ที่ปลอดภัยและเข้าถึงได้ง่าย

- ลดภาระในการติดตั้งและบำรุงรักษา: ผู้ใช้งานไม่จำเป็นต้องติดตั้งซอฟต์แวร์ที่ซับซ้อน เช่น Python, JupyterLab, หรือไลบรารีทางวิทยาศาสตร์ต่างๆ บนเครื่องคอมพิวเตอร์ของตนเอง ซึ่งช่วยลดปัญหาความเข้ากันได้ของระบบปฏิบัติการและ dependencies ที่ยุ่งยาก
- จัดการผู้ใช้และสิทธิ์การเข้าถึงอย่างมีประสิทธิภาพ: ตั้งค่า JupyterHub ให้สามารถจัดการบัญชีผู้ใช้งาน, กำหนดสิทธิ์การเข้าถึง (Admin/User), และยืนยันตัวตนด้วยระบบที่มีอยู่แล้ว (PAM Authenticator) ซึ่งช่วยเพิ่มความปลอดภัยและง่ายต่อการบริหารจัดการ
- อำนวยความสะดวกในการเข้าถึง: ใช้ Nginx เป็น Reverse Proxy เพื่อให้ผู้ใช้สามารถเข้าถึง JupyterHub ผ่าน URL มาตรฐานได้ทันที โดยไม่ต้องระบุพอร์ตที่ซับซ้อน ทำให้การใช้งานง่ายขึ้น

2.2 ส่งเสริมการทำงานร่วมกันและแบ่งปันข้อมูล

- สร้างพื้นที่ทำงานร่วมกัน: ตั้งค่า Shared Directory เพื่อให้ผู้ใช้ในทีมวิจัยสามารถแชร์ข้อมูล, Notebooks, และผลการวิเคราะห์ร่วมกันได้อย่างรวดเร็วและปลอดภัย โดยใช้สิทธิ์การเข้าถึงแบบกลุ่ม (jupyterhub-users) เพื่อจัดการข้อมูล
- เป็นแพลตฟอร์มสำหรับการสอนและการฝึกอบรม: ใช้ JupyterHub เป็นเครื่องมือในการจัดการชั้นเรียนหรือการอบรมเชิงปฏิบัติการทางวิทยาศาสตร์ ทำให้ผู้เรียนทุกคนสามารถเข้าถึงเครื่องมือและข้อมูลการทดลองที่เหมือนกันได้อย่างพร้อมเพรียงกัน

2.3 เพิ่มขีดความสามารถในการควบคุมและวิเคราะห์ข้อมูล

- เชื่อมต่อโดยตรงกับระบบควบคุม: ติดตั้งและตั้งค่าไลบรารีเฉพาะทาง เช่น pyepics และ pyvisa เพื่อให้ผู้ใช้งานสามารถเขียนโค้ด Python ใน Jupyter Notebook เพื่อสื่อสาร, อ่านค่า, และส่งคำสั่งควบคุมไปยังอุปกรณ์ของเครื่องซินโครตรอนได้โดยตรง
- สร้างสภาพแวดล้อมที่พร้อมใช้งานสำหรับการวิเคราะห์ข้อมูล: ติดตั้งแพ็กเกจที่จำเป็นสำหรับการจัดการข้อมูล, การคำนวณเชิงวิทยาศาสตร์, และการสร้างภาพข้อมูล เช่น pandas, scipy, matplotlib, และ plotly เพื่อให้ผู้ใช้งานสามารถเริ่มต้นการวิเคราะห์ข้อมูลเชิงลึกได้ทันที

3. แนวคิด/ทฤษฎี/หลักการ

การพัฒนาและปรับปรุงระบบ JupyterHub สำหรับการใช้งานร่วมกันของหลายผู้ใช้ (Multi-user Environment) ในสถาบันวิจัยหรือองค์กร มีรากฐานมาจากแนวคิดและทฤษฎีสำคัญหลายประการ รวมถึงผลการวิจัยและเทคโนโลยีที่มีการพัฒนาในระดับสากล ซึ่งสามารถสรุปได้ดังนี้

3.1 แนวคิดหลักการประมวลผลเชิงโต้ตอบ (Interactive Computing)

แนวคิดนี้มีจุดมุ่งหมายเพื่อให้ผู้ใช้สามารถ โต้ตอบกับข้อมูลและโค้ดได้แบบเรียลไทม์ ซึ่งแตกต่างจากการทำงานแบบสคริปต์ที่ต้องรันทั้งหมดก่อนจะเห็นผลลัพธ์ Jupyter Notebook และ JupyterLab สร้างขึ้นจากแนวคิดนี้ โดยใช้ IPython Kernel เป็นกลไกหลัก ทำให้สามารถประมวลผลโค้ดทีละเซลล์ แสดงผลลัพธ์ทันที และผสานเข้ากับการทำงานเชิงกราฟิก เช่น การสร้างกราฟ แผนภาพ และการแสดงผลข้อมูลเชิงโต้ตอบ

ตามงานวิจัยของ Pérez และ Granger (2007) การใช้ IPython Interactive System เป็นจุดเริ่มต้นของสถาปัตยกรรม Jupyter ปัจจุบัน [Pérez & Granger, 2007]

3.2 สถาปัตยกรรมแบบ Multi-User และการควบคุมสิทธิ์ (Authentication & Authorization)

JupyterHub ถูกออกแบบบนแนวคิดของ Server-Client Model โดยที่ Hub ทำหน้าที่เป็นศูนย์กลางบริหารจัดการผู้ใช้ (User Management) และ Spawner ทำหน้าที่สร้างเซสชัน Notebook ให้แต่ละผู้ใช้บนระบบ ซึ่งหลักการสำคัญที่นำมาใช้ คือ

1. การควบคุมสิทธิ์การเข้าถึง (Access Control) ผ่าน Authenticator เช่น PAM, OAuth, LDAP เพื่อให้มั่นใจว่าผู้ใช้ที่ได้รับอนุญาตเท่านั้นจึงสามารถเข้าใช้งานได้
2. การแยกสภาพแวดล้อมผู้ใช้ (User Isolation) โดยใช้การสร้างโฟลเดอร์ Home แยก หรือการรันเซสชันบน Container (DockerSpawner, KubernetesSpawner) เพื่อลดความเสี่ยงจากการเข้าถึงข้อมูลร่วมกัน

แนวคิดเหล่านี้สอดคล้องกับทฤษฎีด้าน ระบบผู้ใช้หลายคน (Multi-user System Theory) ที่มุ่งเน้นการจัดสรรทรัพยากรคอมพิวเตอร์อย่างปลอดภัยและมีประสิทธิภาพ

3.3 หลักการใช้ทรัพยากรคอมพิวเตอร์ร่วมกัน (Shared Resource Principle)

ในการศึกษานี้มีการกำหนด Shared Directory เพื่อเป็นพื้นที่กลางสำหรับเก็บข้อมูลที่ต้องการให้ผู้ใช้หลายคนเข้าถึงได้ ซึ่งยึดตามแนวคิด Collaboration in Computing และหลักการ การกำหนดสิทธิ์ไฟล์ในระบบ Linux (Linux File Permission Model) ที่อ้างอิงมาตรฐาน POSIX เพื่อควบคุมการอ่าน เขียน และรันไฟล์ร่วมกัน โดยใช้ UMASK, setgid และการกำหนดสิทธิ์แบบกลุ่ม (Group Permission)

3.4 แนวคิดการขยายความสามารถด้วย JupyterLab

JupyterLab เป็นวิวัฒนาการจาก Jupyter Notebook ที่มุ่งเน้นการสร้าง Modular Interface ทำให้ผู้ใช้สามารถปรับแต่งได้เอง เช่น การเปิดหลายแท็บ, ลากวางองค์ประกอบ, และการใช้งาน Extension เพื่อเพิ่มฟังก์ชัน เช่น Terminal, File Browser, Data Visualization Tools เป็นต้น

3.5 งานวิจัยและผลงานที่เกี่ยวข้อง

การศึกษาและพัฒนาในครั้งนี้ได้รับอิทธิพลจากผลงานวิจัยและแพลตฟอร์มที่เกี่ยวข้อง ซึ่งมีดังนี้:

- **โครงการ Jupyter Notebook:** Jupyter Notebook ได้รับการพัฒนาขึ้นโดยมีเป้าหมายเพื่อเป็นเครื่องมือสำหรับการวิจัยทางวิทยาศาสตร์ที่สามารถรวมโค้ด, ผลลัพธ์, และข้อความอธิบายไว้ในที่เดียวกัน โดยผลงานนี้ได้ถูกนำไปใช้ในวงการวิทยาศาสตร์และวิศวกรรมอย่างกว้างขวาง [5]
- **The Littlest JupyterHub (TLJH):** TLJH แสดงให้เห็นถึงแนวคิดในการทำให้การติดตั้ง JupyterHub เป็นเรื่องง่ายและเข้าถึงได้สำหรับผู้เริ่มต้น โดยใช้สถาปัตยกรรมที่เรียบง่าย ซึ่งเป็นแรงบันดาลใจในการนำไปประยุกต์ใช้กับงานที่ต้องการความซับซ้อนมากขึ้น [6]
- **งานวิจัยด้านการควบคุมเครื่องมือทางฟิสิกส์:** การใช้ระบบควบคุมมาตรฐานอย่าง **EPICS** ร่วมกับเครื่องมือทางวิทยาศาสตร์ข้อมูลอย่าง Jupyter Notebooks ได้รับการพิสูจน์แล้วว่ามีประสิทธิภาพในงานวิจัยทางฟิสิกส์พลังงานสูงและฟิสิกส์เครื่องเร่งอนุภาค [7] ซึ่งเป็นแนวทางสำคัญในการออกแบบระบบนี้

การประยุกต์ใช้แนวคิดและผลงานวิจัยเหล่านี้ ทำให้ระบบ JupyterHub สำหรับเครื่องกำเนิดแสงซินโครตรอนไม่ได้เป็นเพียงแค่การติดตั้งซอฟต์แวร์ แต่เป็นการสร้างแพลตฟอร์มที่ครอบคลุมและมีประสิทธิภาพ ซึ่งช่วยส่งเสริมการทำงานวิจัยและพัฒนาในระดับที่สูงขึ้น

4. วิธีการดำเนินงาน

การติดตั้ง JupyterHub บน Ubuntu Server เป็นขั้นตอนสำคัญสำหรับการให้บริการ Jupyter Notebook และ JupyterLab แบบ Multi-user โดยเฉพาะสำหรับงานวิจัยและวิศวกรรมที่ต้องการสภาพแวดล้อมคำนวณร่วมกัน บทความนี้จะแสดงขั้นตอนการติดตั้งอย่างละเอียด พร้อมคำอธิบายแต่ละคำสั่งว่ามีหน้าที่ทำอะไร

4.1 การติดตั้ง JupyterHub

4.1.1 ติดตั้งแพ็กเกจที่จำเป็นสำหรับระบบ

คำสั่งนี้จะติดตั้งแพ็กเกจพื้นฐานที่จำเป็นสำหรับการติดตั้งและเรียกใช้งาน JupyterHub รวมถึงเครื่องมือสำหรับจัดการแพ็กเกจและ Virtual Environment

```
sudo apt install -y python3 python3-pip python3-venv git nodejs npm
```

- sudo apt install -y: คำสั่งสำหรับติดตั้งแพ็กเกจด้วยสิทธิ์ผู้ดูแลระบบ (sudo) และยอมรับการติดตั้งทั้งหมด (-y) โดยไม่ต้องยืนยัน
- python3: ตัวแปลภาษา Python เวอร์ชัน 3
- python3-pip: เครื่องมือสำหรับติดตั้งแพ็กเกจ Python
- python3-venv: โมดูลสำหรับสร้าง Virtual Environment ของ Python ซึ่งช่วยให้เราจัดการแพ็กเกจเฉพาะโปรเจกต์ได้
- git: ระบบควบคุมเวอร์ชัน เพื่อใช้ในการดาวน์โหลดซอร์สโค้ด (แม้ว่าในตัวอย่างนี้จะไม่ได้ใช้ แต่เป็นเครื่องมือพื้นฐานที่ควรมี)
- nodejs npm: เครื่องมือสำหรับรันโค้ด JavaScript ฝั่งเซิร์ฟเวอร์และตัวจัดการแพ็กเกจ (Package Manager) ซึ่งจำเป็นสำหรับ configurable-http-proxy

4.1.2 ติดตั้ง Configurable HTTP Proxy

JupyterHub จำเป็นต้องมี Proxy Server เพื่อจัดการการรับส่งคำขอของผู้ใช้แต่ละคน โดยเราจะใช้ configurable-http-proxy ซึ่งเป็นแพ็กเกจที่เขียนด้วย Node.js

```
sudo npm install -g configurable-http-proxy
```

- sudo npm install -g: คำสั่งสำหรับติดตั้งแพ็กเกจ Node.js แบบ Global (-g) ซึ่งทำให้สามารถเรียกใช้งานได้จากทุกที่ในระบบ

4.1.3 ตั้งค่าสภาพแวดล้อมและสิทธิ์การเข้าถึง

สร้าง Virtual Environment เพื่อแยกส่วนการติดตั้ง JupyterHub และกำหนดสิทธิ์การเข้าถึงให้ผู้ใช้ที่เหมาะสม

```
sudo python3 -m venv /opt/jupyterhub
```

- sudo python3 -m venv: ใช้โมดูล venv ของ Python เพื่อสร้าง Virtual Environment
- /opt/jupyterhub: กำหนดให้ Virtual Environment ถูกสร้างขึ้นที่ไดเรกทอรีนี้ ซึ่งเป็นตำแหน่งมาตรฐานสำหรับติดตั้งซอฟต์แวร์ที่ไม่ใช่แพ็กเกจระบบ

```
sudo chown -R jupyteradmin:jupyteradmin /opt/jupyterhub/
```

- sudo chown -R: คำสั่งสำหรับเปลี่ยนเจ้าของไฟล์/ไดเรกทอรีทั้งหมด (-R หรือ Recursive)
- jupyteradmin:jupyteradmin: กำหนดให้ผู้ใช้ jupyteradmin เป็นเจ้าของและเจ้าของกลุ่มของไดเรกทอรีนี้ ทำให้ผู้ใช้ jupyteradmin มีสิทธิ์ในการติดตั้งและจัดการ JupyterHub ได้โดยไม่ต้องใช้ sudo

4.1.4 ติดตั้ง JupyterHub และ JupyterLab

เข้าสู่ Virtual Environment ที่สร้างไว้และติดตั้งแพ็กเกจที่จำเป็น

```
source /opt/jupyterhub/bin/activate
```

- source: คำสั่งสำหรับเปิดใช้งาน Virtual Environment ทำให้คำสั่งต่างๆ เช่น pip หรือ python ชี้ไปยัง Virtual Environment ที่เราสร้างไว้

```
pip install --upgrade pip
```

- pip install --upgrade pip: อัปเกรด pip ใน Virtual Environment ให้เป็นเวอร์ชันล่าสุด

```
pip install jupyterhub jupyterlab notebook
```

- pip install: ติดตั้งแพ็กเกจหลักของ JupyterHub และ JupyterLab
    - jupyterhub: ตัวจัดการหลักสำหรับผู้ใช้หลายคน

- jupyterlab: อินเทอร์เฟซผู้ใช้งานแบบใหม่ที่ทันสมัย
- notebook: อินเทอร์เฟซผู้ใช้งานแบบคลาสสิก (ติดตั้งเพื่อความเข้ากันได้)

4.1.5 สร้างไฟล์ตั้งค่าและกำหนดค่าเริ่มต้น

เราจะสร้างไฟล์ตั้งค่าเริ่มต้นและแก้ไขเพื่อกำหนดการทำงานของ JupyterHub

```
cd /opt/jupyterhub
jupyterhub --generate-config -f etc/jupyterhub_config.py
```

- jupyterhub --generate-config: คำสั่งสำหรับสร้างไฟล์ตั้งค่าเริ่มต้น
- -f etc/jupyterhub_config.py: กำหนดให้สร้างไฟล์ตั้งค่าในไดเรกทอรี etc ที่ชื่อว่า jupyterhub_config.py

*4.1.6 ออกจาก **Virtual Environment***

เมื่อติดตั้งแพ็กเกจเสร็จสิ้น เราจะออกจาก Virtual Environment เพื่อกลับสู่สภาพแวดล้อมของระบบปกติ

```
Deactivate
```

- deactivate: คำสั่งสำหรับปิดใช้งาน Virtual Environment

4.1.7 แก้ไขไฟล์ตั้งค่า

เปิดไฟล์ jupyterhub_config.py เพื่อแก้ไขการตั้งค่าสำคัญ

```
nano etc/jupyterhub_config.py
```

จากนั้นเพิ่มการตั้งค่าดังนี้:

```
c = get_config()

# ให้ JupyterHub ทำงานบนทุก IP บนพอร์ต 8000
c.JupyterHub.bind_url = 'http://:8000'

# ให้ JupyterHub ทำงานใน Subdirectory /jupyter/
c.JupyterHub.base_url = '/jupyter/'
```

```
# ใช้การยืนยันตัวตนผ่านระบบ PAM
c.JupyterHub.authenticator_class =
'jupyterhub.auth.PAMAuthenticator'

# ใช้ jupyterhub-singleuser จาก Virtual Environment นี้
c.Spawner.cmd = ['/opt/jupyterhub/bin/jupyterhub-
singleuser']

# เปิด JupyterLab เป็นหน้าแรก
c.Spawner.default_url = '/lab'

# ตั้งค่า PATH สำหรับการใช้งาน EPICS และเครื่องมืออื่น
c.Spawner.environment = {
    'PATH':
'/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bi
n:/snap/bin:/sps/prod/epics/base/bin/linux-
x86_64:/opt/jupyterhub/bin',
}

# กำหนดผู้ใช้ที่อนุญาตและผู้ดูแลระบบ
c.Authenticator.allowed_users = {'jupyteradmin',
'popcomputer'}
c.Authenticator.admin_users = {'jupyteradmin',
'popcomputer'}
```

4.1.8 เพิ่ม PATH ของ JupyterHub ในระบบ

เพื่อความสะดวกในการรัน JupyterHub เราจะเพิ่มเส้นทางของไฟล์รันใน Virtual Environment เข้าไปใน PATH ของระบบ

```
sudo nano /etc/environment
```

- :/opt/jupyterhub/bin: การเพิ่มเส้นทางนี้จะทำให้ผู้ใช้สามารถเรียกใช้คำสั่งที่อยู่ในไดเรกทอรี /opt/jupyterhub/bin ได้โดยไม่ต้องระบุเส้นทางเต็ม

4.1.9 เริ่มการทำงานของ JupyterHub

สุดท้ายคือการเริ่มการทำงานของ JupyterHub โดยใช้ไฟล์ตั้งค่าที่เราได้แก้ไขไว้

```
sudo /opt/jupyterhub/bin/jupyterhub -f
/opt/jupyterhub/etc/jupyterhub_config.py
```

- sudo /opt/jupyterhub/bin/jupyterhub: เรียกใช้โปรแกรม JupyterHub ด้วยสิทธิ์ผู้ดูแลระบบ
- -f /opt/jupyterhub/etc/jupyterhub_config.py: กำหนดให้ใช้ไฟล์ตั้งค่าที่เราได้แก้ไขไว้

4.2 การตั้งค่า JupyterHub ให้ทำงานอัตโนมัติและเชื่อมต่อผ่าน Reverse Proxy ด้วย Nginx

เมื่อทำการติดตั้งและคอนฟิก JupyterHub เรียบร้อยแล้ว ขั้นตอนต่อไปคือการทำให้บริการ JupyterHub ทำงานอัตโนมัติหลังจากการบูตเครื่อง และการตั้งค่า **Reverse Proxy** ด้วย **Nginx** เพื่อให้เข้าถึงผ่าน **HTTPS** และโดเมนเนมที่เป็นมิตรต่อผู้ใช้งาน

4.2.1 วิธีการตั้งค่าให้ JupyterHub ทำงานอัตโนมัติด้วย system

systemd คือตัวจัดการบริการและระบบเริ่มต้น (Init System) ที่ใช้กันอย่างแพร่หลายใน Linux ซึ่งเราจะใช้มันเพื่อจัดการบริการของ JupyterHub

สร้าง Service File สำหรับ JupyterHub เราจะสร้างไฟล์ service ชื่อ jupyterhub.service ในไดเรกทอรี /etc/systemd/system/ ซึ่งเป็นที่เก็บไฟล์บริการของระบบ

```
sudo nano /etc/systemd/system/jupyterhub.service
```

คัดลอกและวางเนื้อหาต่อไปนี้ลงในไฟล์:

```
[Unit]
Description=JupyterHub
After=network.target

[Service]
Type=simple
User=jupyteradmin
Group=jupyteradmin
WorkingDirectory=/opt/jupyterhub
ExecStart=/opt/jupyterhub/bin/jupyterhub -f
/opt/jupyterhub/etc/jupyterhub_config.py
Restart=always
RestartSec=10

[Install]
```

```
WantedBy=multi-user.target
```

คำอธิบายแต่ละส่วน:

- [Unit]: ส่วนนี้ใช้อธิบายและกำหนดเงื่อนไขของบริการ
  - Description: คำอธิบายสั้นๆ ของบริการ
  - After=network.target: ระบุว่าบริการนี้ควรเริ่มทำงานหลังจากที่ระบบเครือข่ายพร้อมใช้งานแล้ว
- [Service]: ส่วนนี้กำหนดวิธีการทำงานของบริการ
  - Type=simple: กำหนดประเภทของบริการ
  - User=jupyteradmin, Group=jupyteradmin: กำหนดให้บริการนี้รันด้วยสิทธิ์ของผู้ใช้และกลุ่ม jupyteradmin เพื่อให้มีสิทธิ์เข้าถึงไฟล์ที่เกี่ยวข้อง
  - WorkingDirectory: กำหนดไดเรกทอรีการทำงาน
  - ExecStart: คำสั่งที่ใช้ในการเริ่มต้นบริการ
  - Restart=always: กำหนดให้รีสตาร์ทบริการอัตโนมัติเสมอหากบริการหยุดทำงาน
  - RestartSec=10: กำหนดระยะเวลา 10 วินาทีก่อนที่จะพยายามรีสตาร์ทอีกครั้ง
- [Install]: ส่วนนี้กำหนดว่าเมื่อใดควรเปิดใช้งานบริการ
  - WantedBy=multi-user.target: กำหนดให้เปิดใช้งานบริการนี้เมื่อเข้าสู่โหมดผู้ใช้หลายคน (Multi-user)

เปิดใช้งานและเริ่มต้น Service

```
sudo systemctl daemon-reexec
sudo systemctl daemon-reload
sudo systemctl enable jupyterhub
sudo systemctl start jupyterhub
```

- daemon-reexec: สั่งให้ systemd อ่านไฟล์ตั้งค่าใหม่ทั้งหมด
- daemon-reload: สั่งให้ systemd โหลดไฟล์ตั้งค่าของบริการใหม่
- enable jupyterhub: คำสั่งนี้จะสร้าง symlink เพื่อให้ JupyterHub เริ่มทำงานอัตโนมัติทุกครั้งที่ระบบบูท
- start jupyterhub: คำสั่งนี้จะเริ่มต้นบริการ JupyterHub ทันที

คุณสามารถตรวจสอบสถานะของบริการได้ด้วยคำสั่ง sudo systemctl status jupyterhub

4.3 การตั้งค่า Shared Directory สำหรับแชร์ข้อมูลระหว่างผู้ใช้งาน

เพื่อให้นักวิจัยและผู้ใช้ JupyterHub สามารถทำงานร่วมกันได้อย่างมีประสิทธิภาพ การสร้างไดเรกทอรีสำหรับแชร์ข้อมูลจึงเป็นสิ่งสำคัญ บทความนี้จะอธิบายขั้นตอนการตั้งค่า Shared Directory เพื่อให้ผู้ใช้ทุกคนสามารถเข้าถึงและทำงานกับข้อมูลเดียวกันได้ รวมถึงการแก้ไขไฟล์ตั้งค่าเพื่อป้องกันปัญหาด้านสิทธิ์การเข้าถึง

วิธีการตั้งค่า Shared Directory สำหรับแชร์ข้อมูลระหว่างผู้ใช้ Shared Directory จะเป็นพื้นที่ส่วนกลางที่ผู้ใช้ทุกคนสามารถอ่านและเขียนไฟล์ร่วมกันได้ โดยเราจะใช้สิทธิ์การเข้าถึงแบบกลุ่ม (Group Permissions) เพื่อจัดการการเข้าถึงนี้

4.3.1 สร้างไดเรกทอรีสำหรับแชร์ข้อมูล

```
sudo mkdir -p /srv/shared
```

- sudo mkdir -p: คำสั่งนี้ใช้สร้างไดเรกทอรีชื่อ /srv/shared ด้วยสิทธิ์ผู้ดูแลระบบ (sudo) และใช้ตัวเลือก -p เพื่อให้สร้างไดเรกทอรีระดับบน (parent directory) หากยังไม่มีอยู่

4.3.2 สร้างกลุ่มผู้ใช้และเพิ่มสมาชิก

สร้างกลุ่มผู้ใช้ใหม่ชื่อ jupyterhub-users เพื่อรวมผู้ใช้ JupyterHub ทั้งหมดไว้ด้วยกัน จากนั้นจึงกำหนดสิทธิ์การเข้าถึงของไดเรกทอรีให้เฉพาะกลุ่มนี้

```
sudo groupadd jupyterhub-users
```

- sudo groupadd: คำสั่งสำหรับสร้างกลุ่มผู้ใช้ใหม่ด้วยสิทธิ์ผู้ดูแลระบบ

4.3.3 กำหนดเจ้าของและสิทธิ์การเข้าถึงของ Shared Directory

เปลี่ยนเจ้าของกลุ่มของไดเรกทอรี /srv/shared และตั้งค่าสิทธิ์การเข้าถึงที่เหมาะสม

```
sudo chown root:jupyterhub-users /srv/shared
```

- sudo chown root:jupyterhub-users: เปลี่ยนเจ้าของไฟล์/ไดเรกทอรี โดยให้ผู้ใช้ root เป็นเจ้าของ (owner) และกลุ่ม jupyterhub-users เป็นเจ้าของกลุ่ม (group owner)

```
sudo chmod 2775 /srv/shared
```

- sudo chmod 2775: กำหนดสิทธิ์การเข้าถึงของไดเรกทอรี /srv/shared
- 2: เป็น Sticky Bit หรือ SetGID ซึ่งหมายความว่าไฟล์หรือไดเรกทอรีที่สร้างขึ้นภายใน /srv/shared จะถูกกำหนดให้มีเจ้าของกลุ่มเป็น jupyterhub-users โดยอัตโนมัติ
- 775: สิทธิ์การเข้าถึงแบบ Unix
    - 7: เจ้าของไฟล์ (root) สามารถอ่าน, เขียน, และรันได้
    - 7: เจ้าของกลุ่ม (jupyterhub-users) สามารถอ่าน, เขียน, และรันได้
    - 5: ผู้ใช้อื่นๆ สามารถอ่านและรันได้ แต่ไม่สามารถเขียนได้

4.3.4 สร้าง Symbolic Link ใน Template Directory

สร้างลิงก์สำหรับ Shared Directory ไว้ใน /etc/skel ซึ่งเป็นไดเรกทอรี template สำหรับผู้ใช้ใหม่ที่ถูกสร้างขึ้นในระบบ เพื่อให้ผู้ใช้ใหม่เห็น Shared Directory นี้โดยอัตโนมัติ

```
sudo ln -s /srv/shared /etc/skel/shared
```

หมายเหตุ: หากผู้ใช้ของคุณถูกสร้างขึ้นแล้วก่อนหน้านี้ คุณจะต้องสร้าง Symbolic Link ด้วยตนเองในไดเรกทอรี Home ของผู้ใช้นั้นๆ ด้วยคำสั่ง ln -s /srv/shared /home/jupyteradmin/shared เป็นต้น

4.3.5 แก้ไขสิทธิ์สำหรับ .ipynb_checkpoints

Jupyter Lab, Notebook อาจสร้างไดเรกทอรี .ipynb_checkpoints ด้วยสิทธิ์ที่ไม่ถูกต้อง ทำให้ผู้ใช้คนอื่นไม่สามารถเข้าถึงได้ เราจึงต้องแก้ไขสิทธิ์ของไดเรกทอรีนี้ด้วยตนเอง

```
sudo chmod 775 /srv/shared/.ipynb_checkpoints/
```

หลังจากนี้ ผู้ใช้ในกลุ่ม jupyterhub-users ทุกคนจะสามารถเข้าถึงไดเรกทอรี /srv/shared ได้อย่างสมบูรณ์ และสามารถสร้าง, แก้ไข, และลบไฟล์ร่วมกันได้โดยไม่มีปัญหาเรื่องสิทธิ์การเข้าถึงอีกต่อไป

4.4 การเพิ่มชื่อผู้ใช้งานใหม่

4.4.1 เพิ่มผู้ใช้งานใหม่เข้าสู่ระบบ

ขั้นตอนแรกคือการสร้างบัญชีผู้ใช้งานใหม่บน Ubuntu Server โดยใช้คำสั่ง adduser

```
sudo adduser newuser
```

- sudo adduser: คำสั่งสำหรับเพิ่มผู้ใช้งานใหม่เข้าสู่ระบบด้วยสิทธิ์ผู้ดูแลระบบ
- newuser: ชื่อผู้ใช้งานที่คุณต้องการสร้าง (ในตัวอย่างนี้คือ **newuser**)

เมื่อรันคำสั่งนี้ ระบบจะให้คุณตั้งรหัสผ่านสำหรับผู้ใช้งานใหม่และป้อนข้อมูลพื้นฐานอื่นๆ เช่น ชื่อเต็ม, หมายเลขห้องทำงาน เป็นต้น

4.4.2 เพิ่มผู้ใช้งานใหม่เข้าในกลุ่ม JupyterHub

เพื่อให้ผู้ใช้งานใหม่สามารถเข้าถึง Shared Directory และใช้งาน JupyterHub ได้ คุณจะต้องเพิ่มผู้ใช้งานใหม่เข้าในกลุ่ม jupyterhub-users ที่เราได้ตั้งค่าไว้ก่อนหน้านี้

```
sudo usermod -a -G jupyterhub-users newuser
```

- sudo usermod: คำสั่งสำหรับแก้ไขข้อมูลผู้ใช้งาน
- -a: ตัวเลือกนี้หมายถึงการเพิ่ม (-a) ผู้ใช้เข้าไปในกลุ่ม (-G) โดยไม่ลบกลุ่มเดิมที่ผู้ใช้นั้นเป็นสมาชิกอยู่
- -G jupyterhub-users: ระบุชื่อกลุ่มที่ต้องการเพิ่มผู้ใช้เข้าไป
- newuser: ชื่อผู้ใช้งานที่คุณต้องการเพิ่ม

คุณสามารถตรวจสอบรายชื่อสมาชิกในกลุ่ม jupyterhub-users เพื่อยืนยันว่าผู้ใช้ใหม่ถูกเพิ่มเข้าไปอย่างถูกต้องหรือไม่

```
getent group jupyterhub-users
```

4.4.3 แก้ไขไฟล์ตั้งค่า JupyterHub

ถึงแม้ว่าผู้ใช้ใหม่จะอยู่ในกลุ่มที่ถูกต้องแล้ว แต่คุณยังต้องเพิ่มชื่อผู้ใช้ใหม่เข้าไปในรายการที่ได้รับอนุญาตให้เข้าถึง JupyterHub ในไฟล์ตั้งค่า jupyterhub_config.py ด้วย

```
nano /opt/jupyterhub/etc/jupyterhub_config.py
```

เลื่อนหาบรรทัดที่กำหนดค่า c.Authenticator.allowed_users แล้วเพิ่มชื่อผู้ใช้ newuser เข้าไปในรายการ

```
c.Authenticator.allowed_users = {'user1', 'user2',
'newuser'}
```

- c.Authenticator.allowed_users: กำหนดรายชื่อผู้ใช้งานที่ได้รับอนุญาตให้เข้าสู่ระบบ JupyterHub ได้
- 'newuser': เพิ่มชื่อผู้ใช้งานใหม่เข้าไปในกลุ่ม allowed_users

4.4.4 รีสตาร์ท JupyterHub

เมื่อทำการแก้ไขไฟล์ตั้งค่าเรียบร้อยแล้ว ให้รีสตาร์ทบริการ JupyterHub เพื่อให้การตั้งค่าใหม่มีผลทันที

```
sudo systemctl restart jupyterhub
```

หลังจากนี้ ผู้ใช้งาน newuser จะสามารถเข้าสู่ระบบ JupyterHub ด้วยรหัสผ่านที่ตั้งไว้ในขั้นตอนที่ 1 และสามารถเข้าถึง Shared Directory เพื่อทำงานร่วมกับผู้ใช้คนอื่นๆ ได้อย่างสมบูรณ์

4.5 การติดตั้งโปรแกรมจัดการแพ็คเกจสำหรับภาษา Python (PIP)

PIP เป็นเครื่องมือมาตรฐานสำหรับจัดการแพ็กเกจ Python ซึ่งช่วยให้ผู้ใช้สามารถติดตั้ง, อัปเกรด, และลบแพ็กเกจต่างๆ ได้อย่างง่ายดาย การใช้งาน PIP ใน Virtual Environment ของ JupyterHub จะทำให้การจัดการแพ็กเกจมีความเป็นระเบียบและไม่ส่งผลกระทบต่อระบบหลัก

4.5.1 เข้าสู่ระบบ JupyterHub Virtual Environment

ก่อนที่จะเริ่มติดตั้งแพ็กเกจ ต้องเปิดใช้งาน Virtual Environment ของ JupyterHub เสียก่อน เพื่อให้แน่ใจว่าแพ็กเกจจะถูกติดตั้งในสภาพแวดล้อมที่ถูกต้อง

```
source /opt/jupyterhub/bin/activate
```

ติดตั้งแพ็กเกจที่จำเป็น

หลังจากเข้าสู่ Virtual Environment แล้ว คุณสามารถใช้คำสั่ง pip install เพื่อติดตั้งแพ็กเกจที่ต้องการได้

```
pip install pandas scipy matplotlib seaborn pyepics pyvisa
plotly ipympl
```

4.5.2 คุณสมบัติของแพ็กเกจที่จำเป็น

แพ็กเกจเหล่านี้ได้รับการคัดเลือกมาเพื่อให้ครอบคลุมการทำงานด้านวิทยาศาสตร์ข้อมูลและการควบคุมอุปกรณ์ที่หลากหลาย ซึ่งมีบทบาทดังนี้

แพ็กเกจสำหรับจัดการและวิเคราะห์ข้อมูล

- **pandas**: เป็นไลบรารีที่ทรงพลังสำหรับการจัดการและวิเคราะห์ข้อมูลที่มีโครงสร้างเป็นตาราง โดยสามารถใช้ในการอ่าน, จัดการ, และทำความสะอาดข้อมูลการวัดค่าจากเครื่องซินโครตรอนได้อย่างมีประสิทธิภาพ
- **scipy**: เป็นไลบรารีที่รวบรวมเครื่องมือทางวิทยาศาสตร์และคณิตศาสตร์ขั้นสูง ซึ่งเหมาะสำหรับการวิเคราะห์ข้อมูลเชิงลึกทางฟิสิกส์และสถิติจากเครื่องซินโครตรอน เช่น การวิเคราะห์สัญญาณหรือการหาค่าเหมาะสมที่สุดในการปรับค่าอุปกรณ์

แพ็กเกจสำหรับสร้างกราฟและแสดงผล

- **matplotlib**: เป็นไลบรารีพื้นฐานที่ใช้ในการสร้างกราฟและภาพทางวิทยาศาสตร์ที่หลากหลายและปรับแต่งได้สูง เพื่อแสดงผลข้อมูลการวัดค่าของเครื่องซินโครตรอนแบบ Real-time
- **seaborn**: เป็นไลบรารีที่สร้างกราฟทางสถิติที่สวยงามและซับซ้อนมากขึ้น เพื่อทำความเข้าใจความสัมพันธ์ระหว่างตัวแปรต่างๆ ของเครื่องได้อย่างชัดเจน
- **plotly**: เป็นไลบรารีสำหรับสร้างกราฟแบบ Interactive ที่ผู้ใช้สามารถโต้ตอบได้ เช่น การซูมหรือการแสดงข้อมูลเพิ่มเติม ซึ่งเหมาะสำหรับการสร้าง Dashboard และการนำเสนอข้อมูลแบบมีชีวิตชีวา
- **ipympl**: เป็น Jupyter Widget ที่ช่วยให้กราฟของ Matplotlib สามารถโต้ตอบกับผู้ใช้ใน Jupyter Notebooks ได้โดยตรง ทำให้ผู้ใช้สามารถซูม, เลื่อน, และบันทึกกราฟได้ง่าย

แพ็กเกจสำหรับเชื่อมต่อและควบคุมอุปกรณ์

- **pyepics**: เป็นไลบรารีที่ใช้เชื่อมต่อกับระบบควบคุม **EPICS** ซึ่งเป็นมาตรฐานของเครื่องมือวิทยาศาสตร์ขนาดใหญ่ ทำให้สามารถอ่านค่าตัวแปรและส่งคำสั่งควบคุมไปยังอุปกรณ์ของเครื่องซินโครตรอนได้โดยตรง
- **pyvisa**: เป็นไลบรารีที่ช่วยในการควบคุมเครื่องมือทางวิทยาศาสตร์ต่างๆ เช่น Oscilloscopes หรือ Multimeters ผ่านพอร์ต Serial (GPIB, USB) ซึ่งใช้สำหรับเชื่อมต่อกับอุปกรณ์ที่ไม่ได้ใช้ระบบ EPICS

หลังจากติดตั้งแพ็กเกจเหล่านี้แล้ว ผู้ใช้งานใน JupyterHub ก็จะพร้อมสำหรับการทำงานด้านการควบคุม, การวิเคราะห์ข้อมูล, และการแสดงผลได้อย่างสมบูรณ์ โดยไม่จำเป็นต้องติดตั้งซอฟต์แวร์เพิ่มเติมบนเครื่องของตนเอง

4.6 การตั้งค่า Reverse Proxy ด้วย Nginx

การใช้ Reverse Proxy จะช่วยให้เราเข้าถึง JupyterHub ได้ง่ายขึ้นผ่าน URL ที่กำหนดเอง โดยไม่ต้องระบุพอร์ต 8000 โดยมีการตั้งค่าดังต่อไปนี้

สร้างไฟล์ตั้งค่าใหม่สำหรับ JupyterHub โดยเฉพาะในไดเรกทอรี /etc/nginx/sites-available/

```
sudo nano /etc/nginx/sites-available/jupyterhub
```

คัดลอกและวางเนื้อหาต่อไปนี้ลงในไฟล์ (แก้ไข your_domain_or_ip ให้เป็นของเครื่อง JupyterHub):

```
server {
    listen 80;
    server_name your_domain_or_ip;

    location /jupyter/ {
        proxy_pass http://localhost:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

- listen 80: กำหนดให้ Nginx รับฟังการเชื่อมต่อบนพอร์ต 80 (พอร์ต HTTP มาตรฐาน)
- server_name: ชื่อโดเมนหรือ IP Address ของเซิร์ฟเวอร์
- location /jupyter/: กำหนดว่าหากมีคำขอเข้ามาที่ URL ที่มี /jupyter/ จะถูกส่งต่อไปยัง JupyterHub
- proxy_pass http://localhost:8000: ส่งต่อคำขอไปยัง JupyterHub ที่รันอยู่บนพอร์ต 8000
- ส่วน proxy_set_header จะเป็นการส่งข้อมูล Header ที่จำเป็นเพื่อให้ JupyterHub รับรู้ข้อมูลของผู้ใช้งานได้อย่างถูกต้อง

เปิดใช้งานไฟล์ตั้งค่าและรีสตาร์ท Nginx

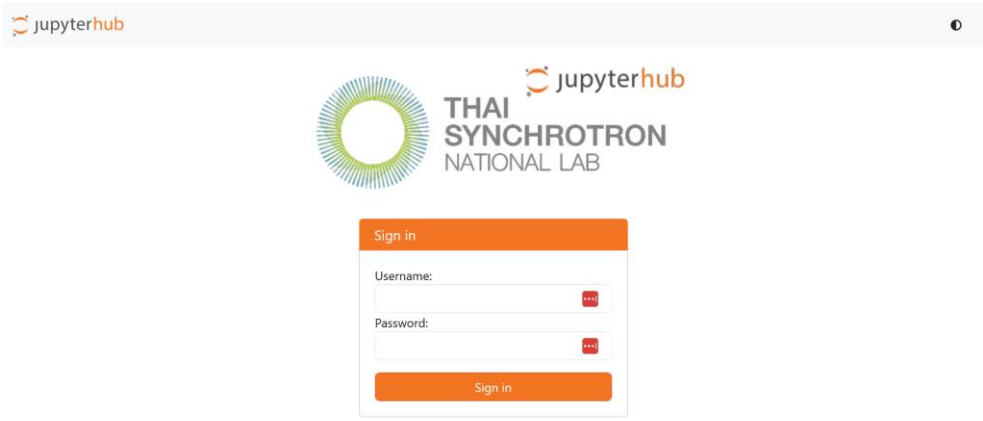
```
sudo ln -s /etc/nginx/sites-available/jupyterhub
/etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl restart nginx
```

- ln -s: สร้าง Symlink (Symbolic Link) เพื่อเปิดใช้งานไฟล์ตั้งค่าในไดเรกทอรี sites-enabled
- nginx -t: ตรวจสอบ Syntax ของ Nginx
- systemctl restart nginx: รีสตาร์ท Nginx เพื่อให้การตั้งค่าใหม่มีผล

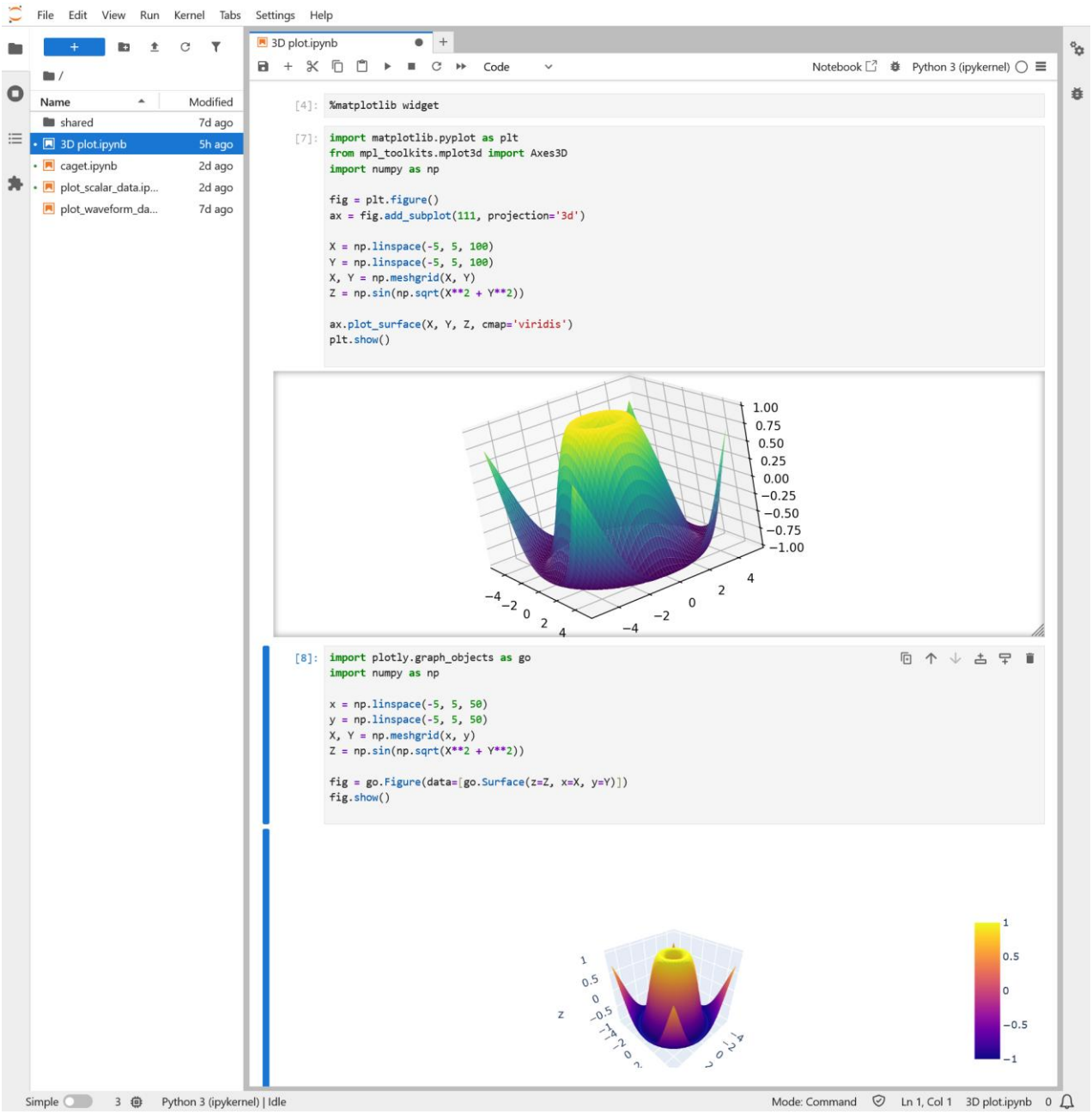
5. ผลลัพธ์ และอภิปรายผล

เมื่อติดตั้งและตั้งค่า JupyterHub สำเร็จ ผู้ใช้จะสามารถเข้าสู่ระบบผ่านเว็บเบราว์เซอร์ด้วย Username และ Password ของตนเอง ตามภาพที่ 2 โดยระบบจะทำการสร้าง Jupyter Notebook/JupyterLab Environment ที่เป็นส่วนตัวให้โดยอัตโนมัติ ตามภาพที่ 3 ซึ่งสภาพแวดล้อมนี้จะมาพร้อมกับไลบรารีและเครื่องมือที่จำเป็นสำหรับการเชื่อมต่อกับระบบควบคุมเครื่องกำเนิดแสงซินโครตรอน เช่น:

- **ไลบรารีการควบคุม:** สำหรับส่งคำสั่งไปยัง EPICS IOC และอุปกรณ์ต่างๆ ของเครื่อง
- **ไลบรารีการวิเคราะห์ข้อมูล:** เช่น NumPy, SciPy, Pandas
- **ไลบรารีการแสดงผล:** เช่น Matplotlib, Plotly
- **การเชื่อมต่อฐานข้อมูล:** เพื่อดึงข้อมูลการทำงานของเครื่องมาวิเคราะห์

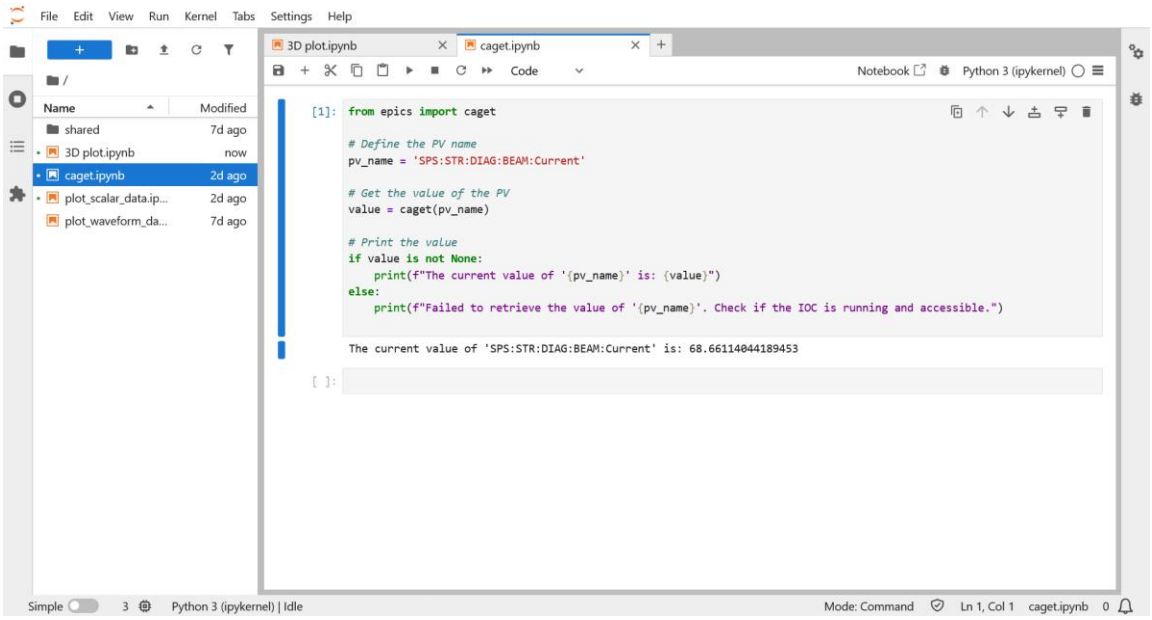


ภาพที่ 2 แสดงหน้า Login ของซอฟต์แวร์ JupyterHub ที่ถูกติดตั้ง

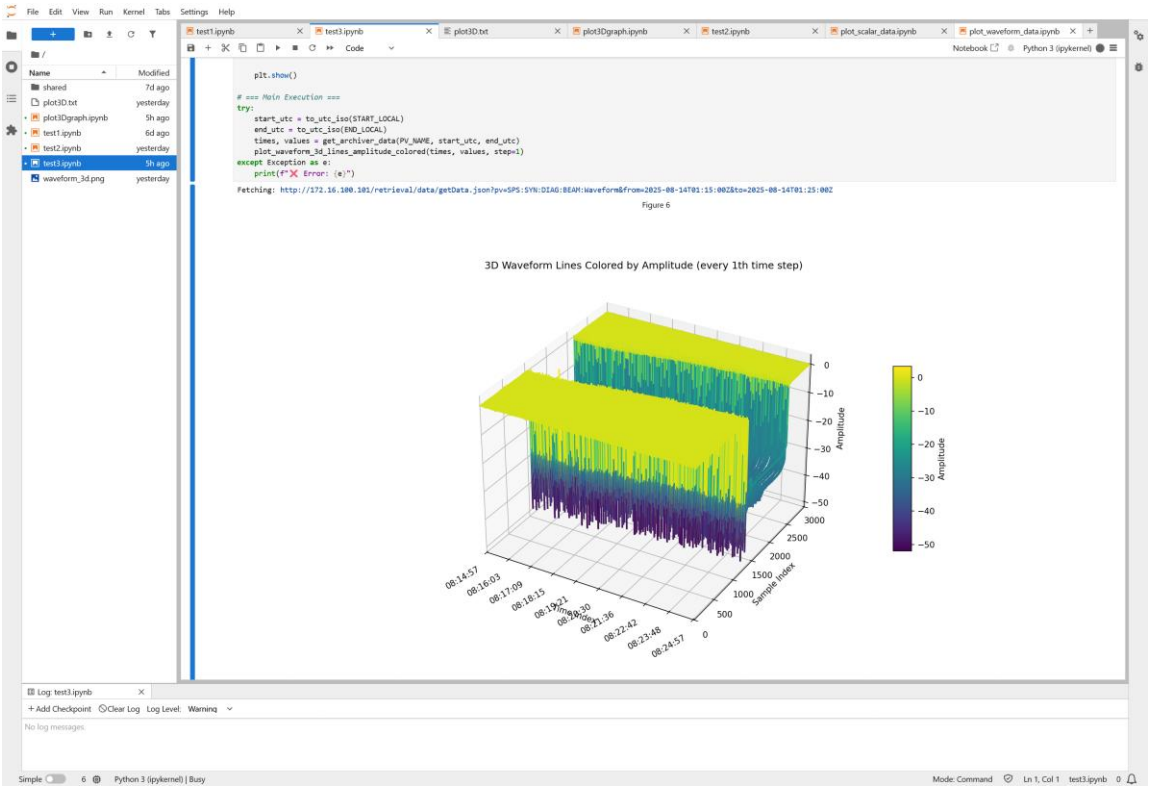


ภาพที่ 3 แสดงหน้า JupyterLab Environment หลังจากผ่านการ Login มาแล้ว

ผู้ใช้สามารถเขียนโค้ด Python เพื่อควบคุมเครื่อง, ดึงข้อมูลแบบ Real-time, และแสดงผลในรูปแบบที่เข้าใจง่ายผ่าน Jupyter Lab, Notebook ได้ทันที ดังแสดงในภาพที่ 4 และ 5



ภาพที่ 4 แสดงการดึงค่าสถานะของ Beam Current (Real-time) ของเครื่องกำเนิดแสงซินโครตรอน



ภาพที่ 5 แสดงการวิเคราะห์ข้อมูลและแสดงผลแบบ 3 มิติ โดยทำการดึงข้อมูลจาก EPICS Archiver

วิธีการใช้งานเบื้องต้น

1. เข้าสู่ระบบ: เปิดเว็บเบราว์เซอร์และไปที่ URL ของ JupyterHub (https://accelerator.slri.or.th/jupyter) จากนั้นใส่ Username และ Password เพื่อเข้าสู่ระบบ
2. เปิดใช้งาน Notebook: เมื่อเข้าสู่ระบบแล้ว ผู้ใช้จะเห็นหน้าจอหลักของ JupyterLab/Jupyter Notebook สามารถสร้างไฟล์ใหม่ (New Notebook), เปิดไฟล์เดิม, หรืออัปโหลดไฟล์จากคอมพิวเตอร์ของตนเองได้
3. เขียนโค้ด: ใน Notebook คุณสามารถเขียนโค้ด Python เพื่อเชื่อมต่อกับระบบควบคุมของเครื่องซินโครตรอนผ่านไลบรารีที่ได้ติดตั้งไว้ล่วงหน้า

6. สรุปผล

การติดตั้งและตั้งค่า JupyterHub บน Ubuntu Server 24.04 เพื่อใช้งานร่วมกับระบบควบคุมเครื่องกำเนิดแสงซินโครตรอน ได้บรรลุวัตถุประสงค์หลักในการสร้างแพลตฟอร์มการทำงานแบบรวมศูนย์ที่มีประสิทธิภาพและปลอดภัย ซึ่งช่วยส่งเสริมการทำงานร่วมกันของบุคลากรในทีมวิจัยและผู้ใช้งานได้อย่างมีนัยสำคัญ

การดำเนินงานได้ผ่านกระบวนการและบรรลุผลลัพธ์สำคัญดังนี้

6.1 ผลลัพธ์ด้านการสร้างแพลตฟอร์ม

1. JupyterHub ถูกติดตั้งสำเร็จใน Virtual Environment เพื่อแยกแพ็กเกจและ dependencies ออกจากระบบหลัก ทำให้การจัดการมีความยืดหยุ่นและลดความเสี่ยงที่อาจเกิดขึ้นกับระบบปฏิบัติการ
2. ระบบยืนยันตัวตน (Authentication) ถูกตั้งค่าให้ใช้ PAM Authenticator ซึ่งทำให้ผู้ใช้สามารถเข้าสู่ระบบด้วยบัญชีผู้ใช้ของระบบปฏิบัติการได้โดยตรง จึงไม่ต้องสร้างระบบจัดการผู้ใช้ใหม่
3. Nginx ได้รับการตั้งค่าให้เป็น Reverse Proxy เพื่อทำหน้าที่เป็นตัวกลางในการส่งต่อคำขอจากพอร์ต 80 ไปยังพอร์ต 8000 ของ JupyterHub ทำให้ผู้ใช้สามารถเข้าถึงแพลตฟอร์มผ่าน URL มาตรฐานที่เข้าใจง่าย
4. systemd ถูกใช้เป็นเครื่องมือในการจัดการบริการ ทำให้ JupyterHub สามารถทำงานโดยอัตโนมัติเมื่อระบบบูท และสามารถรีสตาร์ทตัวเองได้เมื่อเกิดข้อผิดพลาด ซึ่งรับประกันความต่อเนื่องในการให้บริการ

6.2 ผลลัพธ์ด้านการอำนวยความสะดวกในการใช้งาน

1. Shared Directory ถูกสร้างขึ้นและกำหนดสิทธิ์การเข้าถึงแบบกลุ่ม (jupyterhub-users) อย่างถูกต้อง ทำให้ผู้ใช้งานทุกคนสามารถแชร์ข้อมูล, โค้ด, และผลการทดลองร่วมกันได้อย่างราบรื่น

2. สภาพแวดล้อมการทำงาน ของผู้ใช้แต่ละคนถูกกำหนดค่าให้เริ่มต้นด้วย JupyterLab ซึ่งเป็นอินเทอร์เฟซที่ทันสมัยและมีประสิทธิภาพสูง
3. ชุดไลบรารี Python ที่จำเป็นสำหรับการวิเคราะห์ข้อมูลและการควบคุมระบบได้รับการติดตั้งครบถ้วน ได้แก่:
    - pyepics และ pyvisa: สำหรับการเชื่อมต่อและควบคุมอุปกรณ์ของเครื่องซินโครตรอนได้โดยตรง
    - pandas, scipy, matplotlib, seaborn, plotly, และ ipympl: สำหรับการจัดการ, วิเคราะห์, และสร้างภาพข้อมูลเชิงลึก
4. มีการกำหนดค่า Environment Variable (PATH) ให้กับเซิร์ฟเวอร์ของผู้ใช้แต่ละคน เพื่อให้สามารถเรียกใช้คำสั่งจากระบบควบคุม EPICS ได้ทันทีโดยไม่ต้องระบุเส้นทางเต็ม

6.3 ผลลัพธ์ด้านประโยชน์ที่ได้รับจากกลุ่มผู้ใช้งาน

การนำ JupyterHub มาใช้ได้สร้างประโยชน์อย่างชัดเจนให้กับกลุ่มผู้ใช้งานต่างๆ ดังนี้:

1. นักวิทยาศาสตร์และนักวิจัย: สามารถทำการวิเคราะห์ข้อมูลเชิงลึกและสร้างโมเดลทางฟิสิกส์ได้อย่างรวดเร็วและแม่นยำ โดยไม่ต้องกังวลเรื่องการติดตั้งซอฟต์แวร์ที่ยุ่งยาก
2. วิศวกรและผู้ควบคุมระบบ: ได้รับเครื่องมือที่มีประสิทธิภาพในการตรวจสอบสถานะของเครื่องแบบ Real-time, ทดสอบคำสั่งควบคุมใหม่ๆ, และสามารถแก้ไขปัญหาได้รวดเร็วยิ่งขึ้น
3. นักเรียนและนักศึกษา: สามารถเข้าถึงสภาพแวดล้อมการทดลองจริงได้ในรูปแบบที่ปลอดภัยและเรียนรู้ได้ง่าย ทำให้การฝึกอบรมมีประสิทธิภาพสูงขึ้น
4. บุคลากรด้านเทคนิคและซอฟต์แวร์: สามารถใช้แพลตฟอร์มนี้ในการพัฒนาและทดสอบซอฟต์แวร์ควบคุมใหม่ๆ รวมถึงใช้เป็นเครื่องมือในการสร้างเอกสารประกอบการใช้งานในรูปแบบ Interactive

6.4 ข้อเสนอแนะเพื่อการพัฒนาในอนาคต

1. เพิ่มการยืนยันตัวตนแบบสองปัจจัย (2FA) เพื่อเพิ่มความปลอดภัย
2. เชื่อมต่อ LDAP หรือ Active Directory สำหรับการบริหารจัดการผู้ใช้งานในองค์กรขนาดใหญ่
3. สำรองข้อมูลและทำ Snapshot ของระบบอย่างสม่ำเสมอ เพื่อป้องกันการสูญหายของข้อมูล
4. พิจารณาใช้ Docker หรือ Kubernetes เพื่อเพิ่มความสามารถในการขยายระบบ (Scalability)

6.5 สรุปผลโดยรวม

โดยสรุปแล้ว การติดตั้ง JupyterHub ในครั้งนี้ได้ก่อให้เกิดแพลตฟอร์มที่ครอบคลุมและมีประสิทธิภาพสูง ซึ่งเป็นศูนย์กลางในการขับเคลื่อนงานวิจัยและพัฒนาของเครื่องกำเนิดแสงซินโครตรอนได้อย่างแท้จริง ระบบที่สร้างขึ้นมีความมั่นคง ปลอดภัย และสามารถรองรับการใช้งานจากผู้ใช้หลายคนได้อย่างไร้รอยต่อ ซึ่งเป็นก้าวสำคัญในการนำเทคโนโลยีวิทยาศาสตร์ข้อมูลมาประยุกต์ใช้กับงานวิจัยทางวิทยาศาสตร์ขั้นสูง

7. กลุ่มผู้ใช้ประโยชน์

การนำ JupyterHub มาใช้กับระบบควบคุมเครื่องกำเนิดแสงซินโครตรอน จะช่วยให้กลุ่มผู้ใช้หลากหลายกลุ่มสามารถเข้าถึงและใช้ประโยชน์จากเครื่องมือนี้ได้อย่างเต็มที่ โดยสามารถแบ่งกลุ่มผู้ใช้หลักๆ ได้ดังนี้

1. **นักวิทยาศาสตร์และนักวิจัย:** กลุ่มนี้คือผู้ใช้งานหลักที่ต้องการวิเคราะห์ข้อมูลเชิงลึกและสร้างโมเดลทางฟิสิกส์ นักวิจัยสามารถใช้ JupyterHub เพื่อเขียนโค้ด Python ในการดึงข้อมูลการทดลองแบบ Real-time มาวิเคราะห์, สร้างกราฟเพื่อแสดงผลความสัมพันธ์ของข้อมูล, และพัฒนาอัลกอริทึมใหม่ๆ เพื่อควบคุมการทดลองได้อย่างแม่นยำยิ่งขึ้น นอกจากนี้ยังช่วยให้สามารถทำงานร่วมกันในทีมได้ง่ายขึ้นด้วยการแชร์ Notebooks และผลการวิเคราะห์
2. **วิศวกรและผู้ควบคุมระบบ:** กลุ่มนี้มีหน้าที่ดูแลการทำงานของเครื่องให้เป็นไปอย่างราบรื่น วิศวกรสามารถใช้ JupyterHub เพื่อเขียนสคริปต์ในการตรวจสอบสถานะของอุปกรณ์ต่างๆ, ดูค่าพารามิเตอร์ที่สำคัญแบบ Real-time, และสร้างระบบแจ้งเตือนอัตโนมัติเมื่อเกิดความผิดปกติ นอกจากนี้ยังสามารถใช้ Notebooks เพื่อทดสอบการทำงานของคำสั่งควบคุมใหม่ๆ ก่อนที่จะนำไปใช้จริงบนระบบหลัก ซึ่งช่วยลดความเสี่ยงในการเกิดข้อผิดพลาดได้
3. **นักศึกษาและนักฝึกอบรม:** กลุ่มนี้ใช้ JupyterHub เป็นเครื่องมือการเรียนรู้ที่ทรงพลัง นักศึกษาสามารถเข้าถึงสภาพแวดล้อมการทำงานของเครื่องซินโครตรอนได้โดยตรงภายใต้การควบคุมดูแลของอาจารย์และผู้เชี่ยวชาญ ทำให้สามารถฝึกฝนการเขียนโค้ดเพื่อควบคุมอุปกรณ์จริง, การประมวลผลข้อมูลขนาดใหญ่, และทำความเข้าใจหลักการทำงานของเครื่องมือทางวิทยาศาสตร์ขั้นสูงได้ง่ายขึ้น โดยไม่จำเป็นต้องติดตั้งซอฟต์แวร์ที่ซับซ้อนด้วยตนเอง
4. **บุคลากรด้านเทคนิคและซอฟต์แวร์:** กลุ่มนี้คือผู้พัฒนาและดูแลระบบควบคุมของเครื่อง บุคลากรสามารถใช้ JupyterHub เพื่อพัฒนาไลบรารีหรือฟังก์ชันการทำงานใหม่ๆ สำหรับการควบคุมเครื่อง, ทดสอบโค้ดที่เขียนขึ้น, และสร้างเอกสารประกอบการใช้งานในรูปแบบของ Jupyter Notebook ซึ่งทำให้การสื่อสารระหว่างทีมง่ายขึ้นและช่วยให้ผู้ใช้คนอื่นๆ สามารถเข้าใจวิธีการใช้งานได้อย่างรวดเร็ว

เอกสารอ้างอิง

[1] The JupyterHub Team. "JupyterHub Official Documentation." [Online]. Available: https://jupyterhub.readthedocs.io/en/stable/

[2] The Littlest JupyterHub. "The Littlest JupyterHub (TLJH)." [Online]. Available: https://tljh.jupyter.org/

[3] Docker Inc. "Docker Official Documentation." [Online]. Available: https://docs.docker.com/

[4] Nginx. "Nginx Official Documentation." [Online]. Available: https://nginx.org/en/docs/

[5] PyEPICS Development Team. "pyepics Official Documentation." [Online]. Available: https://pyepics.github.io/pyepics/

ภาคผนวก

ตัวอย่างการตั้งค่า jupyterhub_config.py

```
c = get_config()

# Bind on all IP addresses, port 8000
c.JupyterHub.bind_url = 'http://:8000'

# Set JupyterHub to run under the /jupyter/ subdirectory
c.JupyterHub.base_url = '/jupyter/'

# Use system authentication
c.JupyterHub.authenticator_class = 'jupyterhub.auth.PAMAuthenticator'

# Use jupyterhub-singleuser from this virtual environment
c.Spawner.cmd = ['/opt/jupyterhub/bin/jupyterhub-singleuser']

# Launch JupyterLab interface
c.Spawner.default_url = '/lab'

# Set environment variables for user servers to ensure correct PATH
c.Spawner.environment = {
    'PATH':
'/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin:/sps/pr
od/epics/base/bin/linux-x86_64;/opt/jupyterhub/bin',
}

# Whitelist and admin settings
c.Authenticator.allowed_users = {'user1','user2','user3',}
c.Authenticator.admin_users = {'admin1', 'admin2'}

c.JupyterHub.template_paths = ['/opt/jupyterhub/templates']

# Enable terminal
c.Spawner.args = ["--NotebookApp.terminals_enabled=True"]
```

ประวัติการเปลี่ยนแปลง

ฉบับแก้ไข	วันที่	แก้ไขโดย	ส่วนที่แก้ไข	คำอธิบาย
R1.0	18 ส.ค. 68	ณัฐวุฒิ สุรเดช	-	-
R1.1	22 ส.ค. 68	ณัฐวุฒิ สุรเดช	4.1.7	เพิ่ม PATH “:/opt/jupyterhub/bin”