

โครงการเลขที่ วศ.คพ. P069-1/2567

เรื่อง

รายงานสหกิจศึกษา

โดย

ณัฐพงษ์ เทพพิทักษ์ รหัส 640610634

โครงการนี้

เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่
ปีการศึกษา 2567

PROJECT No. CPE P069-1/2567

Cooperative Education Report

Natthaphong Thepphithak 640610634

**A Project Submitted in Partial Fulfillment of Requirements
for the Degree of Bachelor of Engineering
Department of Computer Engineering
Faculty of Engineering
Chiang Mai University
2024**

หัวข้อโครงการ : รายงานสหกิจศึกษา
โดย : ณัฐพงษ์ เพพพิทักษ์ รหัส 640610634
ภาควิชา : วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษา : รศ.ดร. ปภิเวช วุฒิสารวัฒนา
ปริญญา : วิศวกรรมศาสตรบัณฑิต
สาขา : วิศวกรรมคอมพิวเตอร์
ปีการศึกษา : 2567

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่ ได้อนุมัติให้โครงการนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต (สาขาวิศวกรรมคอมพิวเตอร์)

..... หัวหน้าภาควิชาวิศวกรรมคอมพิวเตอร์
(รศ.ดร. สันติ พิทักษ์กิจนุภร)

คณะกรรมการสอบโครงการ

..... ประธานกรรมการ
(รศ.ดร. ปภิเวช วุฒิสารวัฒนา)

..... กรรมการ
(ผศ.ดร. ภาสกร แซ่บประเสริฐ)

หัวข้อโครงการ : รายงานสหกิจศึกษา
โดย : ณัฐพงษ์ เทพพิทักษ์ รหัส 640610634
ภาควิชา : วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษา : รศ.ดร. ปฏิเวช วุฒิสารวัฒนา¹
ปริญญา : วิศวกรรมศาสตรบัณฑิต
สาขา : วิศวกรรมคอมพิวเตอร์
ปีการศึกษา : 2567

บทคัดย่อ

รายงานนี้เป็นนำเสนอการสหกิจของวิศวกรรมศาสตรสาขาวิชาคอมพิวเตอร์ในตำแหน่ง DevOps Engineer ที่ SCB TechX ซึ่งเป็นส่วนหนึ่งของธนาคารไทยพาณิชย์ (SCB) ในระหว่างช่วงเวลาของการทำงาน ได้มีส่วนร่วมในการสนับสนุนทีมพัฒนาซอฟต์แวร์ โดยเฉพาะในด้านการสร้างและการลงทุนทรัพยากร รวมถึงการพัฒนาและบำรุงรักษา Terraform Modules สำหรับโมดูลกลางที่ถูกใช้งานทั่วไปใน SCB TechX และ SCB ซึ่งทั้งหมดนี้เป็นส่วนสำคัญในการเพิ่มประสิทธิภาพและความคล่องตัวให้กับกระบวนการ DevOps ของ บริษัท เพื่อตอบสนองความต้องการของลูกค้าอย่างมีประสิทธิภาพและรวดเร็ว

Project Title : Cooperative Education Report
Name : Natthaphong Thepphithak 640610634
Department : Computer Engineering
Project Advisor : Assoc. Prof. Patiwet Wuttisarnwattana, Ph.D.
Degree : Bachelor of Engineering
Program : Computer Engineering
Academic Year : 2024

ABSTRACT

This report presents a cooperative education experience in Computer Engineering for the position of DevOps Engineer at SCB TechX, a subsidiary of Siam Commercial Bank (SCB). During the work period, I participated in supporting the software development team, particularly in the creation and deletion of resources, as well as the development and maintenance of Terraform Modules for central modules used in both SCB TechX and SCB. All of these activities were crucial in enhancing the efficiency and agility of the company's DevOps processes to effectively and rapidly meet customer needs.

กิตติกรรมประกาศ

จากประสบการณ์การปฏิบัติสหกิจศึกษาที่ SCB TechX ซึ่งเป็นส่วนหนึ่งของธนาคารไทยพาณิชย์ (SCB) ผู้ได้รับโอกาสอันดีในการเรียนรู้และได้รับประสบการณ์ที่มีค่ายิ่ง การจัดทำรายงานสหกิจศึกษาฉบับนี้สำเร็จลุล่วงได้ด้วยการสนับสนุนจากหลายฝ่าย ซึ่งผู้ขอขอบคุณเป็นอย่างสูง ได้แก่

1. คณาจารย์ทุกท่านที่ให้คำแนะนำและสนับสนุนในการทำงาน
2. บุคลากรของ SCB TechX ที่ให้คำปรึกษาและความช่วยเหลือตลอดระยะเวลาการปฏิบัติงาน
3. เพื่อนร่วมงานทุกคนที่ให้ความร่วมมือและมิตรภาพอันดี

รวมถึงบุคคลอื่นๆ ที่ไม่ได้อายนามในรายงานฉบับนี้ ซึ่งได้ให้การสนับสนุนและคำแนะนำในการทำงาน นอก จากนี้ ผู้ขอแสดงความขอบคุณอย่างสูงต่ออาจารย์ที่ปรึกษา ที่ได้ให้คำแนะนำและคำปรึกษาในการทำงาน ตลอดระยะเวลาการปฏิบัติสหกิจศึกษา ณ บริษัทแห่งนี้ สุดท้ายนี้ ผู้หวังว่าประสบการณ์และความรู้ที่ได้รับจากการปฏิบัติสหกิจศึกษาครั้งนี้ จะเป็นประโยชน์ต่อการพัฒนาตนเองและการทำงานในอนาคต

ณัฐพงษ์ เทพพิทักษ์
25 ตุลาคม 2567

สารบัญ

บทคัดย่อ	๑
Abstract	๒
กิตติกรรมประกาศ	๓
สารบัญ	๔
สารบัญรูป	๕
1 ข้อมูลทั่วไปของบริษัท	1
1.1 ประวัติความเป็นมาของบริษัท	1
1.2 บริการและผลิตภัณฑ์ของบริษัท	1
1.3 ผู้บริหารของบริษัท	2
2 รายเอียดการไปสหกิจศึกษา	3
2.1 ปรับความรู้พื้นฐานของการเป็น DevOps	3
2.1.1 Jenkins	4
2.1.2 Terraform	5
2.2 งานที่ได้รับมอบหมาย	6
2.3 สวัสดิการที่ได้รับ	6
3 สรุป	7
บรรณานุกรม	8
ก เอกสารสำคัญที่เกี่ยวข้อง	10

สารบัญรูป

1.1 ผู้บริหารและตำแหน่งของบริษัท	2
--	---

บทที่ 1

ข้อมูลทั่วไปของบริษัท

1.1 ประวัติความเป็นมาของบริษัท

SCB TechX [1] ก่อตั้งขึ้นจากความร่วมมือระหว่าง SCBX กลุ่มธุรกิจการเงินและเทคโนโลยีชั้นนำของไทย และ Publicis Sapient บริษัทที่ปรึกษาด้านดิจิทัลทรานส์ฟอร์เมชันระดับโลก มีจุดมุ่งหมายเพื่อมอบบริการด้านเทคโนโลยีที่ตอบสนองความต้องการของธุรกิจต่าง ๆ ตั้งแต่การสร้างนวัตกรรมและผลิตภัณฑ์ใหม่ไปจนถึงการนำเทคโนโลยีมาเพิ่มประสิทธิภาพในการดำเนินงาน

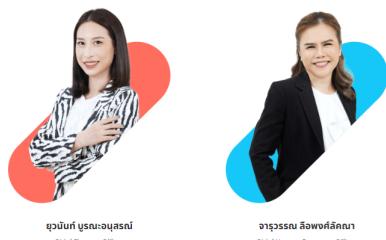
บริษัทมีความเชี่ยวชาญในการพัฒนาโซลูชันในระดับองค์กร (Enterprise-grade solutions) ที่ปล่อยด้วยและรองรับการใช้งานของฐานลูกค้าจำนวนมาก นอกจากนี้ SCB TechX ยังจัดองค์กรในรูปแบบ Startup เพื่อลดความซ้ำซ้อนในการทำงานและส่งเสริมความคิดสร้างสรรค์ใหม่ ๆ ทำให้สามารถพัฒนาโซลูชันให้กับลูกค้าได้อย่างรวดเร็วและมีประสิทธิภาพ

1.2 บริการและผลิตภัณฑ์ของบริษัท

SCB TechX นำเสนอวัตกรรมที่พร้อมใช้งานหลากหลายด้าน [2] ทั้งระบบยืนยันตัวตนแบบดิจิทัลด้วยระบบ KYC [3] ซึ่งทางบริษัทจะเรียกว่า eKYC และแพลตฟอร์มทางการเงินที่หลากหลาย นวัตกรรมเหล่านี้สามารถเชื่อมต่อ กับระบบของลูกค้าได้อย่างรวดเร็วและง่ายดาย พร้อมทั้งปรับแต่งตามความต้องการเฉพาะของธุรกิจ ส่งผลให้ลูกค้าของ SCB TechX สามารถเปิดตัวบริการใหม่หรือยกระดับการให้บริการได้อย่างทันท่วงที

นอกจากนี้ SCB TechX ยังให้บริการที่ครอบคลุมด้านการให้คำปรึกษาทางเทคโนโลยี (Technology Consulting), โซลูชันด้านโครงสร้างพื้นฐานและแพลตฟอร์ม (Infrastructure & Platforms), โซลูชันคลาวด์ (Cloud Solutions), แพลตฟอร์มเทคโนโลยีแบบครบวงจร (xPlatform), การจัดการข้อมูลและความปลอดภัย (Data & Security), และโซลูชันด้านข้อมูลและ AI (TechX Data & AI Solutions) ที่ออกแบบมาเพื่อรองรับและเสริมสร้างศักยภาพให้กับธุรกิจในยุคดิจิทัล

1.3 ផ្នែកប្រឹក្សាអនុបាល



រូបថី 1.1: ផ្នែកប្រឹក្សាអនុបាល

บทที่ 2

รายเอียดการไปสหกิจศึกษา

2.1 ปรับความรู้พื้นฐานของการเป็น DevOps

แนวทางในการเริ่มต้นทำงานในสายงาน DevOps จำเป็นต้องมีการศึกษาและปรับพื้นฐานความรู้ที่สำคัญเพื่อให้แน่ใจว่าพร้อมสำหรับการทำงานจริง เนื่องจาก DevOps เป็นสายงานที่มีความใหม่และมีการพัฒนาอย่างต่อเนื่องในวงการซอฟต์แวร์ โดยหัวข้อที่ได้รับมอบหมายให้ศึกษาเพื่อเตรียมความพร้อมจะประกอบด้วย

- **Docker:** เครื่องมือสำหรับการทำ Containerization เพื่อเพิ่มประสิทธิภาพในการพัฒนาและการส่งมอบซอฟต์แวร์
- **Kubernetes:** ระบบสำหรับการทำ Orchestration และจัดการคอนเทนเนอร์ที่ทำงานในสเกลใหญ่
- **Jenkins:** เครื่องมือสำหรับการทำ Continuous Integration/Continuous Deployment (CI/CD) เพื่อเพิ่มความรวดเร็วและลดข้อผิดพลาดในการพัฒนาซอฟต์แวร์
- **Terraform & IaC:** เครื่องมือสำหรับการทำ Infrastructure as Code (IaC) เพื่อจัดการและปรับแต่งโครงสร้างพื้นฐานด้วยโค้ด
- **Monitoring Tools:** เครื่องมือที่ใช้ในการตรวจสอบและติดตามการทำงานของระบบอย่างมีประสิทธิภาพ
- **ELK Stack:** ระบบสำหรับการจัดการและวิเคราะห์ข้อมูล Logging เพื่อช่วยในการตรวจสอบและวิเคราะห์ปัญหาในระบบ

ทั้งนี้การศึกษาหัวข้อเหล่านี้มีระยะเวลาประมาณ 2-4 สัปดาห์ และท้ายสุดจะต้องมีการนำเสนอสิ่งที่ได้เรียนรู้ ให้กับพี่ ๆ ในทีมได้ฟังและประเมินว่าพร้อมที่จะทำงานจริงหรือไม่ อย่างไรก็ตามรายละเอียดในหัวข้ออยู่ต่าง ๆ หลังจากนี้จะเป็นการนำเสนอสิ่งที่ได้เรียนรู้และได้นำมาประยุกต์ใช้ในการทำงานจริง ส่วนหัวข้อนอกเหนือจากที่จะกล่าวถึงก็สำคัญไม่น้อยเช่นกันแต่จะขอนำเสนอ Documentation ที่ได้ทำสรุปการเรียนรู้มาแล้วนั้นในส่วนภาคผนวก

2.1.1 Jenkins

Jenkins คือ Software (Tool) ตัวนึงที่นำมาใช้ทำ CI/CD [4] เพื่อที่จะสามารถทำให้งานของ Dev & Dev ถูกพัฒนาและส่งมอบให้กับลูกค้าได้เร็วขึ้น โดยที่ Jenkins จะช่วยในการทำงานของการ Build, Test, Deploy ทั้งนี้ในบริบทของการใช้งาน Jenkins ของ DevOps ในงานจริงอาจแตกต่างออกไปดังนั้นในหัวข้อนี้จะเป็นการสรุปว่า DevOps ใช้ Jenkins ทำอะไรบ้าง และ Jenkins ช่วยในการทำงานของ DevOps อย่างไร

Jenkins จะมีหนึ่งความสามารถที่เรียกว่า Jenkins Pipeline ซึ่ง DevOps เองก็ทำความสามารถตรงนี้มาใช้ในการทำงาน ไม่ว่าจะเป็นการ Provisioning & Destroying resources, Deploying โดยการเขียน Script ในรูปแบบของ Jenkinsfile ขึ้นมาเพื่อทำงานตามที่ต้องการ เช่น Pipeline สำหรับ Deploy microservice Pipeline นี้ก็จะทำงานสำหรับการ Deploy โดยเฉพาะ โดยที่เมื่อมีการสั่งให้ทำงาน Jenkins จะทำงานตามที่เขียนไว้ใน Jenkinsfile หลังจาก Jenkins ทำงานเสร็จสิ้นก็เป็นอันว่า microservice นั้น Deploy สำเร็จ

จะเห็นได้ว่าเมื่อเรามี Jenkins เข้ามาช่วยทำงานที่เป็น Routine ที่เราต้องอะไรเดินๆ ทำให้เราสามารถลดเวลาในการทำงานลงได้และยังช่วยให้ Productivity ของทีมทำงานเพิ่มขึ้นอีกด้วย ทั้งจากที่กล่าวมาข้างต้นว่า DevOps ใช้ Jenkins ในการ Provisioning resource ต่างๆ ด้วยนั้นหลักการก็จะคล้ายๆ กับการ Deploy เพียงแต่ Script ที่ใช้สั่งการนั้นจะเปลี่ยนแปลงมหเป็น Script สำหรับ Provisioning resource แทน ทั้งนี้ทำให้เราสามารถทำงานได้เร็วขึ้นและลดความผิดพลาดในการทำงานลงได้

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                echo 'Building...'
            }
        }
        stage('Deploy') {
            steps {
                echo 'Deploying...'
            }
        }
    }
}
```

2.1.2 Terraform

Terraform คือเครื่องมือหนึ่งที่ใช้ในการสร้างและจัดการโครงสร้างพื้นฐาน (Infrastructure) ด้วยแนวคิด *Infrastructure as Code* (IaC) โดยที่ Terraform นั้นมีความสามารถในการ Provision และ Manage resource ต่างๆ ใน cloud provider ได้หลายแห่ง เช่น AWS, Azure, และ GCP รวมถึง on-premises environments อื่นๆ ซึ่งช่วยให้การจัดการโครงสร้างพื้นฐานเป็นไปอย่างมีประสิทธิภาพและสามารถควบคุมได้ง่ายขึ้น

การใช้งาน Terraform ในบริบทของ DevOps

ในบริบทของการใช้งานจริง Terraform มักจะถูกนำมาใช้โดยทีม DevOps เพื่อจัดการและ Provision resource ต่างๆ เช่น เซิร์ฟเวอร์, ฐานข้อมูล, Network configurations และอื่นๆ รวมถึงสามารถใช้ Terraform ในการจัดการ *infrastructure lifecycle* ทั้งหมดไม่ว่าจะเป็นการสร้าง, เปลี่ยนแปลง, หรือการลบ resources ได้อย่างง่ายดายและเป็นอัตโนมัติ

Terraform มีโครงสร้างที่เรียบง่ายในการใช้งาน ซึ่งประกอบด้วยการเขียน Configuration files (มักเป็นไฟล์ที่มีนามสกุล .tf) เพื่อกำหนด resource ที่ต้องการ ซึ่งไฟล์เหล่านี้สามารถจัดเก็บไว้ในระบบ version control (เช่น Git) เพื่อให้สามารถทำการ versioning และการทำงานร่วมกันในทีมได้

นอกจากนี้ Terraform ยังมีความสามารถในการ *plan* และ *preview* การเปลี่ยนแปลงที่จะเกิดขึ้นกับ infrastructure ก่อนที่จะทำการ *apply* จริง ทำให้สามารถลดความเสี่ยงจากการปรับเปลี่ยนโครงสร้างพื้นฐานที่อาจทำให้เกิดปัญหาได้

ข้อสรุปของการใช้งาน Terraform โดย DevOps

- **Provisioning Resources:** ใช้ในการสร้าง resource ต่างๆ ใน cloud หรือ on-premises
- **Infrastructure as Code (IaC):** ทำให้การจัดการ infrastructure มีความคล่องตัวและควบคุมได้ง่ายขึ้น
- **Automation:** ทำให้การจัดการ resource เป็นอัตโนมัติ ลดงาน manual
- **Version Control:** Configuration files สามารถจัดเก็บและ versioning ได้ ทำให้การทำงานร่วมกันในทีมง่ายขึ้น
- **Safety:** สามารถ *plan* และ *preview* การเปลี่ยนแปลงก่อน *apply* จริงเพื่อลดความเสี่ยง

2.2 งานที่ได้รับมอบหมาย

2.3 สวัสดิการที่ได้รับ

บริษัท SCB TechX ให้ความสำคัญกับเรื่องของสวัสดิการที่ดีให้กับพนักงาน โดยเฉพาะนักศึกษาสหกิจศึกษา ที่เข้ามาทำงานในบริษัท ถึงแม้จะไม่ได้เป็นพนักงานประจำ แต่ก็ได้รับสวัสดิการที่ดีจากบริษัทยอย่างเช่น

- Mac Book Pro ให้ใช้งานระหว่างระยะเวลาทำงาน
- อาหารเช้าและน้ำดื่มฟรีทุกวันทำงาน
- Account ของแหล่งเรียนรู้ออนไลน์เช่น Udemy ที่สามารถเรียน Course ใหม่ ๆ ได้ฟรี
- กิจกรรมพัฒนานักศึกษาฝึกงานและสหกิจศึกษาทั้งด้าน Soft Skill และ Hard Skill ตลอดระยะเวลาทำงาน
- เปี้ยนเดือนละ 500 บาท/วันทำงาน

ทั้งนี้ทั้งหมดที่กล่าวมาเป็นเพียงส่วนหนึ่งของสวัสดิการที่ได้รับจากบริษัท SCB TechX และยังมีสวัสดิการอื่น ๆ ที่ยังไม่ได้กล่าวถึง

บทที่ 3

สรุป

បរទេសាន្តកម្ម

- [1] About scb techx. SCB TechX. [Online]. Available: <https://scbtechx.io/th/about-us/>
- [2] Innovative products. SCB TechX. [Online]. Available: <https://scbtechx.io/th/services-products/>
- [3] What is kyc. TMBThanachart Bank. [Online]. Available: <https://www.ttbbank.com/th/corporate/corp-digital-banking-and-other-services/other-service-crop/kyc-cdd>
- [4] Ci/cd. Red Hat. [Online]. Available: <https://www.redhat.com/en/topics/devops/what-is-ci-cd>

ภาคผนวก

ภาคผนวก ก
เอกสารสำคัญที่เกี่ยวข้อง

- เอกสารสำคัญของคณะ
 - วศ.สก.-03/04 รายงานตัวเข้าสหกิจศึกษา (หน้า 11)
- เอกสารสำคัญที่เกี่ยวข้องกับงาน
 - เอกสารสรุปสิ่งที่ได้เรียนรู้และลองทำ Docker (หน้า 18)
 - เอกสารสรุปสิ่งที่ได้เรียนรู้และลองทำ Kubernetes (หน้า 25)
 - เอกสารสรุปสิ่งที่ได้เรียนรู้และลองทำ Jenkins (หน้า 31)
 - เอกสารสรุปสิ่งที่ได้เรียนรู้และลองทำ Terraform (หน้า 41)
 - เอกสารสรุปสิ่งที่ได้เรียนรู้เกี่ยวกับ Monitoring Tools (หน้า 45)
 - เอกสารสรุปสิ่งที่ได้เรียนรู้เกี่ยวกับ ELK Stack (หน้า 51)

ใบรายงานตัวเข้ารับการฝึกศหกิจ

[2086]

ชื่อนักศึกษา ณัฐพงษ์ เทพพิทักษ์
 รหัสนักศึกษา 640610634
 นักศึกษาสาขาวิชาศุภกรรม คอมพิวเตอร์
 คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่
 เบอร์โทรศัพท์นักศึกษา 0955301640
 เบอร์โทรศัพท์ผู้ปกครอง (กรณีฉุกเฉิน) 0928205909
 ให้รายงานตัวเพื่อเข้ารับการฝึกศหกิจ ณ บริษัท เอสซีปี เทคโนโลยี จำกัด
 ที่อยู่ 19 อาคารไทยพาณิชย์ปาร์ค พลาซ่า เวสท์ปี ชั้นที่ 21 ช. 18 ถ. รัชดาภิเษก แขวงจตุจักร เขตจตุจักร กรุงเทพมหานคร 10900

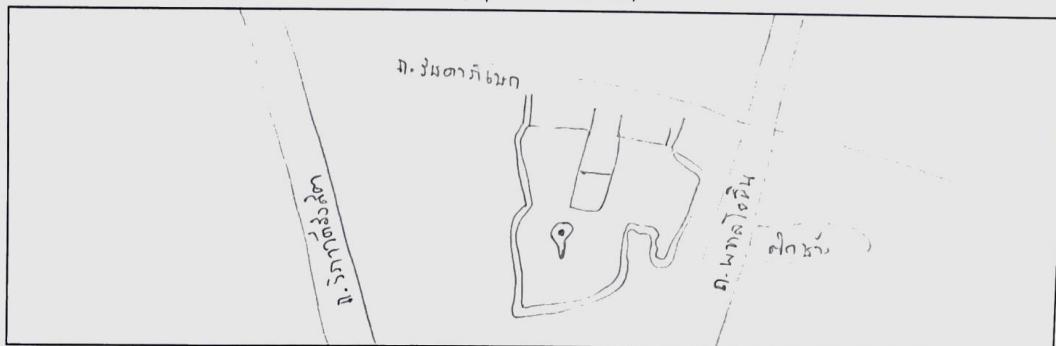
ผู้ควบคุมการฝึกศหกิจ Phataraphon Muakmanee	ตำแหน่ง DevOps Engineer
โทรศัพท์ 0859903803	โทรศัพท์ 028539600
พี่เลี้ยง Phataraphon Muakmanee	โทรศัพท์ 0859903803 อีเมล phataraphon.muakmanee@scbtechx.io
ผู้ประสานงานการนิเทศ Phataraphon Muakmanee	โทรศัพท์ 0859903803 อีเมล phataraphon.muakmanee@scbtechx.io

ลักษณะงานที่นักศึกษารับผิดชอบ Supporting and working on the deployment and development of the development environment.

ระยะเวลาการฝึกศหกิจ 4 มี.ย. 2567 ถึง 31 ต.ค. 2567 เวลาเข้าฝึกศหกิจ 09:00 น. เวลาออกฝึกศหกิจ 18:00 น.

วันหยุดบวชท วันหยุดนันเสาร์ วันหยุดอาทิตย์
 วันหยุดอื่น ๆ 22-Jul-24, 29-Jul-24, 12-Aug-24, 14-Oct-24
 รวมเป็นเวลา 103 วัน (นับเฉพาะวันที่เข้ารับการฝึกศหกิจ)
 ที่พักนักศึกษาจะเดินทางเข้ารับการฝึกศหกิจ JS Residence (Phahon Yothin 23)

แผนที่แสดงตำแหน่งของสถานที่ฝึกศหกิจโดยสังเขป (กรุณาวัดด้วยตนเอง)



ในการฝึกศหกิจครั้งนี้ ข้าพเจ้าจะปฏิบัติตามกฎระเบียบและข้อบังคับ ตลอดจนรับผิดชอบงานที่ได้รับมอบหมายจากทางสถานที่ รับฝึกศหกิจดังกล่าวข้างต้นอย่างเต็มความสามารถ และในการฝึกศหกิจจะไม่นำความลับ ข้อมูล รวมทั้งรูปถ่ายและคลิปวิดีโอ ภายในบริษัทออกมายกเว้นไว้ ณ วันนี้ อนุญาตถูกห้ามไว้

ลงชื่อ..... ก.พ. ๖๗ หมายเหตุ.....

(ณัฐพงษ์ เทพพิทักษ์)

(Phataraphon Muakmanee)

นักศึกษาผู้เข้ารับการฝึกศหกิจ¹
 วันที่ 18 / พฤษภาคม / 2567

ผู้ควบคุมการฝึกศหกิจ²
 วันที่ 18 / พฤษภาคม / 2567

ประทับตรา บริษัท/โรงงาน (ถ้ามี)

Docker Tutorial

What is container :

A process running on host that isolated from host process

What is image :

When we need to run container we need to use an image. Image contain everything needed to run an application - all dependencies, configurations, scripts, binaries. So that means it portable and can be run on any where and any OS.

Docker installation

- Mac
 - Enter this site : <https://docs.docker.com/desktop/install/mac-install/>
 - Then following the description.
- Ubuntu (full docs on this site :
<https://docs.docker.com/engine/install/ubuntu/>)
 - Set up Docker's `apt` repository.
 - install docker package

```
sudo apt-get install \
docker-ce docker-ce-cli \
containerd.io \
docker-buildx-plugin \
docker-compose-plugin \
```

- Add a user group docker to your user via

```
sudo usermod -aG docker $USER
```

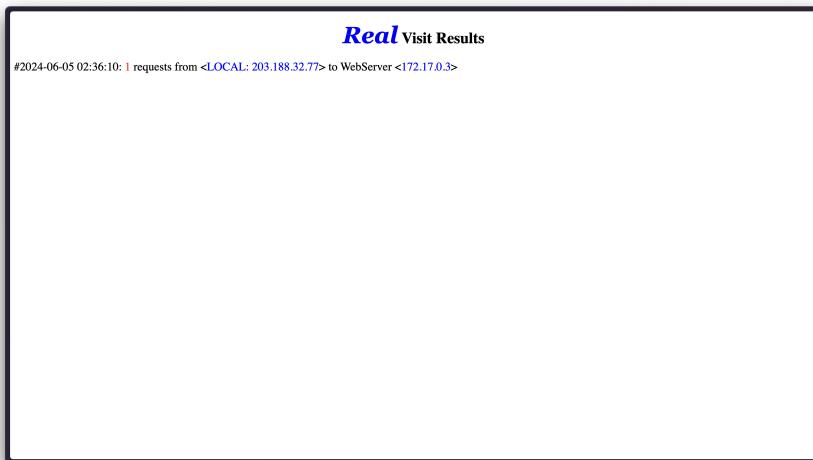
-a mean append user group from exist usergroup
-G mean group

-aG mean append new group from existing group

- Now we are getting and initial setup docker on our device already, next we will try to run some container to show simple page.

```
docker run -dp 80:80 yeasy/simple-web:latest
```

If you are work perfectly you will see the result like this via enter IP:
127.0.0.1 or localhost



Create docker image

when you implement your application you need to build an docker image for that project to be run it on container, in this chapter we will show some simple way to create docker image via dockerFile and try to run.

- Firstly, I will implement some simple nodes application to be an example app for building image.
 - Create empty folder to store our source code.
 - execute this command

```
npm init -y #init node project
```

```
npm i express # install express library
```

- c. go to package.json then add `"type" : "module"` into your package.json file. the result should be like this.

```
{
  "name": "simple-node-app",
  "version": "1.0.0",
  "main": "index.js", //added line
  "type": "module",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "express": "^4.19.2"
  }
}
```

- d. create file index.js then write some simple code like this.

```
import express from "express";
import os from "os";

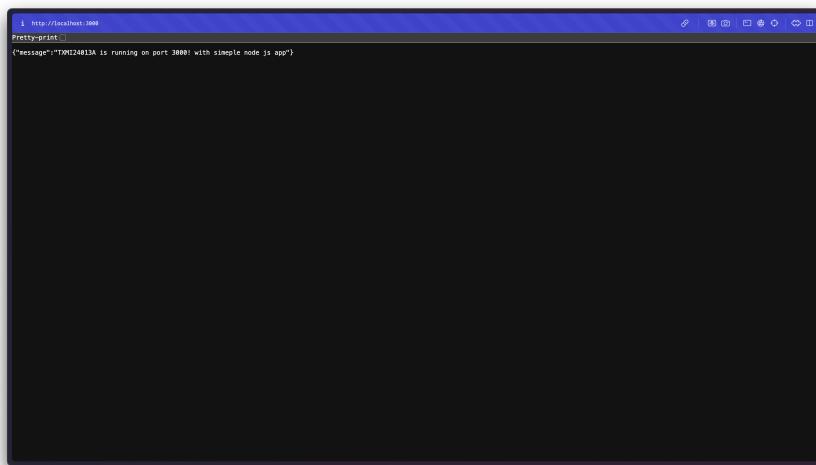
const app = express();
const port = 3000;

app.get("/", (req, res) => {
  res.send({
    message: `${os.hostname()} is running on port ${port}`,
    simple node js app`,
  });
});

app.listen(port, () => {
```

```
        console.log(`Server is running on port ${port}`);
    });
}
```

- e. try to run this app via command `node index.js` , then enter website with localhost:3000 the result should be



- f. Next, we will write DockerFile to create a docker image for this simple app.

- i. Create DockerFile then write some simple script like this

```
// DockerFile

FROM node:lts-alpine
WORKDIR /usr/src/app
COPY ["package.json", "package-lock.json*", "./"]
RUN npm install && mv node_modules ../
COPY .
EXPOSE 3000
RUN chown -R node /usr/src/app
USER node
CMD ["node", "index.js"]
```

- ii. try to build image via `docker build . -t simple-app`

- ,
- iii. if everything work perfectly when you type `docker image` you need to see an simple-app image
- iv. running an app via `docker run -dp 80:3000 simple-app` , then enter website `localhost:80` you will able to see the same result as before running without docker but device name will be change to docker container id.

```
{"message":"5320be4c6af is running on port 3000! with simple node js app"}
```

Push image to registry

- Public registry

Form the previous chapter we had created an docker image call `simple-app` in this part we will push it on DockerHub public registry

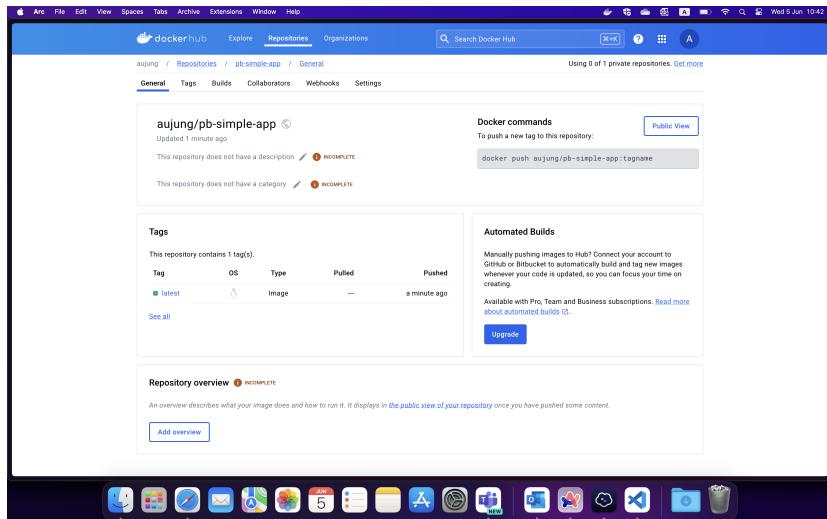
before we push we need to create an repository on docker hub.

- Firstly, we need to give an tag to simple-app before pushing to registry, then push it.

```
docker tag simple-app YOUR-USER-NAME/REPO_NAME
```

```
docker push YOUR-USER-NAME/REPO_NAME
```

- If everything work fine you will see the result look like this



- Private registry

In this part everything is the same as public repository but there are some several thing need to change in execution command.

```
docker tag [OPTIONS] IMAGE[:TAG] [REGISTRYHOST/]USERNA  
 docker push NAME[:TAG]
```

Running container

There are several to run a docker container a way we use it already is run with `docker run` and another version is writing `docker-compose` file, So in this chapter we will create simple docker-compose file to run our simple-app

1. create `docker-compose.yml` then write some script like this.

```
version: "3"  
services:  
  simple-app:  
    image: IMAGETAG //IMAGE TAG FROM previous section  
    ports:  
      - "80:3000"
```

2. run docker `compose up -d` to run container with simple-app image'

Container file system volumes and bind mounts

Each container also gets its own "scratch space" to create/update/remove files. Any changes won't be seen in another container, even if they're using the same image.

volume

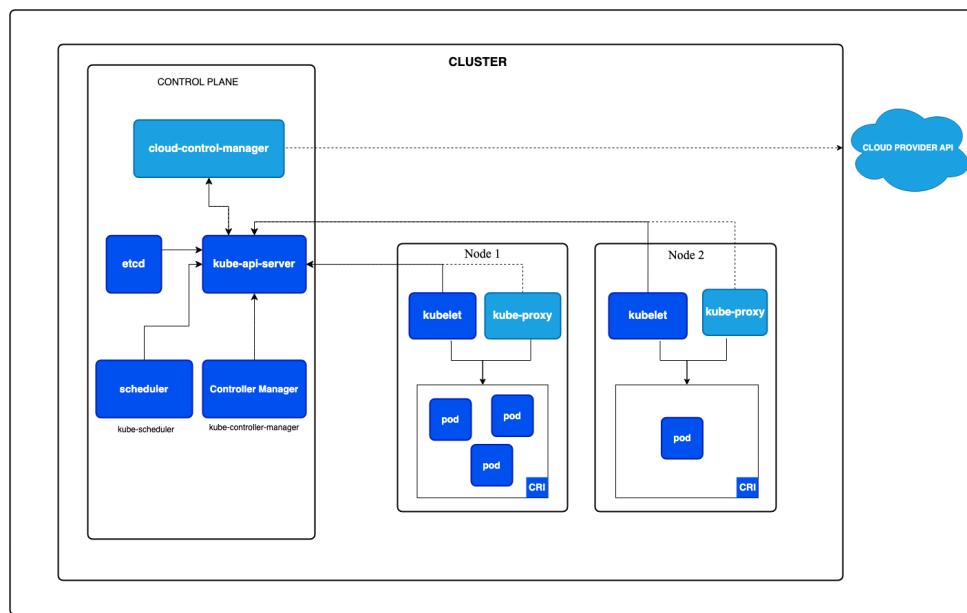
Volumes provide the ability to connect specific filesystem paths of the container back to the host machine. If you mount a directory in the container, changes in that directory are also seen on the host machine. If you mount that same directory across container restarts, you'd see the same files.

Networking

Container networking refers to the ability for containers to connect to and communicate with each other, or to non-Docker workloads.

Kubernetes - K8s

Kubernetes Architecture



In each cluster there are three main components that we need to know before working with K8s

- Master Node (Control plane)
- Worker Node
- etcd

Master Node

- Controlled and managed all of operation in K8s included
 - **kube-api-server** : act as an intermediary for cluster communication via RESTful API
 - **kube-scheduler** : act as an pod manager when new Pos comes in it will find the appropriate worker node for that coming pod which requires information from many sources to make decisions
 - **kube-contoller-manager** :
 - Node Controller
 - Replication Controller : manage pod replication
 - Endpoints Controller

- Service Account & Token controllers : manage service account and API access token
- cloud-controller-manager : this controller is available in only Cloud Provider that support K8s. It act like a kube-controller-manager but working with Cloud Provider instead.

Worker Node

- Container runtime included two services
 - kubelet : Receive commands from kube-api-server and manage container runtime, such as checking if Pods (Containers) are still running on Worker nodes. If there is a problem, it will report to kube-api-server that there is a problem with the Pods and make them manage the Initial Pods again. (manager of Pods)
 - kube-proxy : this service help to make Pods can be connected from outside the Cluster it act like a proxy server to receive request from outside then resend again to appropriate pods inside the node.

etcd

Use to store all of Persistance cluster state this components also in control plane which directly connects by kube-api-server

Networking in Kubernetes

- **Container-to-container** communication within a Pod
- **Pod-to-Pod** communication across nodes
- **Pod-to-Service** communication within the cluster
- **External-to-Service** communication with the outside world

Type of Objects in Kubernetes

- Pods : A Pod can host a single container or a group of containers. It's important to note that, when a Pod contains multiple containers, all of the containers always run on a single worker node. **Pods is that they are ephemeral in nature**
- Deployment : **allows you to declaratively manage the desired state of your application**, such as the number of replicas, the image to use for the Pods, and the resources required.
- ReplicaSets : **Deployments don't manage Pods directly**. That's the job of the ReplicaSet object.
 - ReplicaSet is created automatically

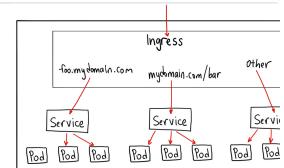
- The ReplicaSet ensures that the desired number of replicas (copies) are running at all times by creating or deleting Pods as needed.
- **StatefulSet**
 - **ensures that each Pod is uniquely identified by a number, starting at zero.**
 - When a Pod in a StatefulSet must be replaced, for example, due to node failure, the new Pod is assigned the same numeric identifier as the previous Pod.
- **DaemonSets**
 - **ensures that a copy of a Pod is running across all, or a subset of nodes in a Kubernetes cluster.**
- **PersistentVolume**
 - **represents a piece of storage that you can attach to your Pod(s).**
 - if your Pod gets deleted, the PersistentVolume will survive.
 - can attach using a PersistentVolume, like local disks, network storage, and cloud storage.
- Service : **a way to access a group of Pods that provide the same functionality.** the Service also provides some simple load balancing.
- Namespaces
 - **a way to divide a single Kubernetes cluster into multiple virtual clusters.**

ClusterIP vs NodePort vs Loadbalancer

Kubernetes NodePort vs LoadBalancer vs Ingress? When should I use what?

Recently, someone asked me what the difference between NodePorts, LoadBalancers, and Ingress were. They are all different ways to get...

 <https://medium.com/google-cloud/kubernetes-nodeport-vs-loadbalancer-vs-ingress-when-should-i-use-what-922f010849e0>



- ClusterIP : the default Kubernetes service. It gives you a service inside your cluster that other apps inside your cluster can access. There is no external access.
- NodePort : **is a way to expose your application to external clients but doesn't contain LoadBalancer**
- LoadBalancer : is NodePort bit including LoadBalancer
 - key different NodePort and LoadBalancer
 - NodePort : Client → node
 - LoadBalancer : client → load balancer → node

Deploying Nginx on Kuberetes Various way.

,

Deploying Nginx on Kubernetes: Exploring Various Methods
Mastering Nginx Deployment in Kubernetes: A Comprehensive Guide to Pods, Deployments, and Beyond
https://medium.com/@muppedaanvesh/deploying-nginx-on-kubernetes-a-quick-guide-04d533414967

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

ConfigMap & Secret

- ConfigMap
 - non-confidential
 - key-value format
 - can be injected into container
- Secret

Kubernetes Config & Secrets: Essential Guide to ConfigMaps
Explore managing Kubernetes ConfigMaps & Secrets for efficient app deployment. Learn best practices & essential tips for containerized environments...
<https://www.getambassador.io/blog/kubernetes-configurations-secrets-configmaps>

Blog
Managing Configurations and Secrets in Kubernetes: ConfigMaps and Secrets

- The same as ConfigMap
- sensitive information

Workshop

Initially, I use docker desktop on Mac and enable to use an Kubernetes already,

- first I will check if Kubernetes is working by
 - run `kubectl cluster-info` to check if there is any response to me
 - the response is

```
kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:6443
CoreDNS is running at https://127.0.0.1:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

- Next I will try to deploy Nginx with many method and strategy. Before we start we need to create new namespace for this learning section via `kubectl create namespace learning`. After we created let's check the result if everything work find let's get started.

```
kubectl get namespace
```

NAME	STATUS	AGE
default	Active	20h
kube-node-lease	Active	20h
kube-public	Active	20h
kube-system	Active	20h
learning	Active	18s

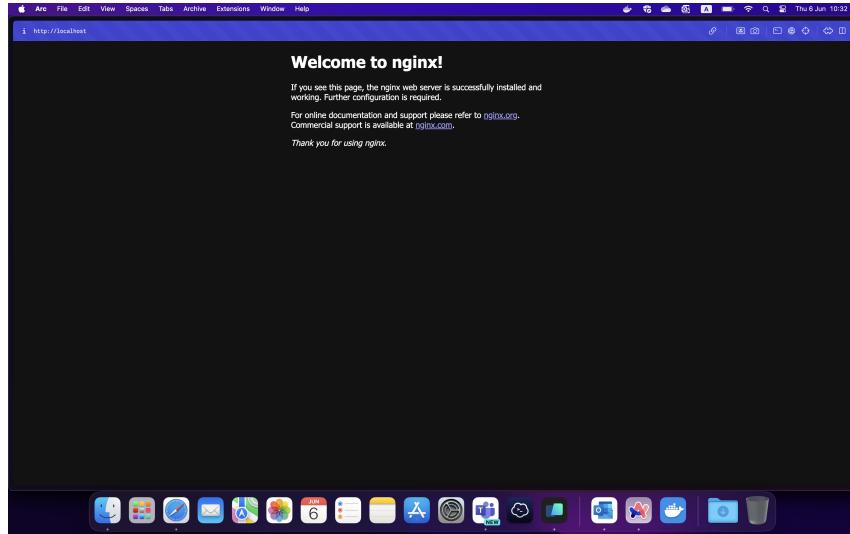
- Imperative Way

- RUN `kubectl run nginx-pod --image=nginx --port=80 -n learning`
- Now check the pod that created

```
kubectl get pod -n learning
```

NAME	READY	STATUS	RESTARTS	AGE
nginx-pod	1/1	Running	0	2m11s

- Now if you need to enter this site this deployed, Sorry you can't because you need to setup for some service that would be the door that external user need to connect to. let's type `kubectl expose pod nginx-pod --type=LoadBalancer --port=80 --name=nginx-service --namespace=learning`
- then check the result in website on localhost:80 should be look like this.



- Use a manifest file

- Create a file call nginx.yaml and copy this script to create pod and service like above.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  labels:
    app: nginx
    namespace: learning
spec:
  containers:
    - name: nginx-container
      image: nginx:latest
      ports:
        - containerPort: 80
  resources:
    limits:
      memory: "128Mi"
      cpu: "500m"
    requests:
      memory: "64Mi"
      cpu: "250m"

  ---
apiVersion: v1
kind: Service
```

```
,  
  
  metadata:  
    name: nginx-service  
    namespace: learning  
  spec:  
    selector:  
      app: nginx  
    ports:  
      - protocol: TCP  
        port: 80  
        targetPort: 80  
    type: LoadBalancer
```

- after that save the file then run `kubectl apply -f nginx.yaml`

```
~/personal/learnKube * docker-desktop (0.228s)  
kubectl apply -f nginx.yaml  
pod/nginx-pod unchanged  
service/nginx-service unchanged
```

- now check the result.

Jenkins

Installation

- Terraform install script
 - Vm setup is the same as Terraform
 - install Jenkins script

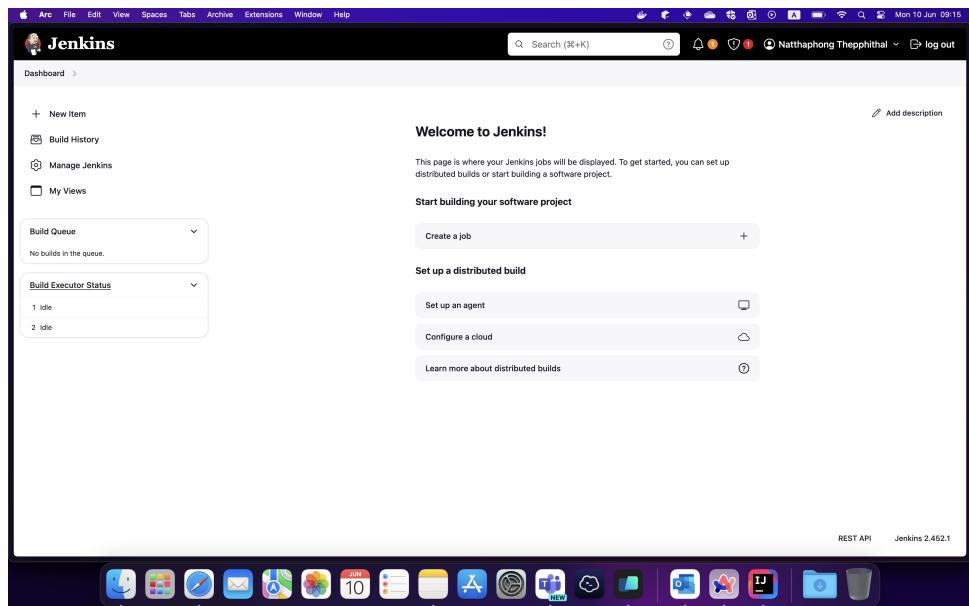
```
resource "null_resource" "install_jenkins" {  
    triggers = {  
        vm_id = azurerm_linux_virtual_machine.vm.id  
    }  
  
    connection {  
        type      = "ssh"  
        host      = azurerm_linux_virtual_machine.vm.p  
        user      = azurerm_linux_virtual_machine.vm.a  
        private_key = file("~/ssh/id_rsa")  
    }  
  
    provisioner "remote-exec" {  
        inline = [  
            "sudo apt-get update -y",  
            "sudo apt-get install -y openjdk-11-jdk",  
            "sudo wget -O /usr/share/keyrings/jenkins-keyr  
            https://pkg.jenkins.io/debian-stable/jenkins.i  
            "echo \"deb [signed-by=/usr/share/keyrings/jen  
            https://pkg.jenkins.io/debian-stable binary/\\""  
            sudo tee /etc/apt/sources.list.d/jenkins.list :  
            "sudo apt-get update -y",  
            "sudo apt-get install -y jenkins",  
            "sudo systemctl start jenkins",  
            "sudo systemctl enable jenkins"  
        ]  
    }  
}
```

enter ip-address:8080

- After that do following Jenkins instructions.

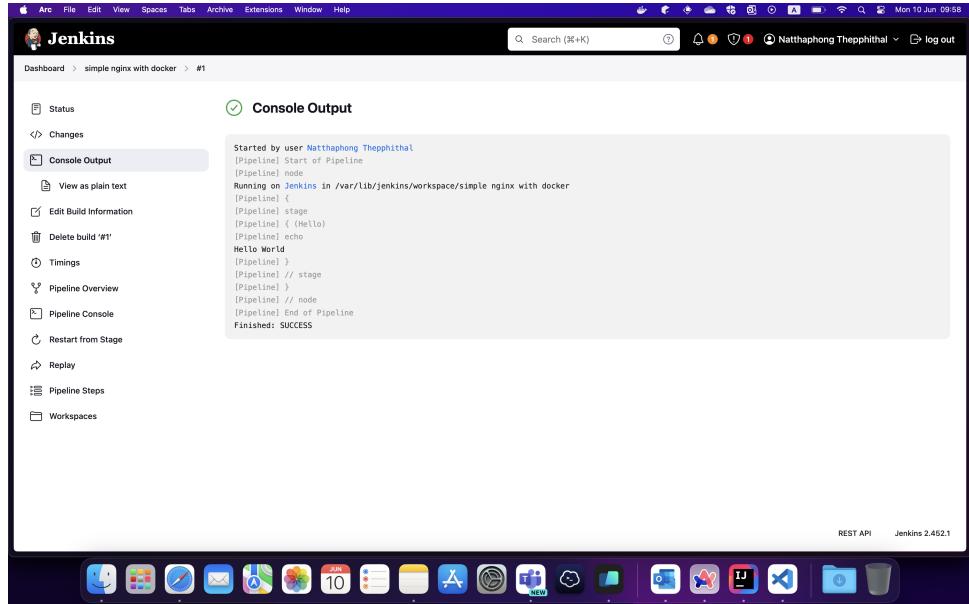


If done with initial setup will see :



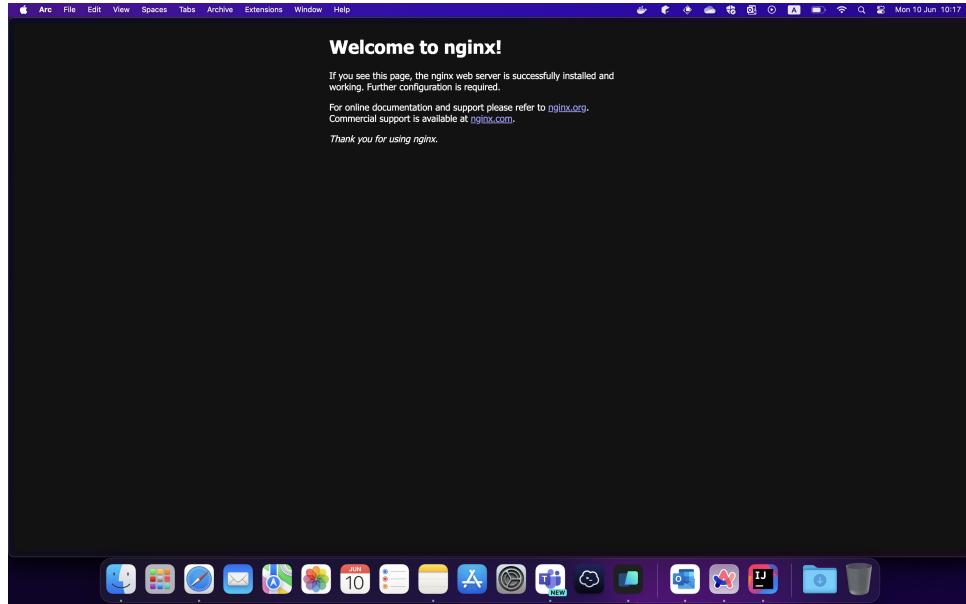
Create first simple groovy script for run Nginx on docker.

- Firstly, let's test more simple script to check if jenkins is working.



```
pipeline {
    agent any

    stages {
        stage('Hello') {
            steps {
                echo 'Hello World'
            }
        }
    }
}
```



- now create let pull Nginx docker image and run it with Jenkins.

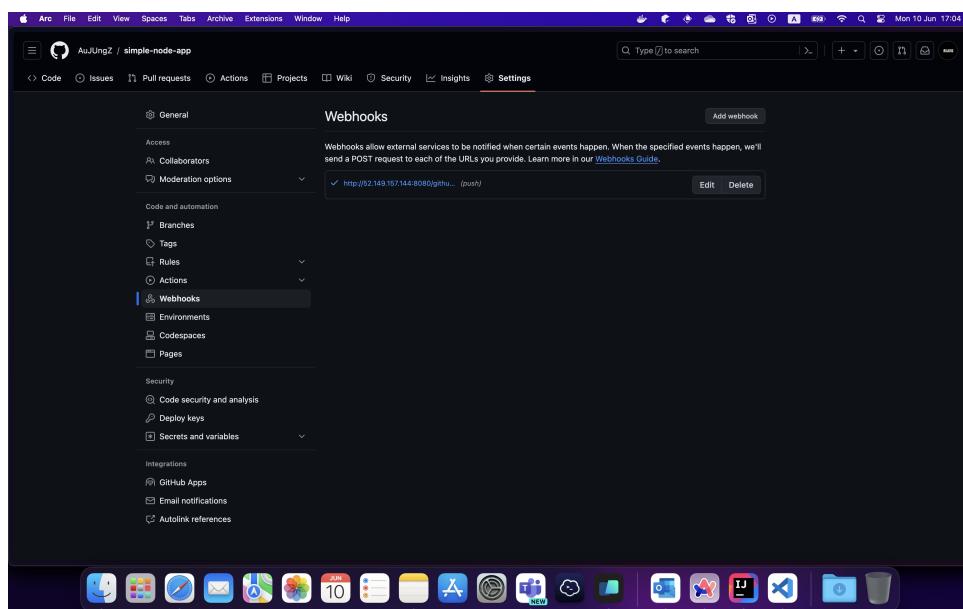
```
pipeline {  
    agent any  
  
    stages {  
        stage('Pull Docker Image') {  
            steps {  
                script {  
                    sh 'docker pull nginx:alpine'  
                }  
            }  
        }  
  
        stage('Remove Docker Container') {  
            steps {  
                script {  
                    sh 'docker rm -f mynginx'  
                }  
            }  
        }  
    }  
}
```

```
        }
    }

    stage('Run Docker Container') {
        steps {
            script {
                sh 'docker run -d -p 80:80 --name myng'
            }
        }
    }
}
```

Github web hook with Jenkins

- add Jenkins web hook for GitHub web hook add `/github-webhook` at the end of Jenkins url.



- Now when on git repository has new commit Jenkins pipeline will automatically redo again.

Set Slave agent

- master Node
 - install java jdk and install jenkins sercie.
 - generate ssh key for that master node via `ssh-keygen`
- Slave Node
 - install only **java jdk**.
 - place public ssh key of master agent on working user on slave node.

Master Node UI settings.

- Create global credential that store ssh private key of Master Node.
- Connect to slave node
 - add new node with ssh method with created credential

Terraform

Installation

- Use Homebrew package manager.

```
brew tap hashicorp/tap
brew install hashicorp/tap/terraform
```

- Try simple to use terraform on local machine.

- Create file .tf

```
#To create a file that contain some text

resource "null_resource" "file-create" {
    provisioner "local-exec" {
        command = "echo 'Hello, World!' > hello.txt"
    }
}

terraform init
terraform plan
terraform apply
```

- Now check inside project folder you will see `hello.txt` file, inside contain "Hello, world"

Terraform with azure

- After installed already if we need to use terraform with any provider we need to do **Authenticating to Azure**
 - **Authenticating to Azure using the Azure CLI**
 - `az login` (login via website method)
 - `az account set --subscription="SUBSCRIPTION_ID"`
 - Let's create simple script for create

- resource group
- storage account including
 - web hosting file
- new file call main.tf

```

provider "azurerm" {
  features {}
}

#create a resource group
resource "azurerm_resource_group" "rg" {
  name      = "rg-learnTerraform"
  location  = "East Asia"
}

#create a Storage Account
resource "azurerm_storage_account" "storage" {
  name                  = "aujungterraformstorage"
  resource_group_name   = azurerm_resource_group.rg.name
  location              = azurerm_resource_group.rg.location
  account_tier          = "Standard"
  account_replication_type = "LRS"
  account_kind = "StorageV2" // StorageV2 is required for static website

  static_website {
    index_document = "index.html"
  }
}

#add index.html file
resource "azurerm_storage_blob" "blob" {
  name            = "index.html"
  storage_account_name = azurerm_storage_account.storage.name
  storage_container_name = "$web"
  type            = "Block"
  content_type    = "text/html"
}

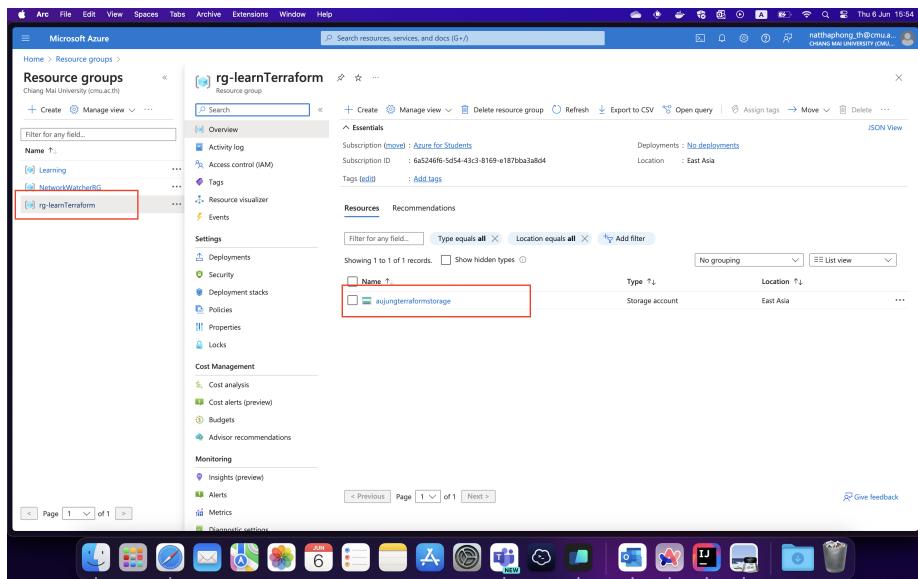
```

```
source_content = "<html><body><h1>Hello, Terraform! from
}
```

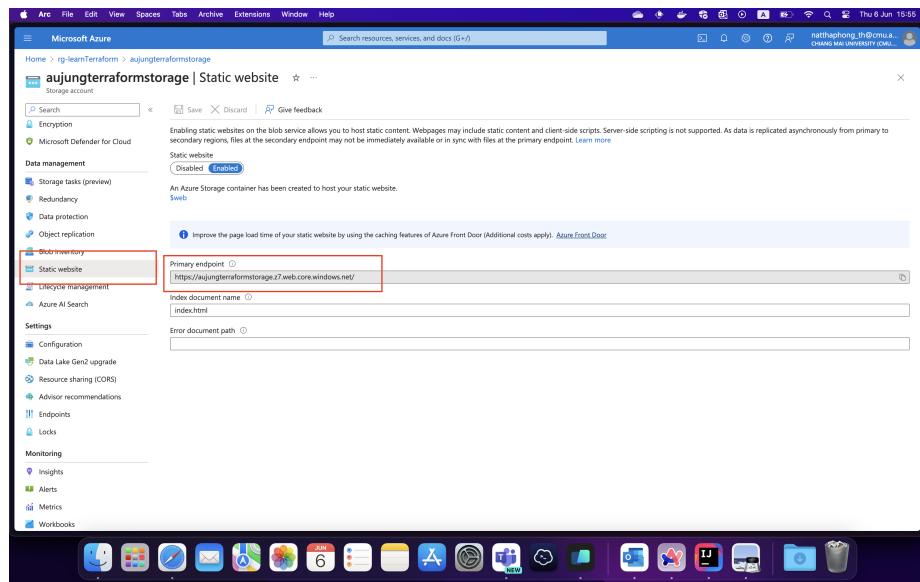
- after that execute this respectively

```
$ terraform init
$ terraform plan
$ terraform apply
```

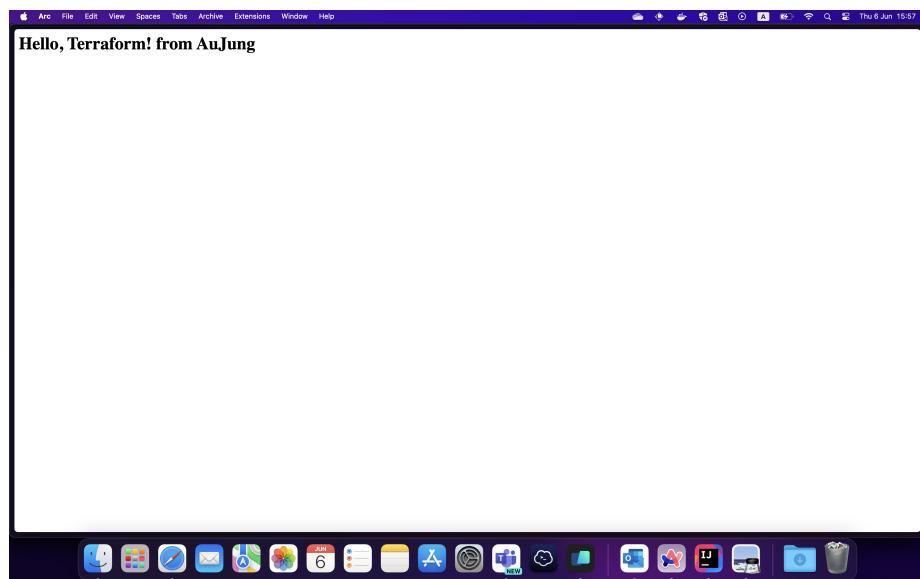
- Everything work fine we will see an resource that was created in azure portal like this.



- Let's see an static web that we created via



- the result on that web should be this.



Create Vm and setup docker with terraform

```
#main.tf
```

```

provider "azurerm" {
    features {}
}

# Create a resource group
resource "azurerm_resource_group" "rg" {
    location = "East Asia"
    name     = "aujung-rg"
}

# Create a virtual network
resource "azurerm_virtual_network" "vnet" {
    name       = "aujung-vnet"
    address_space = ["10.0.0.0/16"]
    location   = azurerm_resource_group.rg.location
    resource_group_name = azurerm_resource_group.rg.name
}

# Create a subnet
resource "azurerm_subnet" "subnet" {
    name           = "aujung-subnet"
    resource_group_name = azurerm_resource_group.rg.name
    virtual_network_name = azurerm_virtual_network.vnet.name
    address_prefixes      = ["10.0.0.0/24"]

}

# Create a public IP
resource "azurerm_public_ip" "ip" {
    location          = azurerm_resource_group.rg.location
    name              = "aujung-ip"
    resource_group_name = azurerm_resource_group.rg.name
    allocation_method = "Dynamic"
}

# Create a network interface

```

```

,
resource "azurerm_network_interface" "nic" {
    location          = azurerm_resource_group.rg.location
    name              = "aujung-nic"
    resource_group_name = azurerm_resource_group.rg.name

    ip_configuration {
        name                = "aujung-ipconfig"
        subnet_id           = azurerm_subnet.subnet.id
        private_ip_address_allocation = "Dynamic"
        public_ip_address_id = azurerm_public_ip.ip.id
    }
}

# Setup inbound security rules
resource "azurerm_network_security_group" "nsg" {
    location          = azurerm_resource_group.rg.location
    name              = "aujung-nsg"
    resource_group_name = azurerm_resource_group.rg.name

    security_rule {
        name          = "SSH"
        priority      = 1001
        direction     = "Inbound"
        access        = "Allow"
        protocol      = "Tcp"
        source_port_range = "*"
        destination_port_range = "22"
        source_address_prefix = "*"
        destination_address_prefix = "*"
    }

    security_rule {
        name          = "HTTP"
        priority      = 1002
        direction     = "Inbound"
        access        = "Allow"
        protocol      = "Tcp"
        source_port_range = "*"
    }
}

```

```

        ,
        destination_port_range      = "80"
        source_address_prefix       = "*"
        destination_address_prefix = "*"
    }
}

# Apply the network security group to vm's network interface
resource "azurerm_network_interface_security_group_association" "vm_nic" {
    network_interface_id      = azurerm_network_interface.nic.id
    network_security_group_id = azurerm_network_security_group.id
}

# Create a vm
resource "azurerm_linux_virtual_machine" "vm" {
    admin_username      = "aujung"
    location           = azurerm_resource_group.rg.location
    name               = "aujung-vm"
    network_interface_ids = [azurerm_network_interface.nic.id]
    resource_group_name = azurerm_resource_group.rg.name
    size               = "Standard_B1s"

    admin_ssh_key {
        public_key = file("~/ssh/id_rsa.pub")
        username   = "aujung"
    }

    os_disk {
        caching          = "ReadWrite"
        storage_account_type = "Standard_LRS"
        disk_size_gb = 30
    }

    source_image_reference {
        publisher = "Canonical"
        offer     = "0001-com-ubuntu-server-jammy"
        sku       = "22_04-lts"
        version   = "latest"
    }
}

```

```

        ,
    }

}

# setup-docker.tf

# Install Docker on the virtual machine
resource "null_resource" "install_docker" {
    triggers = {
        vm_id = azurerm_linux_virtual_machine.vm.id
    }

    connection {
        type          = "ssh"
        host          = azurerm_linux_virtual_machine.vm.public_ip_address
        user          = azurerm_linux_virtual_machine.vm.admin_username
        private_key   = file("~/ssh/id_rsa")
    }

    # Remove old Docker packages
    provisioner "remote-exec" {
        inline = [
            "sudo apt-get remove -y docker docker-engine docker.io"
        ]
    }

    # Install Docker
    provisioner "remote-exec" {
        inline = [
            "sudo apt-get update -y",
            "sudo apt-get install -y -o=APT::Get::Assume-Yes=true curl",
            "sudo rm -rf /etc/apt/keyrings",
            "sudo mkdir -p /etc/apt/keyrings",
            "curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg",
            "echo \"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable\" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null"
        ]
    }
}

```

```

        ,

    "sudo apt-get update -y",
    "sudo apt-get install -y -o=APT::Get::Assume-
    Yes=true
    docker-ce
    docker-ce-cli
    containerd.io",
    "sudo usermod -aG docker ${azurerm_linux_virtual_machine.username}"
]
}

# Configure Docker daemon
provisioner "remote-exec" {
  inline = [
    "sudo mkdir -p /etc/docker",
    "sudo tee /etc/docker/daemon.json > /dev/null <<EOF",
    "{\"",
    "  \"log-driver\": \"json-file\",
    \"  \"log-opt\": {",
    "    \"max-size\": \"10m\",
    "    \"max-file\": \"3\",
    "  }",
    "}",
    "EOF",
    "sudo systemctl daemon-reload",
    "sudo systemctl restart docker",
  ]
}
}

# Deploy nginx container

resource "null_resource" "nginx" {
  depends_on = [null_resource.install_docker]

  triggers = {
    vm_id = azurerm_linux_virtual_machine.vm.id
  }
}

```

```
,  
  
connection {  
    type = "ssh"  
    host = azurerm_linux_virtual_machine.vm.public_ip_address  
    user = azurerm_linux_virtual_machine.vm.admin_username  
    private_key = file("~/ssh/id_rsa")  
}  
  
provisioner "remote-exec" {  
    inline = [  
        "sudo docker run -d -p 80:80 --name nginx nginx",  
    ]  
}  
}
```

Grafana & Prometheus

Grafana is an open-source analytics and interactive visualization web application used for monitoring application performance.

- Need data source to displayed and visualized at this time we will use an Prometheus as datasource

Installation

- via docker-compose file

```
version: '3'
services:
  prometheus:
    image: prom/prometheus:latest
    volumes:
      - ./prometheus/prometheus.yml:/etc/prometheus/prometheus.yml
      - ./prometheus_data:/prometheus
    command:
      - --config.file=/etc/prometheus/prometheus.yml
      - --storage.tsdb.path=/prometheus
      - --storage.tsdb.retention=${PROMETHEUS_RETENTION:-2d}
    restart: always
    networks:
      - monitor-net
  node-exporter:
    image: prom/node-exporter:latest
    networks:
      - monitor-net
  grafana:
    image: grafana/grafana:latest
    volumes:
      - ./grafana_data:/var/lib/grafana
    depends_on:
      - prometheus
    ports:
```

```
,
```

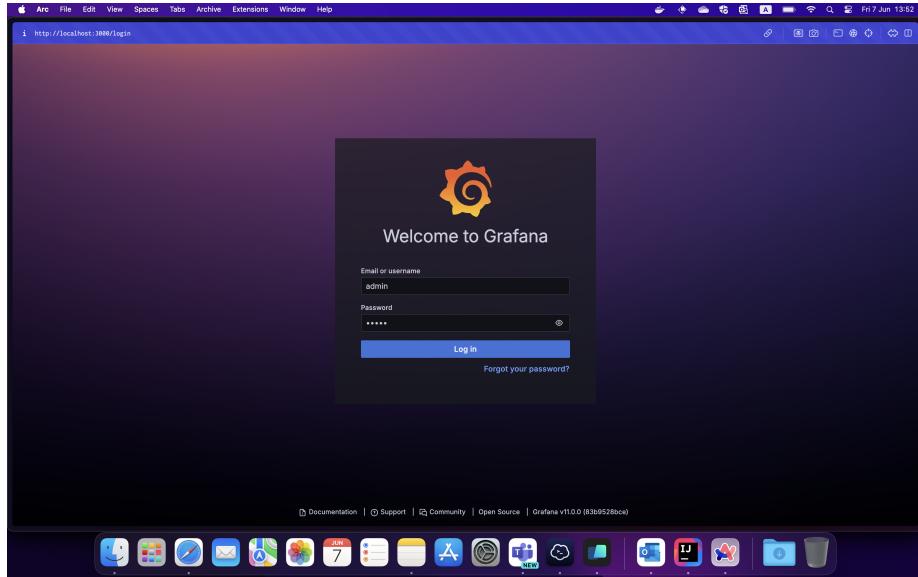
```
    - 3000:3000
networks:
  - monitor-net

networks:
  monitor-net:
    driver: bridge
```

```
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds. The default is 15s.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is 15s.

scrape_configs:
  - job_name: 'job-prometheus'
    scrape_interval: 9s
    static_configs:
      - targets: ['prometheus:9090']
  - job_name: "job-node-exporter"
    scrape_interval: 9s
    static_configs:
      - targets: ["node-exporter:9100"]
```

- `docker-compose up -d`



- After, we got grafana webpage let create some dash board to minitor you devices
 - Add data sourcce
 - Click on connection
 - find Prometheus options, then click add new data source
 - connection url : <http://prometheus:9090> (permetheus come from container name in docker compose file)
 - Click Save & Test
 - Go back to dashboard menu
 - Click new → import
 - go to this website to choose some build in dashboard.
<https://grafana.com/grafana/dashboards/1860-node-exporter-full/>
 - Copy dashboard id then paste it on grafana web site then click load.



ELK Stack

Visualize Apache Logs With Elastic Stack on Ubuntu

- Update system package and install java runtime

```
$ sudo apt-get update && sudo apt-get upgrade  
  
$ sudo apt-get install default-jre-headless
```

- Install the Elastic APT Repository → this package repositories contain all of the necessary packages include Elasticsearch, log stash and kibana

```
$ wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch  
sudo apt-key add -  
$ echo "deb https://artifacts.elastic.co/packages/7.x/apt ."  
sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list  
$ sudo apt-get update
```

- Install Elasticsearch

1. `sudo apt-get install elasticsearch`

2. Set up heap size for JVM

```
File: /etc/elasticsearch/jvm.options  
  
-Xmx(one-quarter of your server's available memory)  
  
Ex.  
-Xmx1g (if host have 4g of memory)
```

3. Start elasticsearch

```
$ sudo systemctl enable elasticsearch  
$ sudo systemctl start elasticsearch
```

4. confirm that the Elasticsearch API is available

```
$ curl localhost:9200

response should be :

{

  "name" : "vm-module",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "TZ0bzoEGSaKuUaktiQQLQQ",
  "version" : {
    "number" : "7.17.21",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "d38e4b028f4a9784bb74de339ac1b877e2d",
    "build_date" : "2024-04-26T04:36:26.745220156Z",
    "build_snapshot" : false,
    "lucene_version" : "8.11.3",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

- Install logstash and cabana
 - `sudo apt-get install logstash`
 - `sudo apt-get install kibana`
- Configure the Elastic Stack
 - Overwrite Elasticsearch setup from create multiple shard to use only one shard.
 - Create a temporary JSON

```
{
  "index_patterns": ["*"],
  "template": {
    "settings": {
      "index": {
        "number_of_shards": 1,
```

```
        "number_of_replicas": 0
    }
}
}
}
```

- Use `curl` to create an index template with these settings that is applied to all indices created hereafter:

```
$ curl -XPUT -H'Content-type: application/json' http://localhost:9200/_index_template/defaults -d @template.json
```

- Configure Logstash

collect Apache access logs, Logstash must be configured to watch any necessary files and then process them, eventually sending them to Elasticsearch.

- Set up heap size for logstash at `/etc/logstash/jvm.options`
- Create the following Logstash configuration

```
File: /etc/logstash/conf.d/apache.conf

input {
  file {
    path => '/var/www/*/logs/access.log'
    start_position => 'beginning'
  }
}

filter {
  grok {
    match => { "message" => "%{COMBINEDAPACHELOG}" }
  }
}

output {
```

```
        elasticsearch { }  
    }
```

- Start all service

```
$ sudo systemctl enable logstash  
$ sudo systemctl start logstash  
  
$ sudo systemctl enable kibana  
$ sudo systemctl start kibana
```

- Install apache2 for test log collecting

- `sudo apt-get install apache2`
- change logs dir to appropriate with logstash configuration

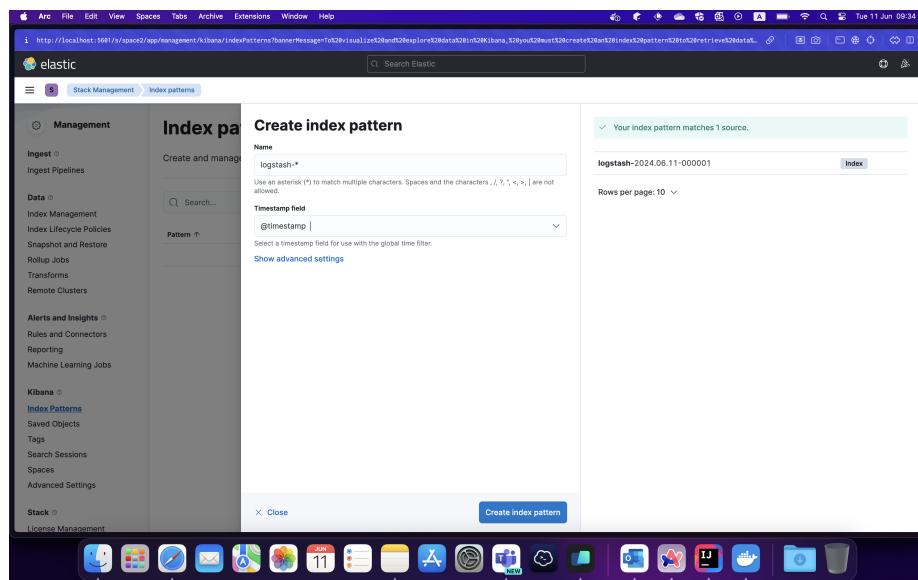
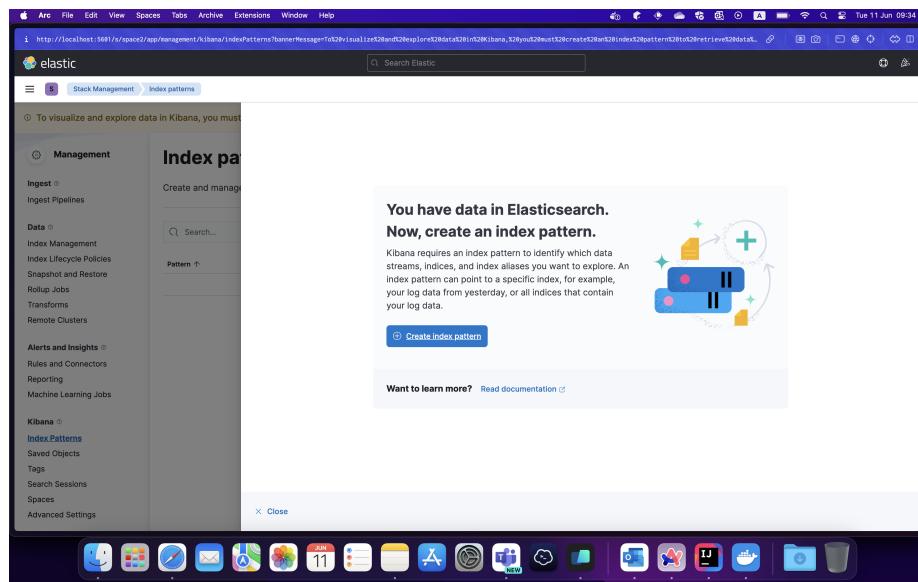
```
$ vim /etc/apache2/sites-available/000-default.conf  
  
ErrorLog /var/www/html/logs/error.log  
CustomLog /var/www/html/logs/access.log combined  
  
$ sudo mkdir /var/www/html/logs  
  
$ sudo systemctl restart apache2
```

- For testing we need some logs data so that we need to enter apache2 landing page to get logs via : `for i in `seq 1 5` ; do curl localhost ; sleep 0.2 ; done`

Watching Logs

- By default, Kibana binds to the local address `127.0.0.1`. This only permits connections that originate from localhost. This is recommended in order to avoid exposing the dashboard to the public internet. use `ssh` command can forward the port to your workstation.
 - `ssh -N -L 5601:localhost:5601 Username@<IP address of kibana>`
- Open Kibana in your browser at <http://localhost:5601>.

- Create an index pattern to collect logs data



- Now when back to Discover page again will see

