BIG DATA THAILAND

Government Big Data Institute

สถาบันส่งเสริมการวิเคราะห์และบริหารข้อมูลขนาดใหญ่ภาครัฐ

Microservices

Kittisak Sa-adaem

Project Manager and Senior Data Engineer

GBDi : Government Big Data Institute

Email: kittisak.sa@depa.or.th

Training : Microservices

Morning :

- Introduction 30 mins
- Cloud Technology (GCP) 30 mins
- Microservice Design 30 mins
- Restful API - Preparation 30 mins

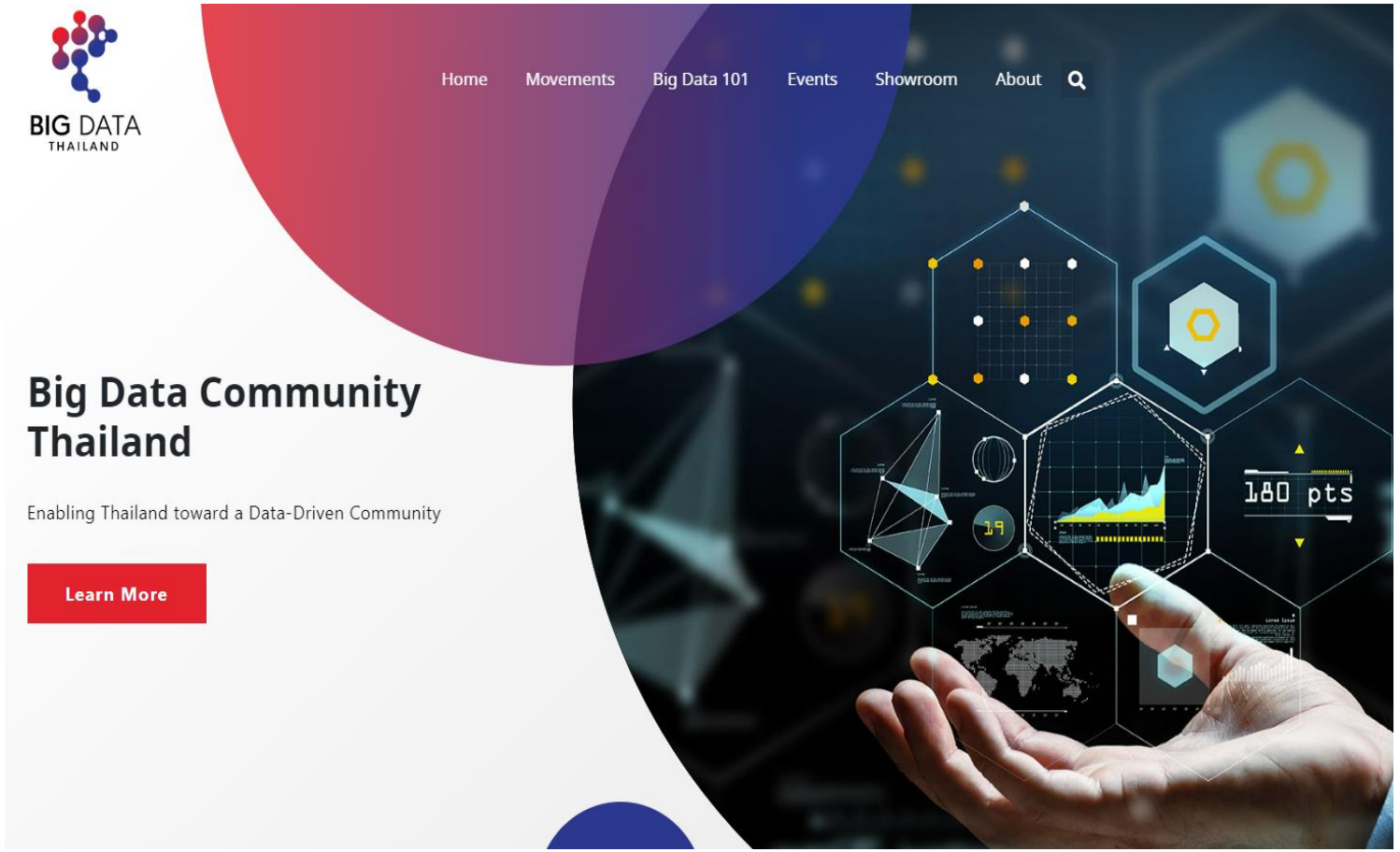
Afternoon :

- Restful API - Lab 30 mins (Continue)
- Kong Gateway 60 mins
- Kong Gateway Lab 120 mins

GBDi
Government Big Data Institute
สถาบันส่งเสริมการวิเคราะห์และบริหารข้อมูลขนาดใหญ่ภาครัฐ (สวข.)

depa

https://bigdata.go.th/



สถาบันส่งเสริมการวิเคราะห์และบริหารข้อมูลขนาดใหญ่ภาครัฐ (สวข.)

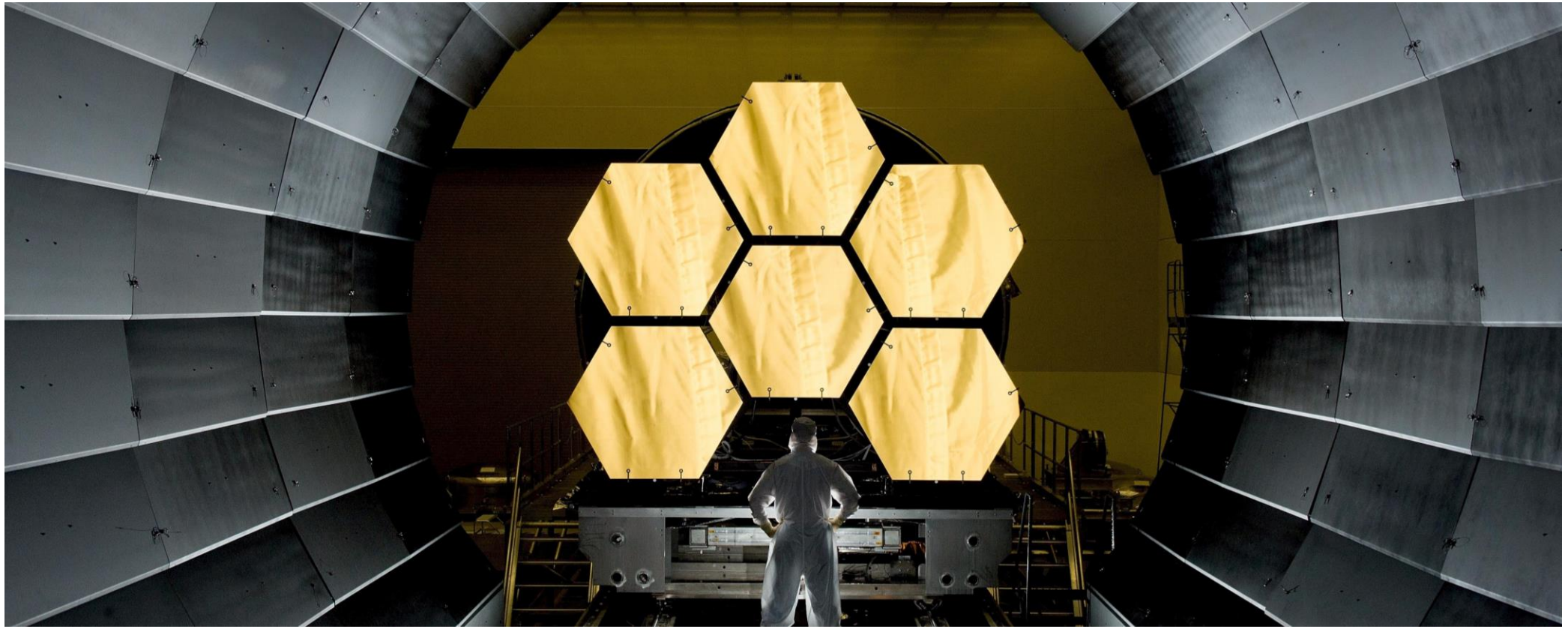
Government Big Data Institute (GBDi)

Address: 80 Lat Phrao 4 Alley, Chom Phon, Chatuchak, Bangkok 10900

Telephone: 02 026 2333 ext. 2508, 2525, 2528

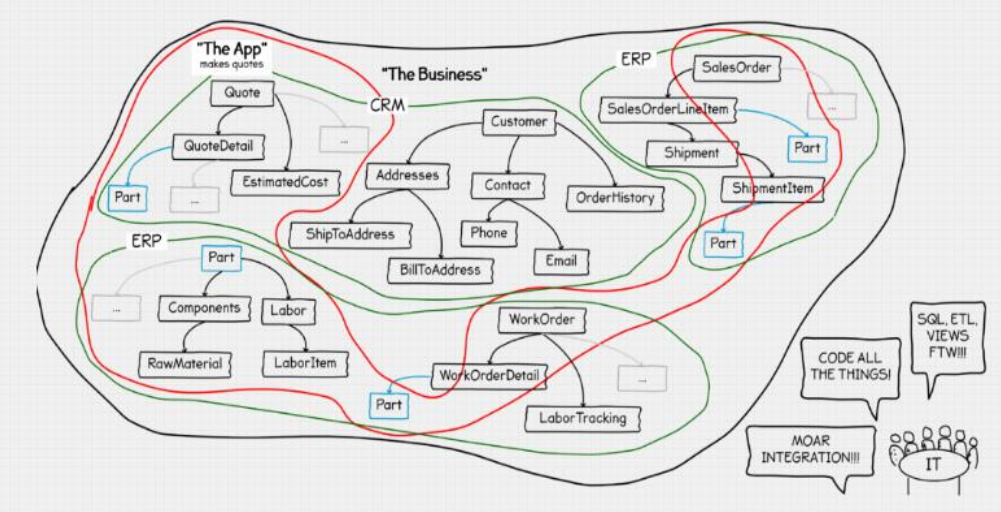
Email: gbdi-info@depa.or.th

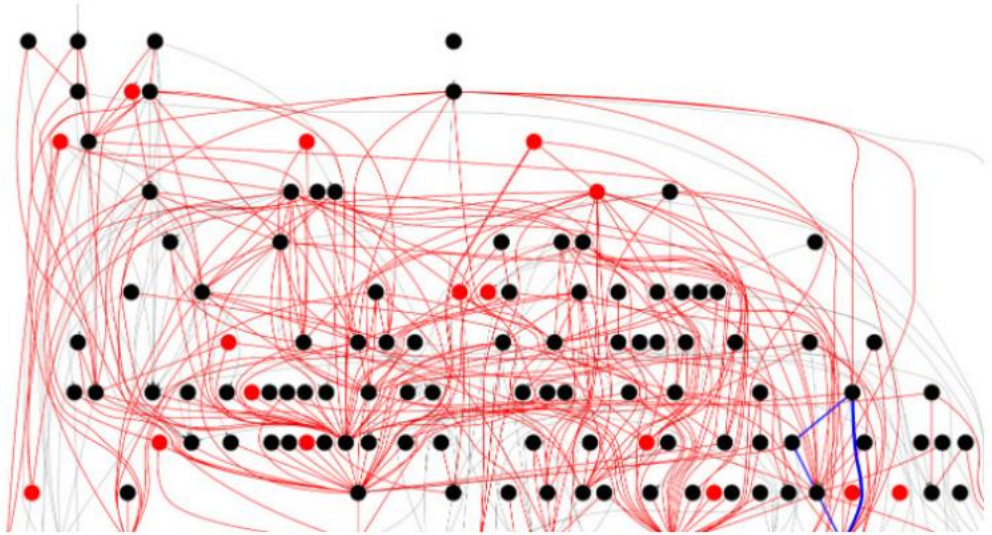
Working hours: Mon – Fri . 9.00 – 17.00



Introduction

Architect Hell/Dependency Hell





Microservice : เป็นเทคนิคหนึ่งในการพัฒนาระบบงาน
เพื่อแยกส่วนการทำงานออกเป็น service ย่อย ๆ ที่มีขนาดเล็ก **(คำถามคือ อะไรคือคำว่าเล็ก ?)**
แต่ละ service มีการทำงานเพียงอย่างเดียว (Single Responsibility)
แต่ละ service ต้องให้ทำงานจบในตัวเอง (Self service) นั่นคือมี data store หรือที่จัดเก็บข้อมูลของแต่ละ service

ผลที่ได้คือ
Service ง่ายต่อการทำความเข้าใจ
Service ง่ายต่อการพัฒนา
Service ง่ายต่อการทดสอบ
Service ง่ายต่อการ deploy
Service ง่ายต่อการ scale

https://www.somkiat.cc/microservices-journey/

https://www.linkedin.com/pulse/dependency-hell-microservices-how-avoid-nabil-hijazi/

Benefit

Micro Services vs. Traditional Apps

Traditional Apps

- Often called "Monolithic Apps"
  - All your code is in one bundle
- Scaling issues
  - Everything or nothing approach
  - Scaling your search system also scales your admin system
    - Not very efficient
- Performance issues
  - One slow part of the code can affect the entire site
- Issues with continuous integration
  - You need to update everything to push a minor change
- Stability issues
  - One bug can bring the entire system down

Micro Services vs. Traditional Apps

Micro Services

A collection of smaller applications all working together to deliver a total experience to the end user.

Increased efficiency

- Splitting your services gives you the ability to scale only the parts of the site that is slow
  - Less wastage of service resource
  - More cost efficient
- An individual slow performing service doesn't slow all services
  - Less user frustration

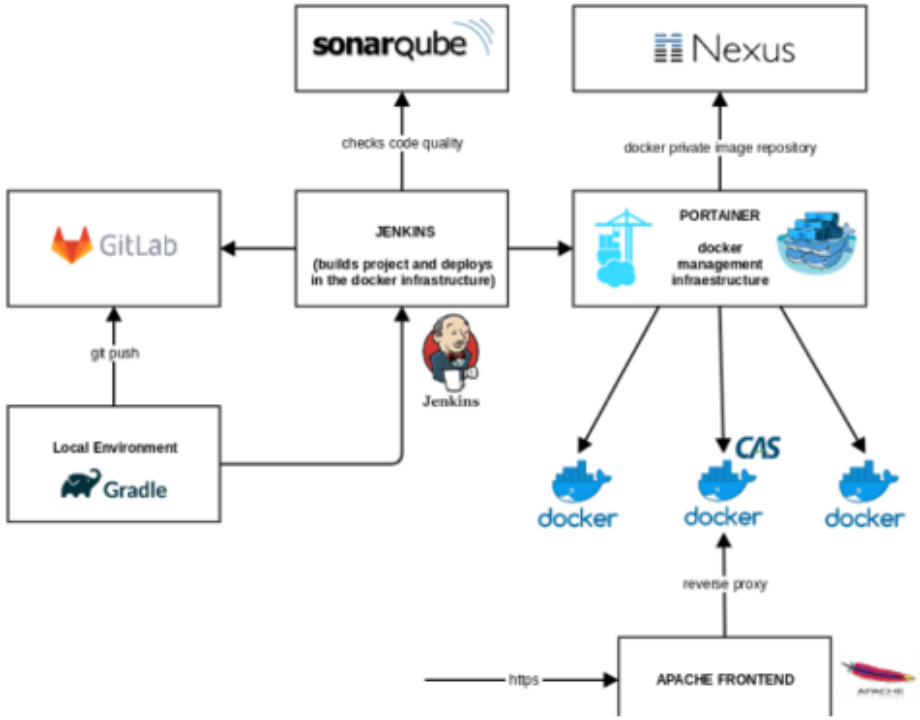
Benefit

Containers + Micro Services

Why are Containers good for Micro Services?

- Designed to run one application per container
  - Natural separation of work load
- Very lightweight
  - Great for scaling quickly
- Better use of resources
  - Containers share the host OS and where appropriate Binaries and Libraries
- Standard container formats such as Docker are cross linux distro compatible
- Incredible easy to move your work load around
  - Balance your system resources better
  - Allow developers to work in mock production environment
    - Removes the "it worked on my laptop" issue

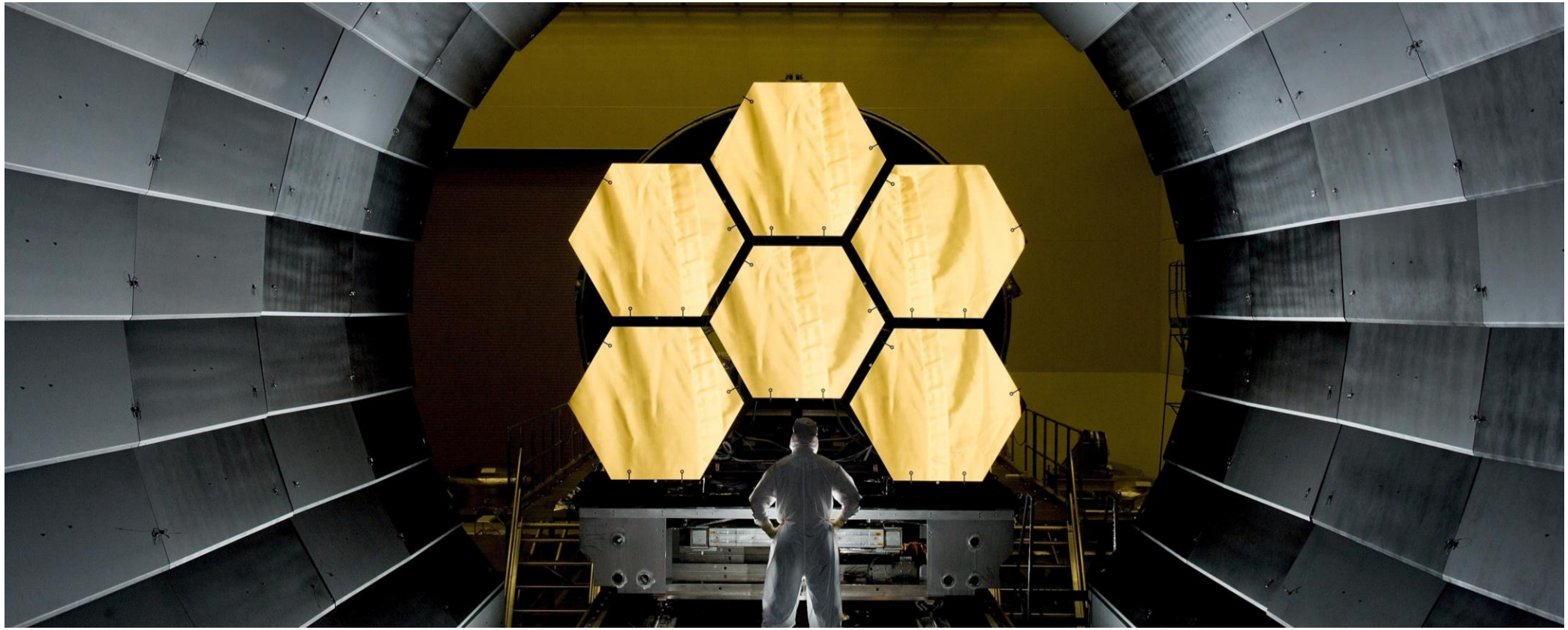
Should Apply CI/CD



Idea

- Large complex application
- Loosely Coupled
- Independent
- Easily Manage and deploy, for example case such as upgrade

https://haristauqir.com/introduction-to-microservices/



Cloud Technology (GCP)

Server for training

4 main platforms that can be prepared for this CKAN session, this presentation covers for only AWS EC2 and Google GCE

Please note that to register for Google Cloud, it's must have to register with your real credit card, for AWS, can use virtual card.

Just ensure that for a platform that you are using can access internet and open for SSH and port 5000 be able to connect.

Amazon AWS Register (Free 750 hours per month for 1 year with Linux on t2.micro – 1GB RAM):

Google GCP Register ($300 for first 12 months, always Free of f1.micro – 0.6 GB RAM): https://cloud.google.com/free

VMWare Player download: https://my.vmware.com/en/web/vmware/free#desktop_end_user_computing/vmware_workstation_player/15_0

Ubuntu 16.04 for VMWare: https://www.linuxvmimages.com/images/ubuntu-1604/

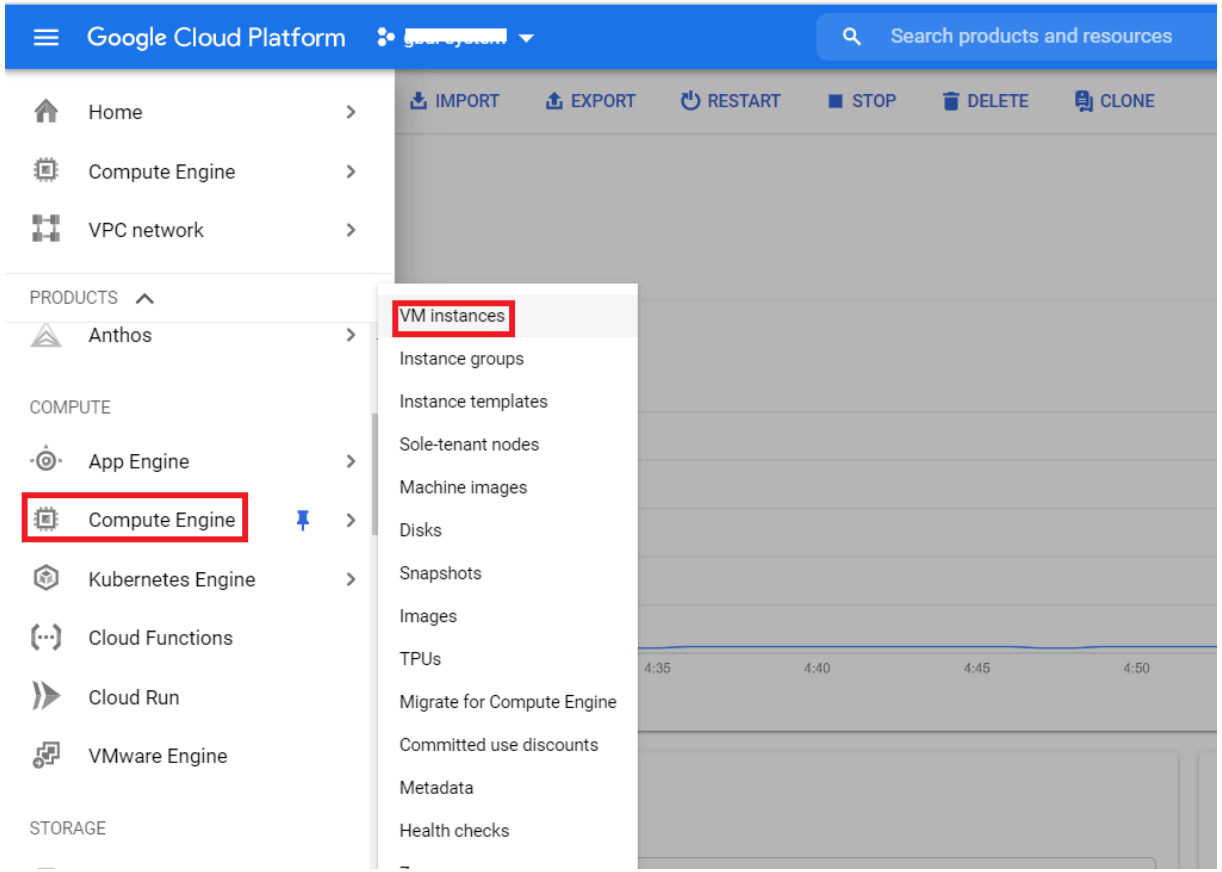
VirtualBox download: https://www.virtualbox.org/wiki/Downloads

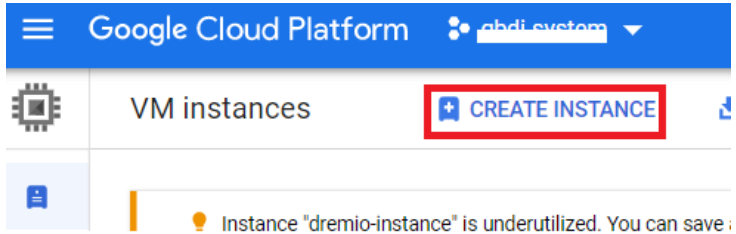
Ubuntu 16.04 for VirtualBox: https://www.linuxvmimages.com/images/ubuntu-1604/

Reference for Free Tier comparision: https://mytechdecisions.com/it-infrastructure/comparing-price-amazon-web-services-aws-google-cloud/

Workshop – Compute Engine (GCE)

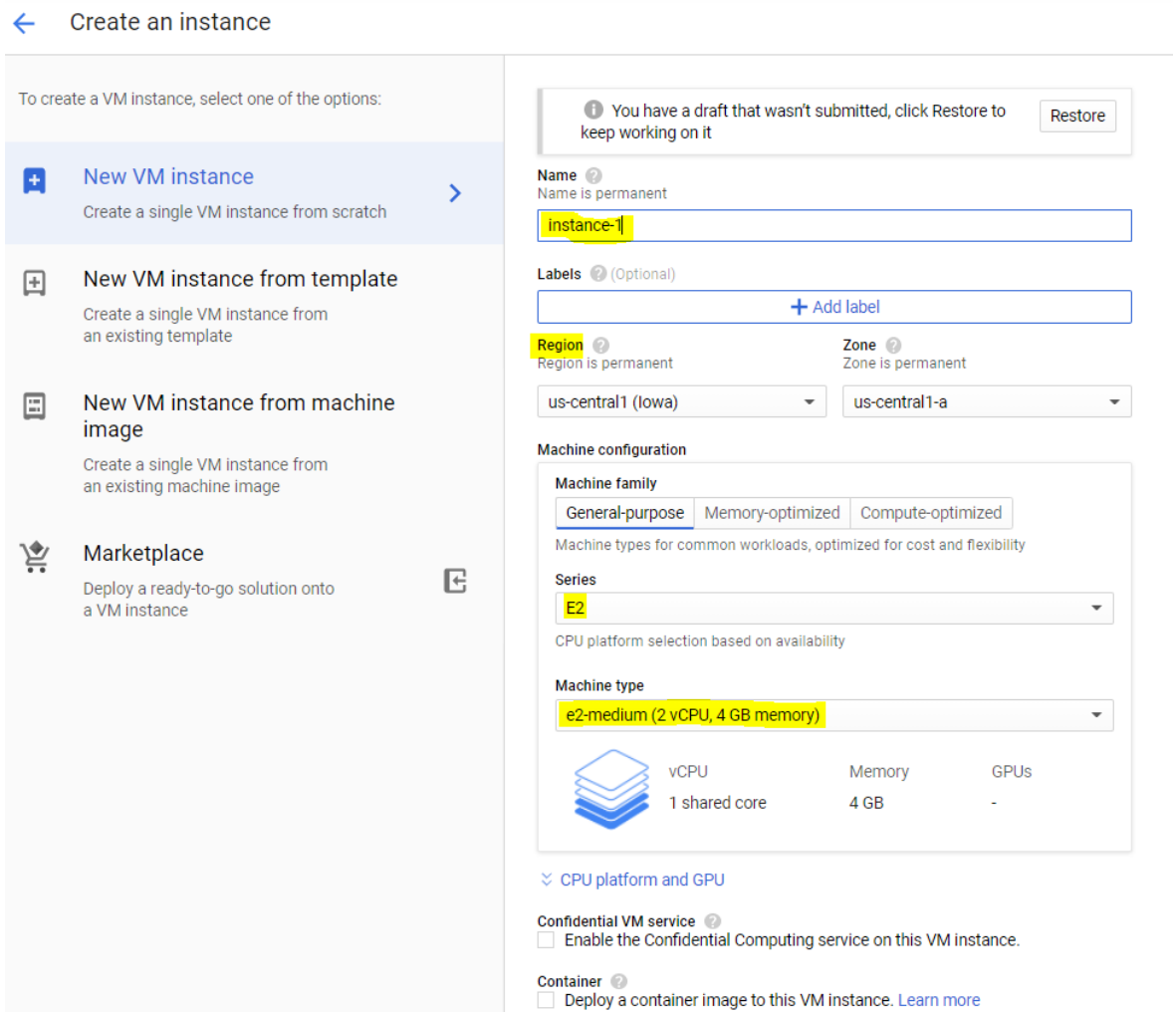
Open Compute Engine Screen and Click Create Instance

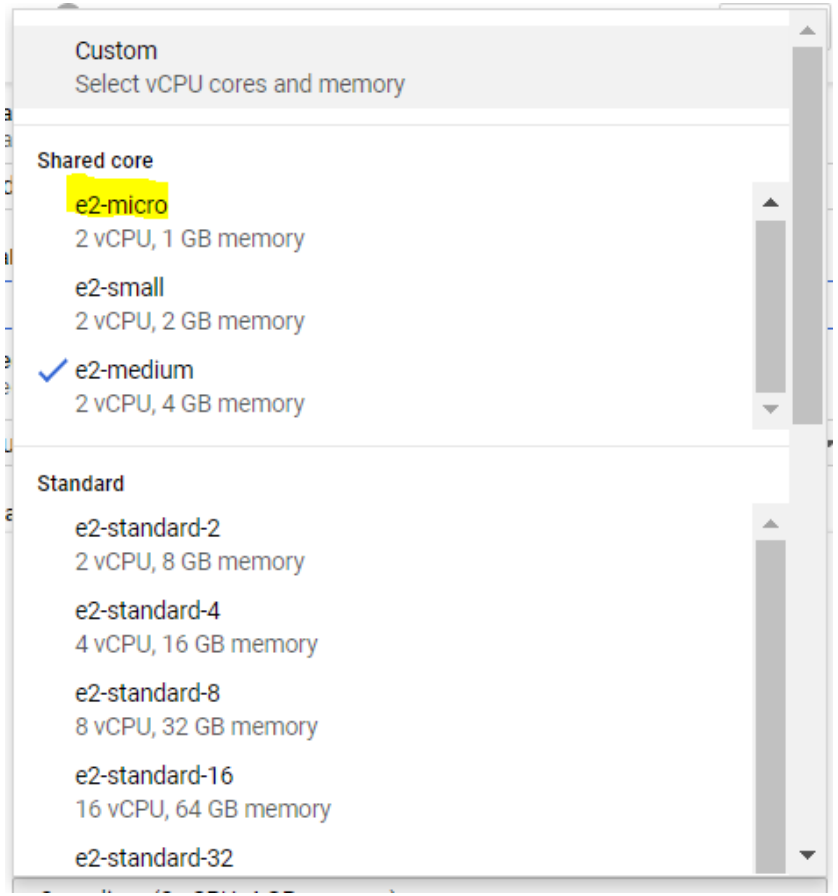




Workshop – Compute Engine (GCE)

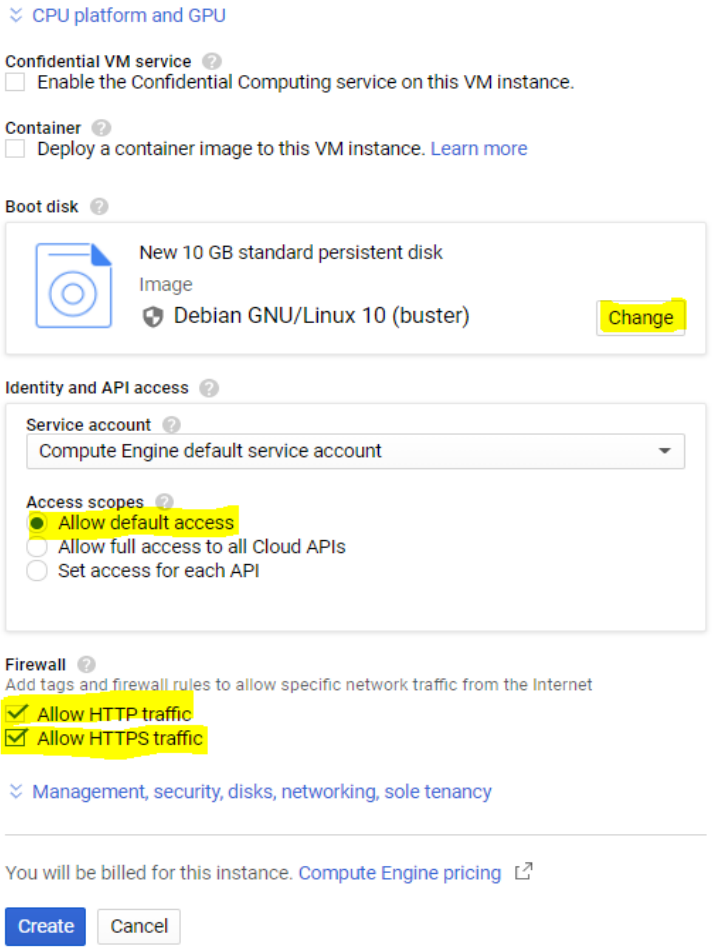
Rename , select region and specify machine type

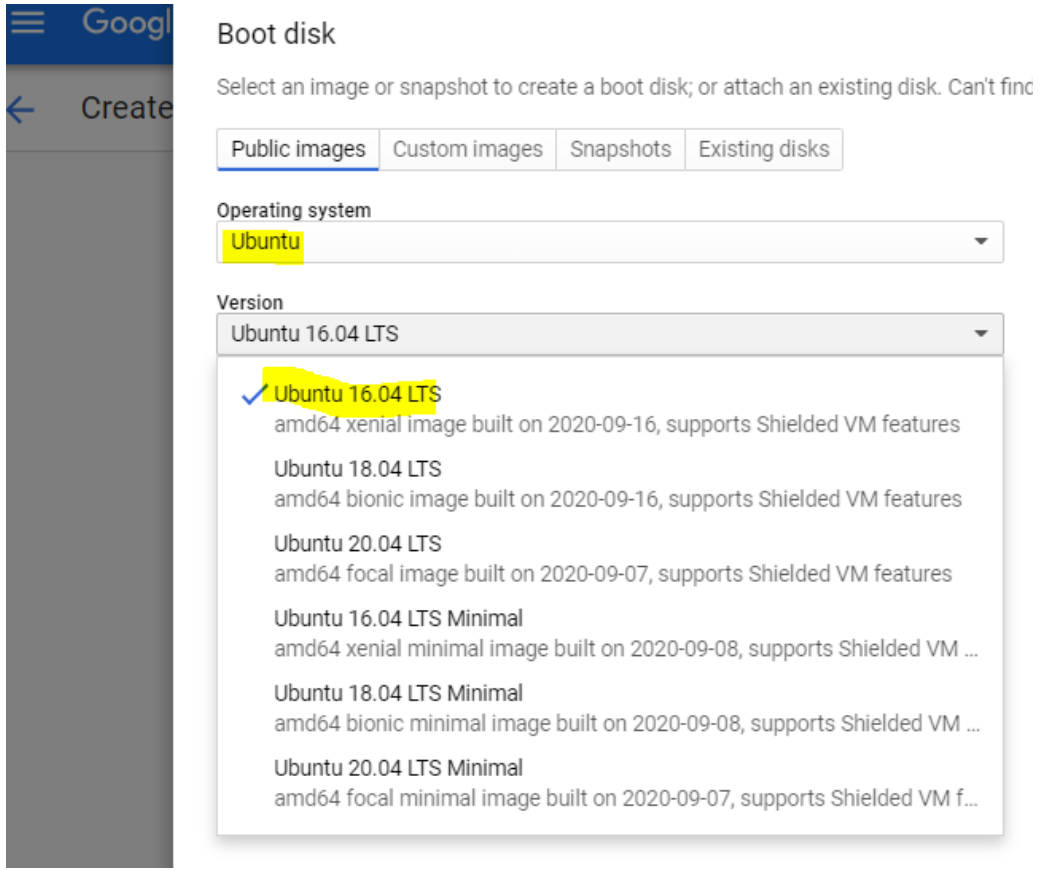




Workshop – Compute Engine (GCE)

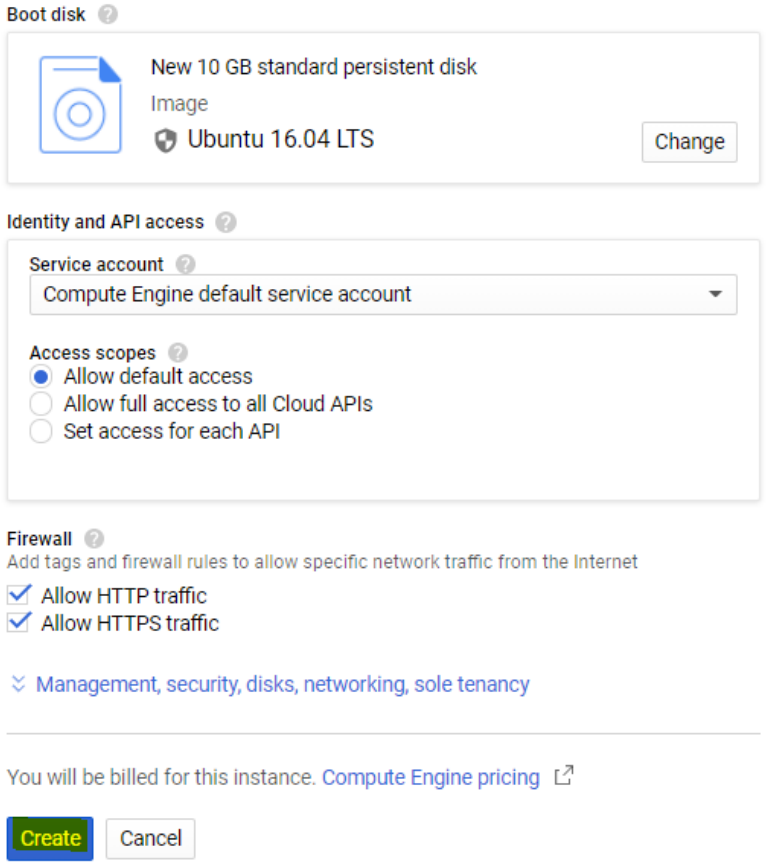
Select machine image (OS) and allow traffic HTTP and HTTPS

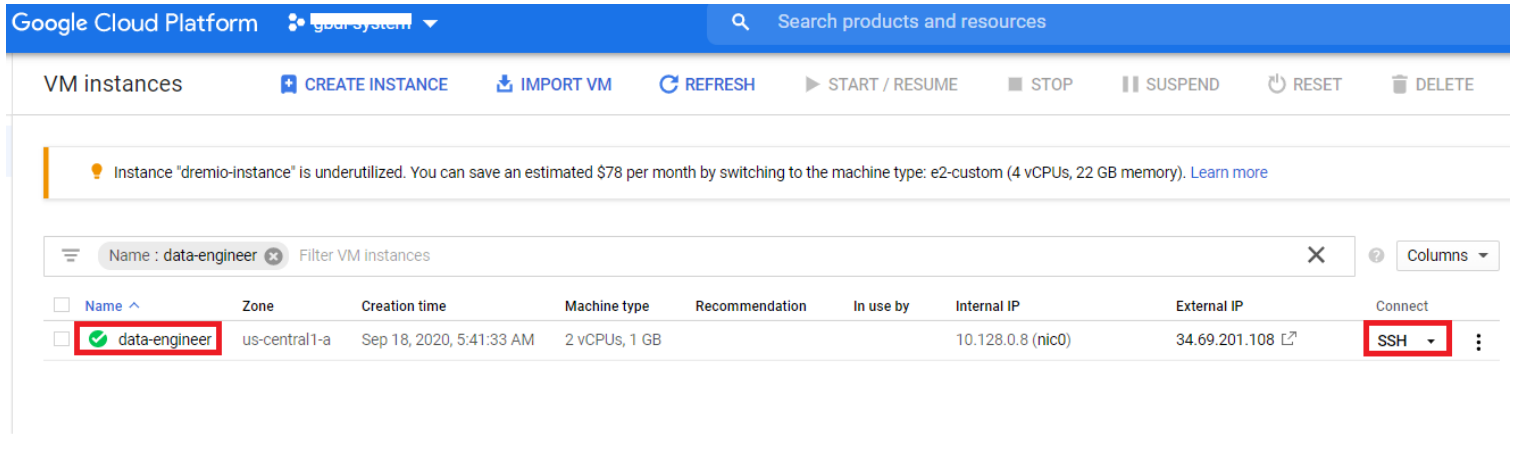


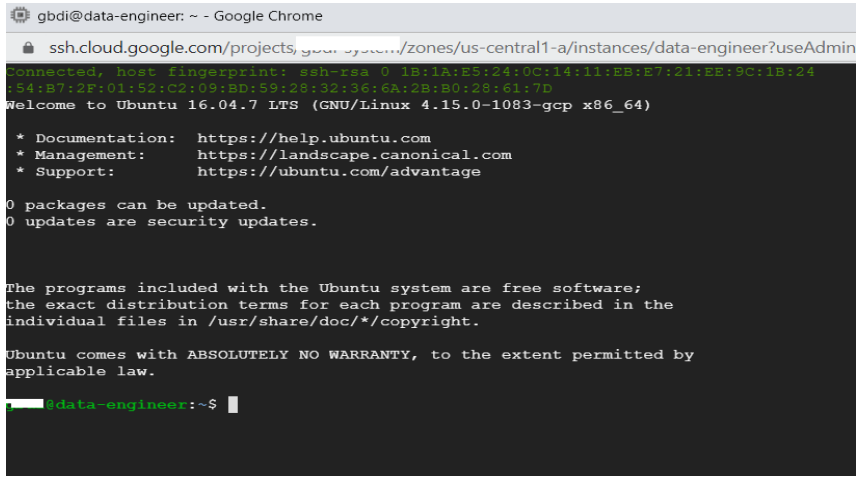


Workshop – Compute Engine (GCE)

Click Create and wait for the instance is ready with running status , click SSH to open Secure Shell

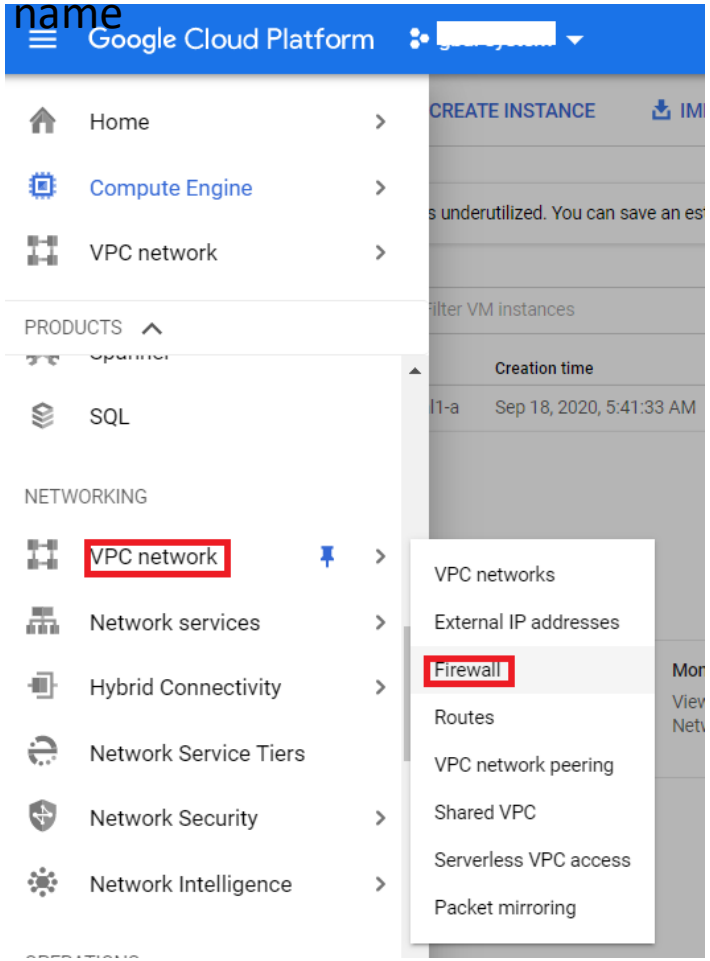




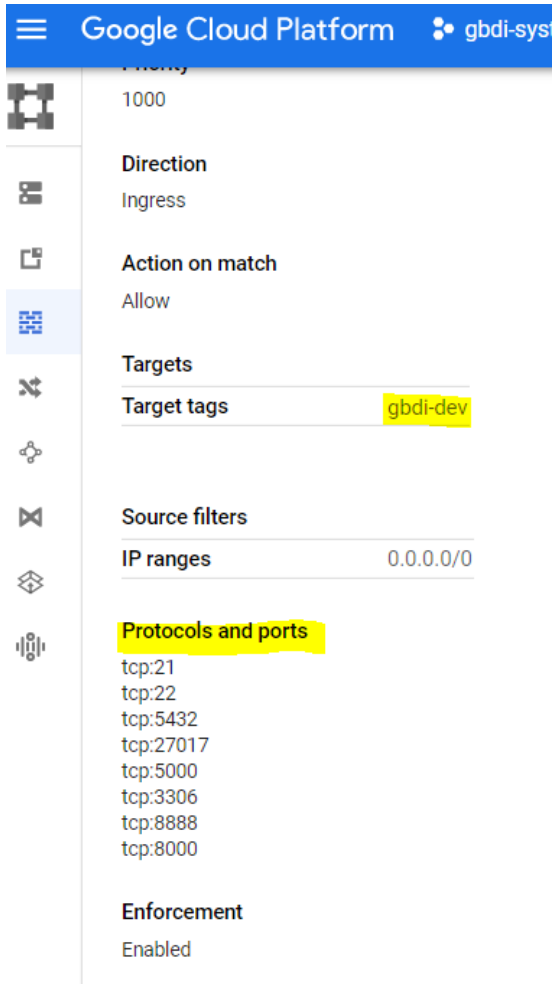


Workshop – Compute Engine (GCE)

Open VPC network for configure Firewall name

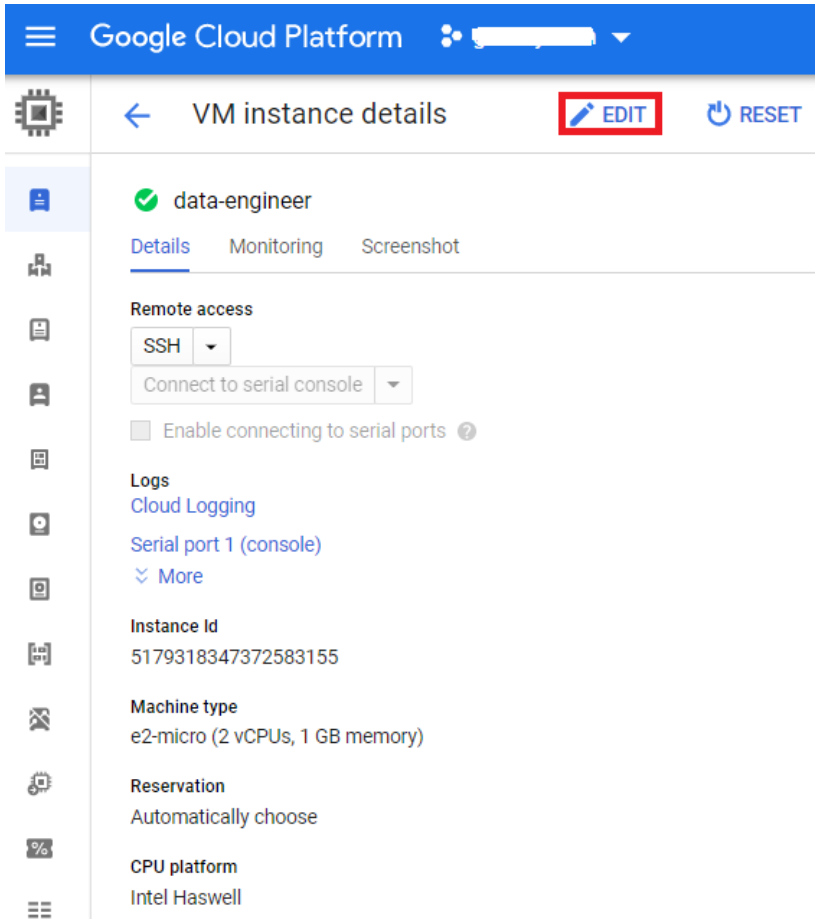


Cerate new rule for required port and give tags

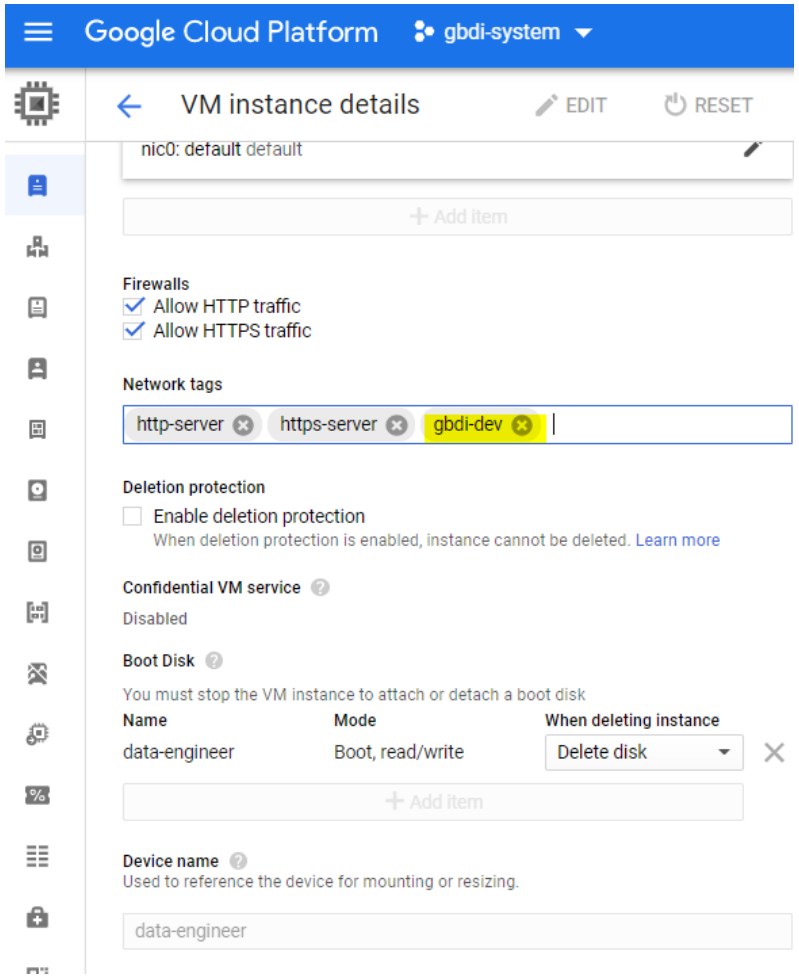


Workshop – Compute Engine (GCE)

Back to your instance and click EDIT

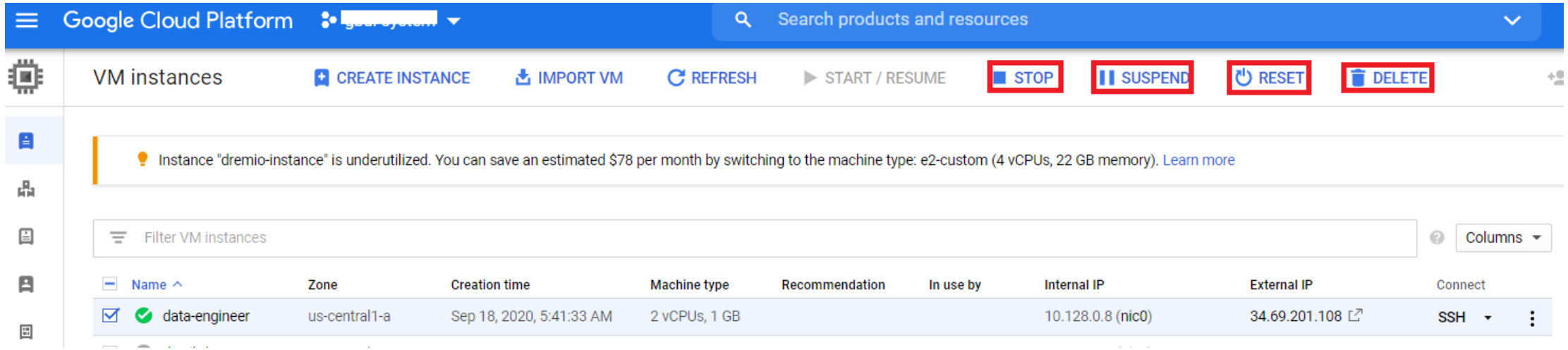


Add tag of the new firewall rule that just crated and save



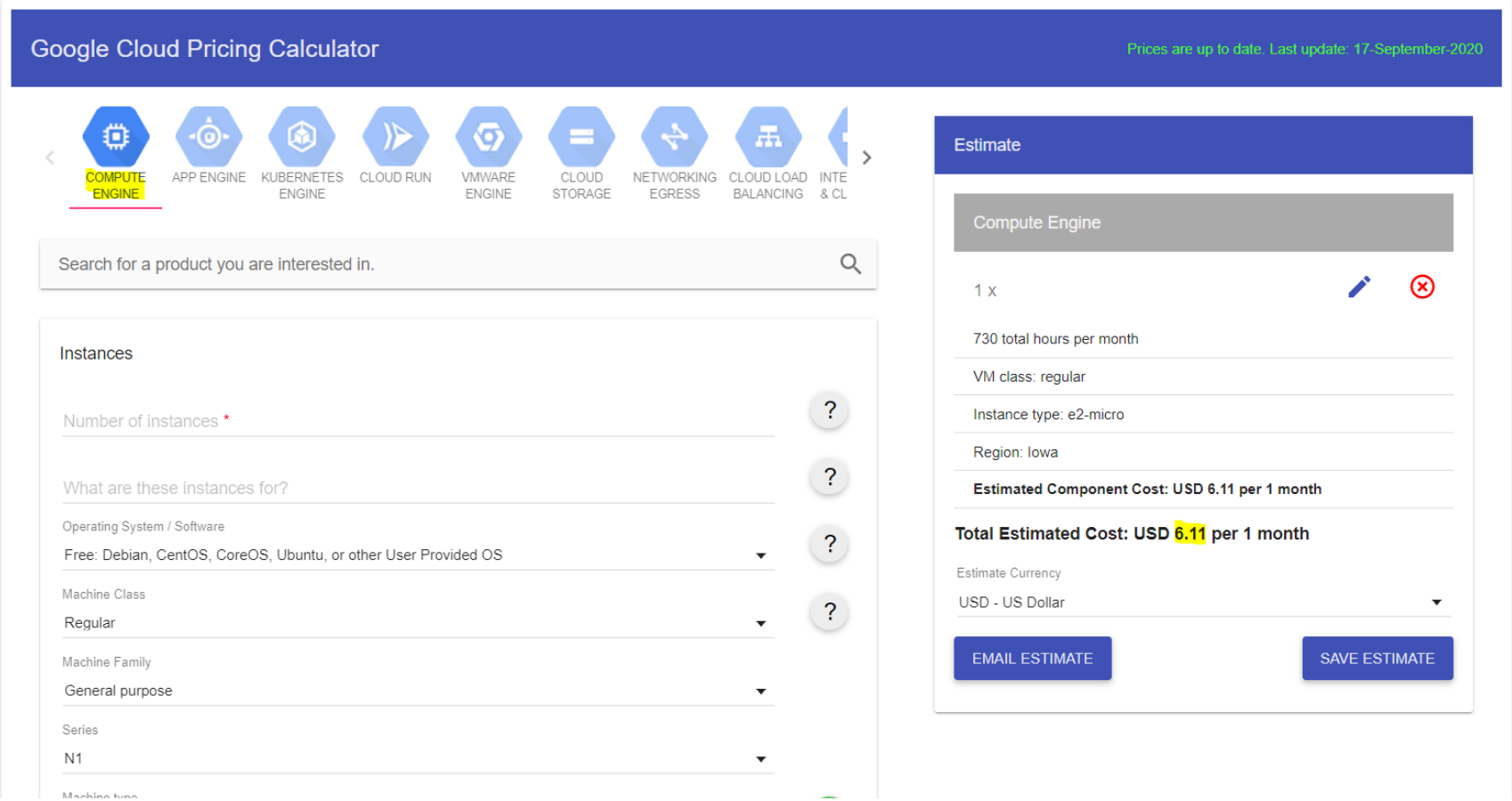
Workshop – Compute Engine (GCE)

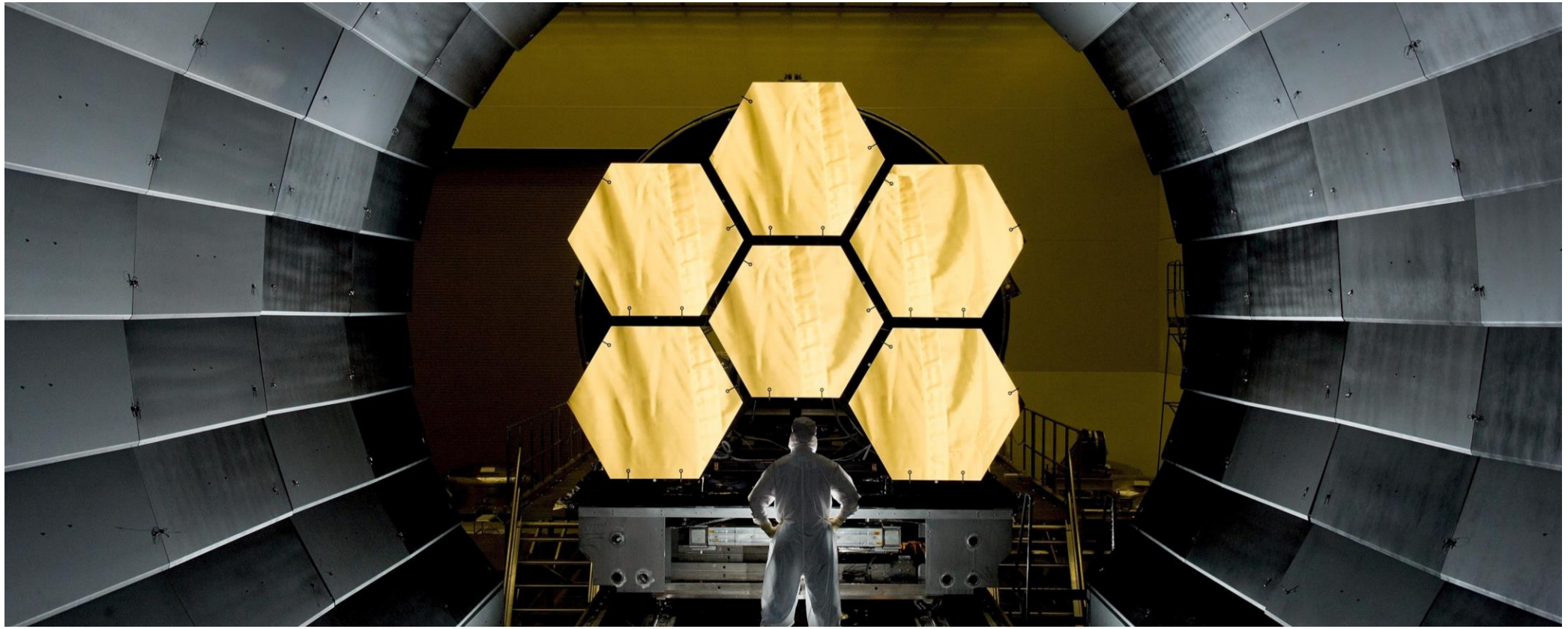
For STOP, SUSPEND, RESET and DELETE instance



Workshop – Compute Engine (GCE)

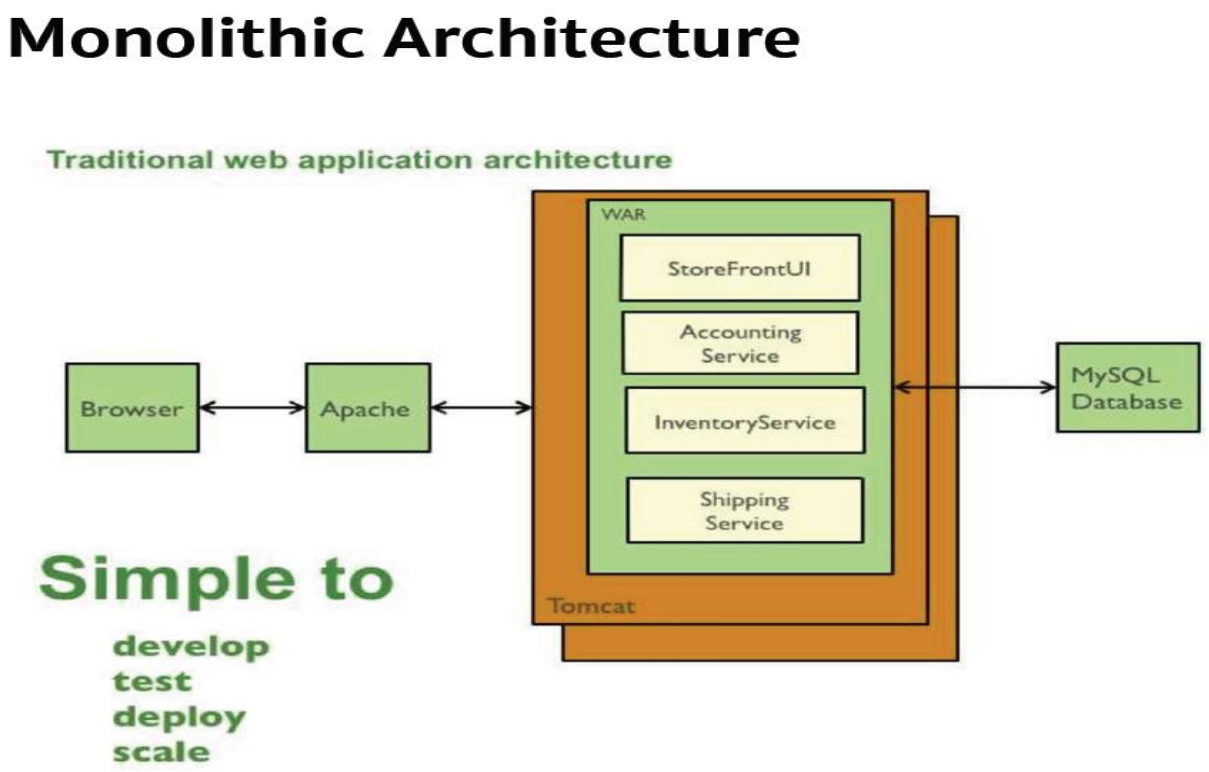
How to calculate price : https://cloud.google.com/products/calculator





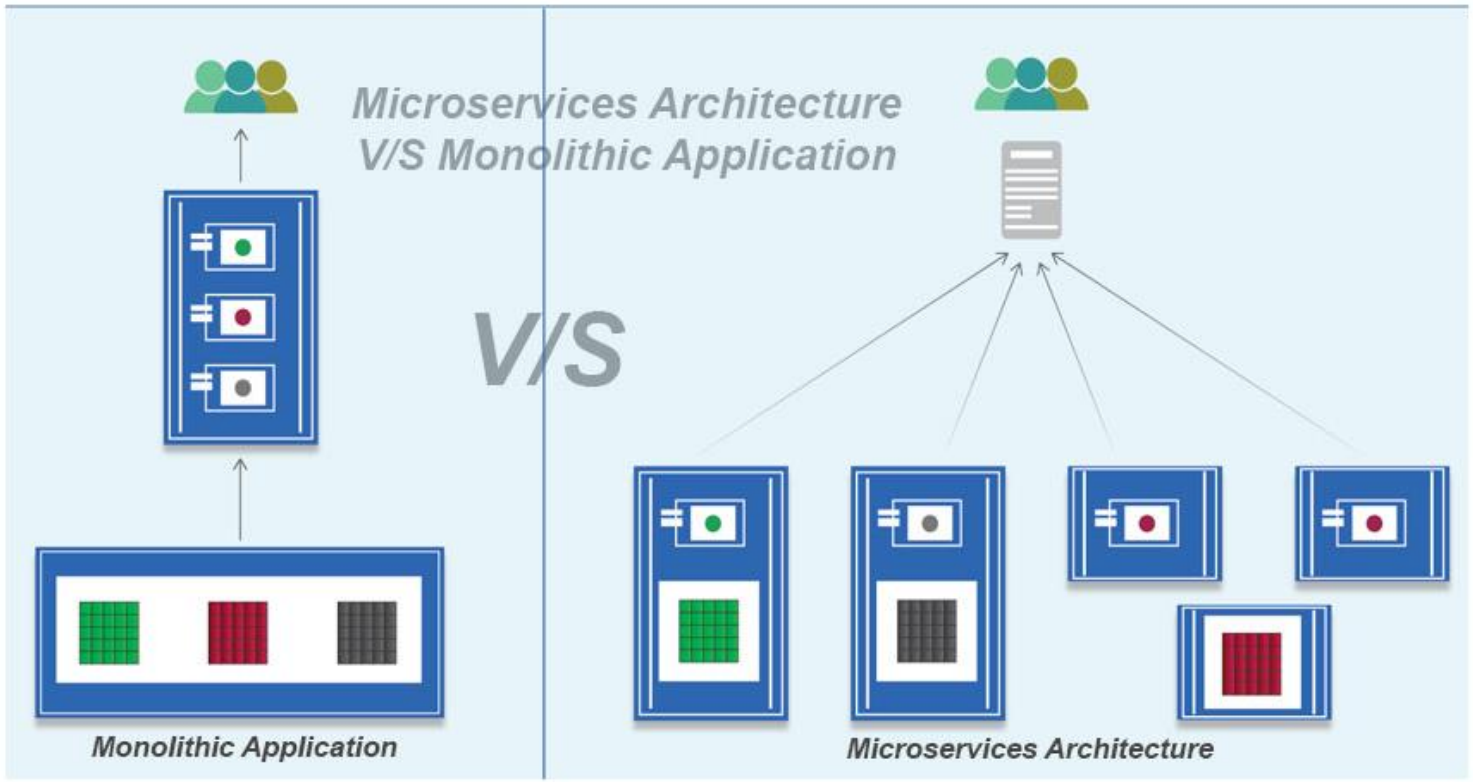
Microservices Design

Monolithic Application



https://haristauqir.com/introduction-to-microservices/

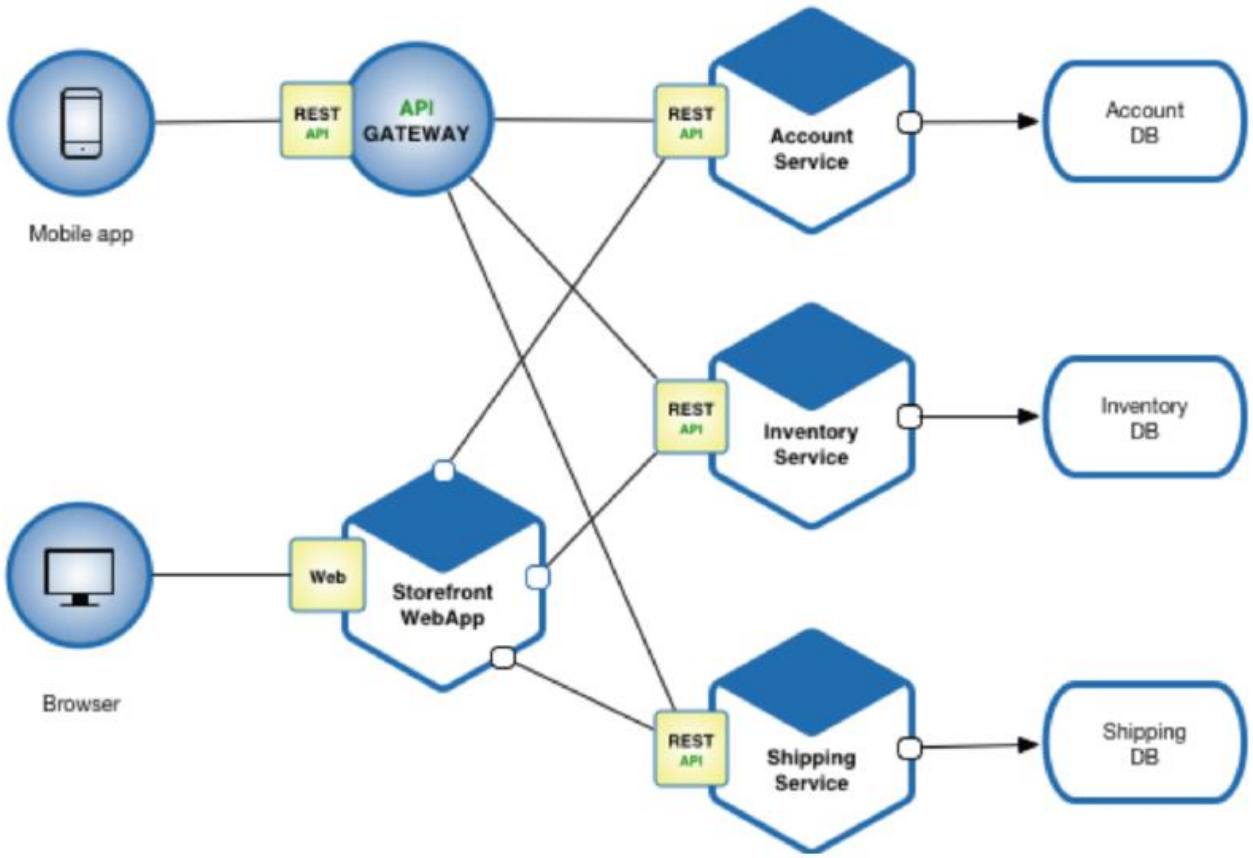
Transition From Monolith to Microservices



Service ง่ายต่อการทำความเข้าใจ
Service ง่ายต่อการพัฒนา
Service ง่ายต่อการทดสอบ
Service ง่ายต่อการ deploy
Service ง่ายต่อการ scale

 https://solutiondots.com/blog/technology-blog/microservices-architecture-vs-monolithic-application/

Microservice Architecture



Why Microservices ?

1. Code can be broken out into smaller microservices that are easier to learn, release and update.
2. Individual microservices can be written using the best tools for the job.
3. Releasing a new service doesn't require synchronization across a whole company.
4. New technology stacks have lower risk since the service is relatively small.
5. Developers can run containers locally, rebuilding and verifying after each commit on a system that mirrors production.
6. Both Docker and Kubernetes are open source and free to use.
7. Access to Docker hub leverages the work of the opensource community.
8. Service isolation without the heavyweight VM. Adding a service to a server does not affect other services on the server.
9. Services can be more easily run on a large cluster of nodes making it more reliable.
10. Some clients will only host in private and not on public clouds.

- There is a cost for running microservices - the build and runtime becomes more complex. This is part of the price to pay and if you've made the right decision in your context, then benefits will exceed the costs.

Trade off

Monitoring, logging, tracing, infrastructure, testing,deployment, etc.

Team topologies

Team cognitive load

Complexity

Organizational culture

Decomposition application

- Decompose by business capability

  Mapping from business boundaries, User journey

- Decompose by subdomain

  Use Domain driven design

- Self-contained service

  Design a service that can respond to synchronous request

  Collaborate with other using CQRS

- Service per team

  Each service owned by a team, sole responsibility for making changes

Microservice Thai Guru

คำถามต่อมาคือ ขนาดของ service จะมีขนาดเล็ก มันต้องเล็กเพียงใด ?

จากข้างต้นบอกว่า แต่ละ service ต้องมีทีมที่ดูแล เรียกว่า Cross-functional team
ดังนั้นขนาดของ service จะใหญ่เพียงใดนั้น
ตอบได้ง่าย ๆ คือ ทีมนั้น ๆ สามารถดูแล service ได้หรือไม่ ? (You build it, You run it)
ระบบงานที่เราสร้างมานั้น ไม่ได้เน้นไปที่จำนวน feature ให้ใช้งาน
แต่เน้นไปที่คุณค่าของระบบงานที่ให้ทางผู้ใช้งานและ business
รวมทั้งขนาดของ service ที่เล็ก จะยิ่งช่วยทำให้
ทีมพัฒนาและผู้ใช้งาน รวมทั้ง business ใกล้ชิดกันมากขึ้น
ซึ่งมันส่งผลดีต่อทุกฝ่าย

ต้อง Balance ทีมและการแยก Microservices ดีๆ
จะจัดการการตรวจสอบปัญหาได้ยากเมื่อข้ามระบบฐานข้อมูลได้

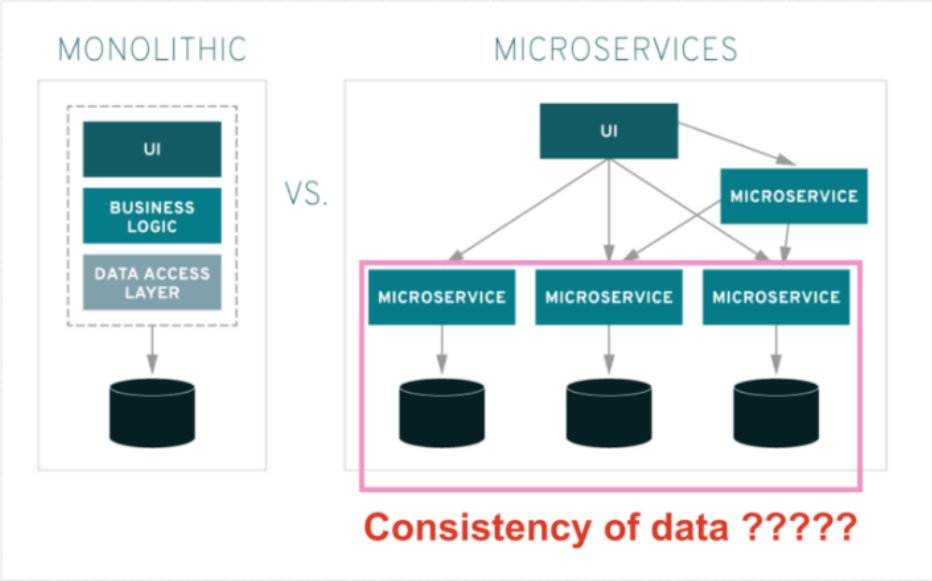
https://www.somkiat.cc/microservices-journey/

เนื่องจากแต่ละ service นั้นต้องมี data store หรือที่จะเก็บข้อมูลเป็นของตัวเอง

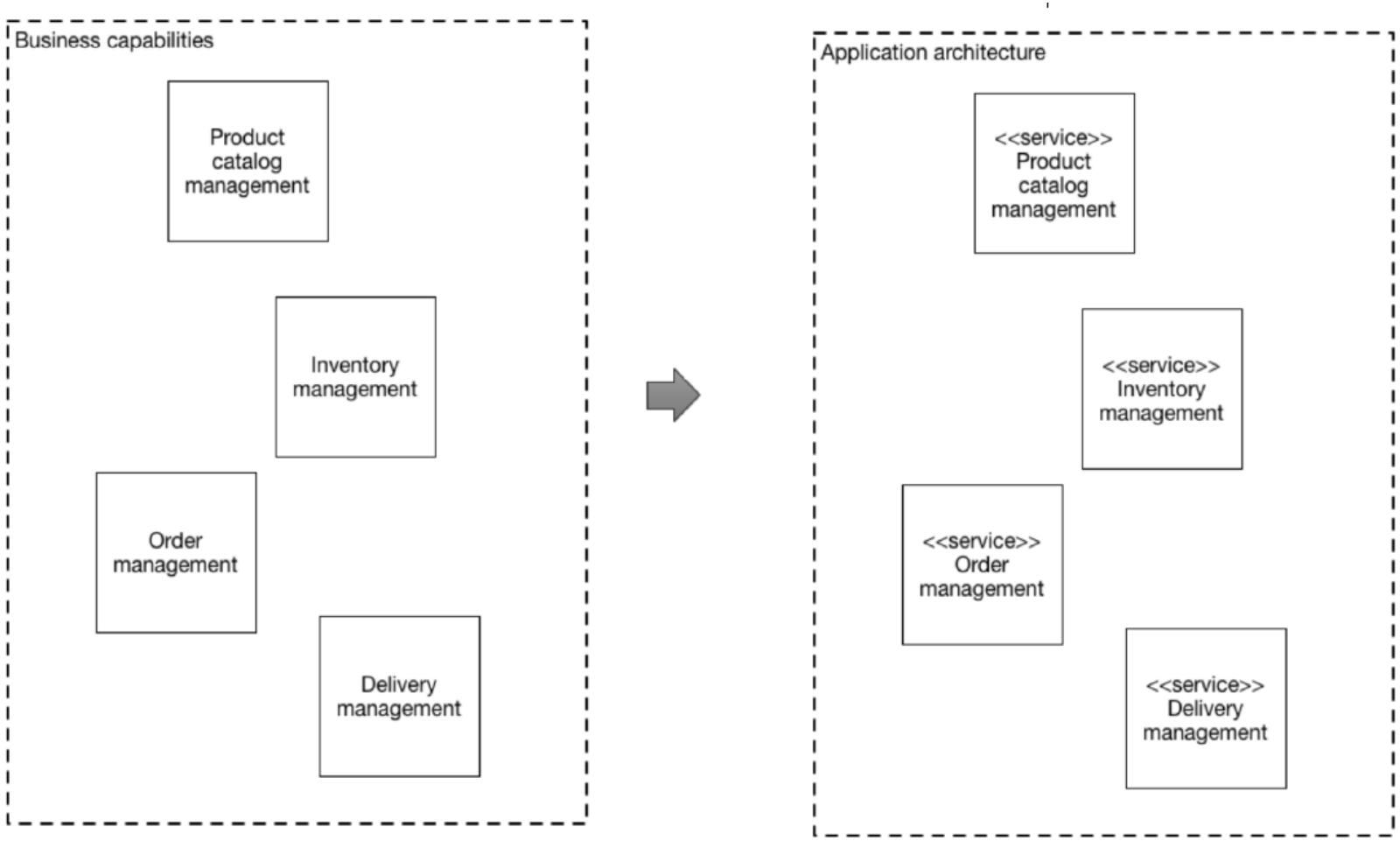
มันทำให้เกิดคำถามหนึ่งขึ้นมาคือ
เราจัดการความถูกต้องของข้อมูลกันอย่างไร
ถ้าข้อมูลหล่านั้นถูกใช้งานข้าม service ?

ปล. ปัญหานี้จะไม่เกิดในระบบ Monolithic อย่างแน่นอน

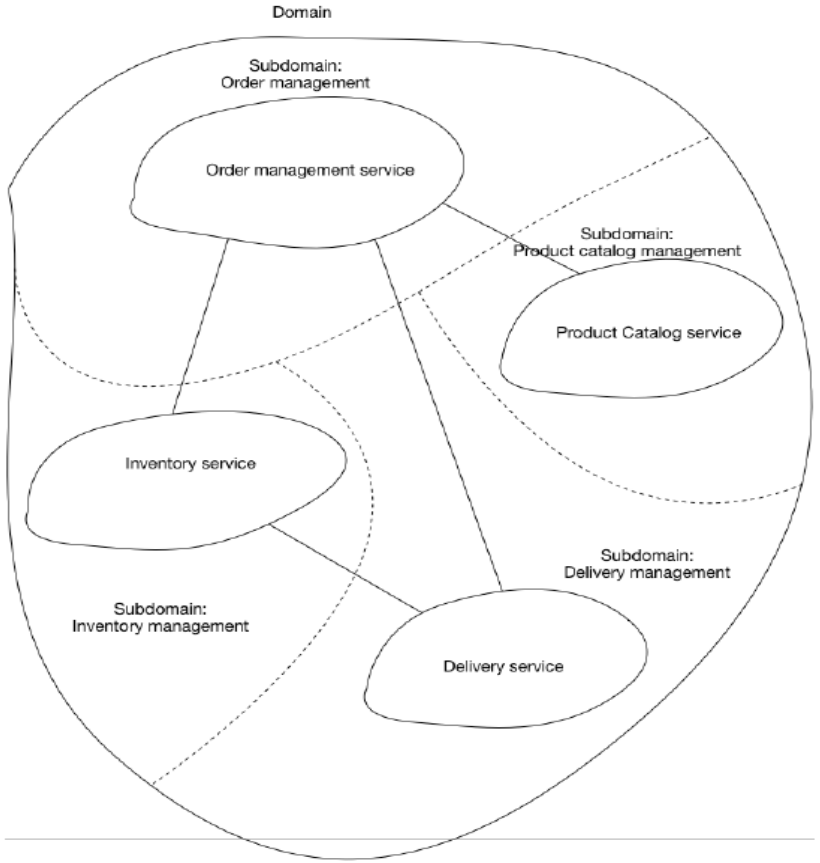
แสดงดังรูป



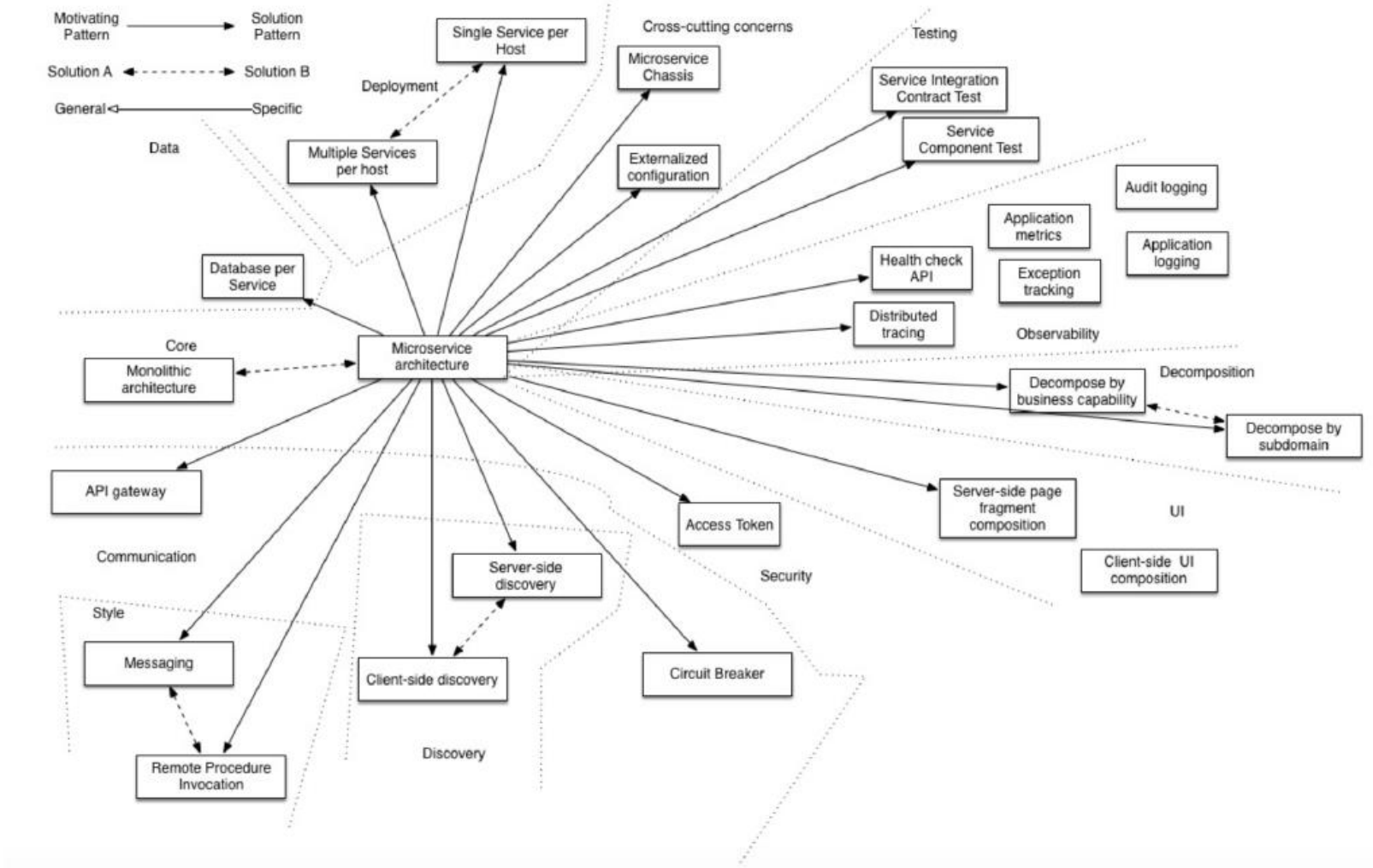
Decompose by business capability

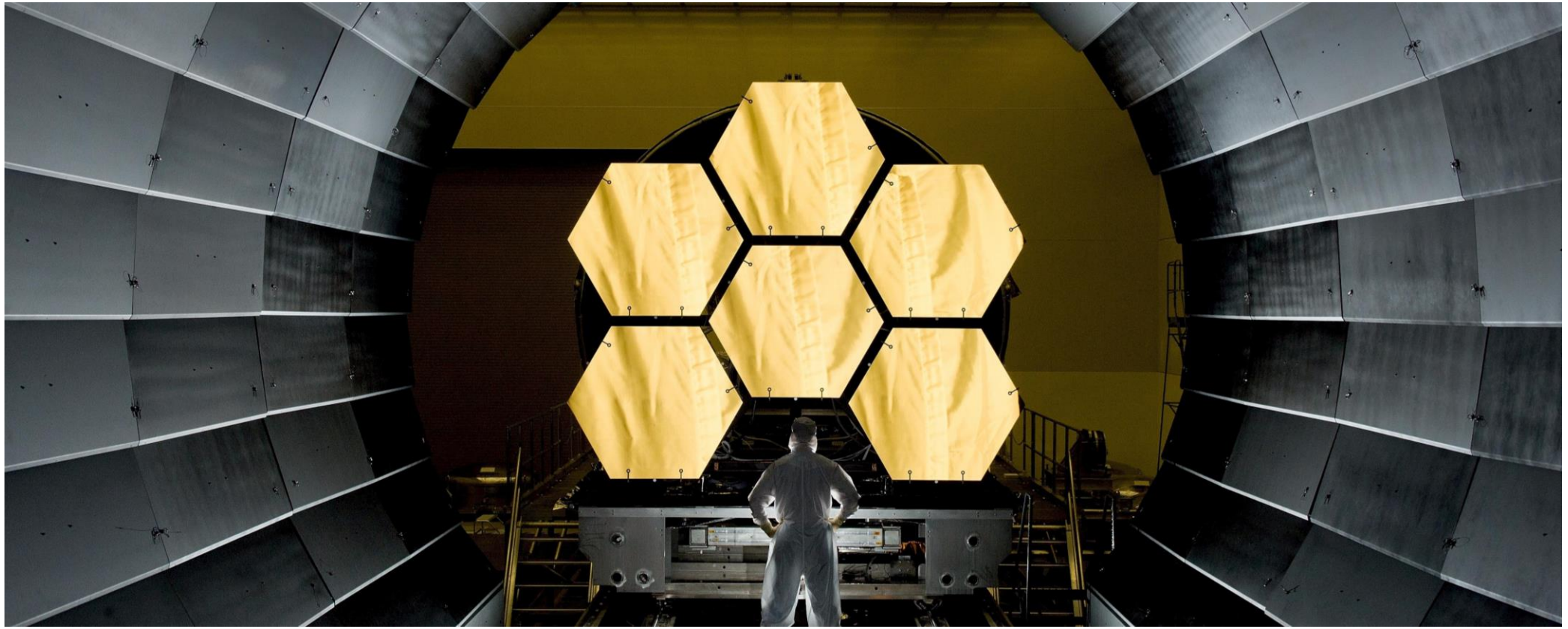


Decompose by domain-driven design subdomain



Related patterns





Restful API

Request / Respones

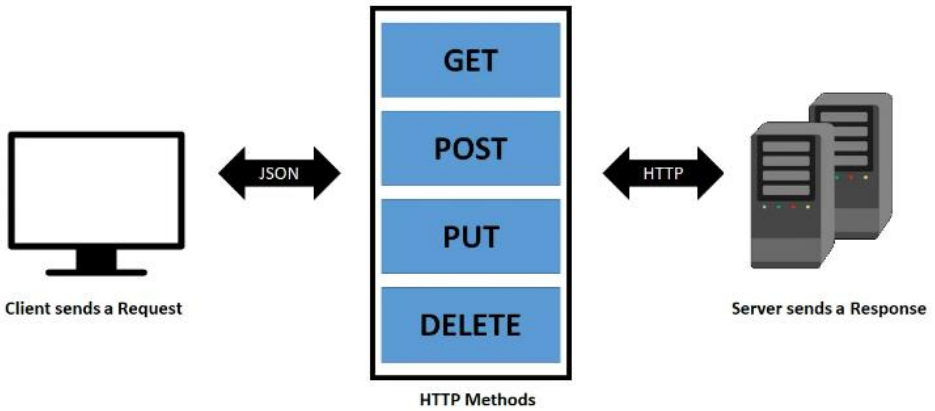
https://reqres.in/

GET	LIST USERS
GET	SINGLE USER
GET	SINGLE USER NOT FOUND
GET	LIST <RESOURCE>
GET	SINGLE <RESOURCE>
GET	SINGLE <RESOURCE> NOT FOUND
POST	CREATE
PUT	UPDATE
PATCH	UPDATE
DELETE	DELETE

Request
/api/users?page=2

Response
200

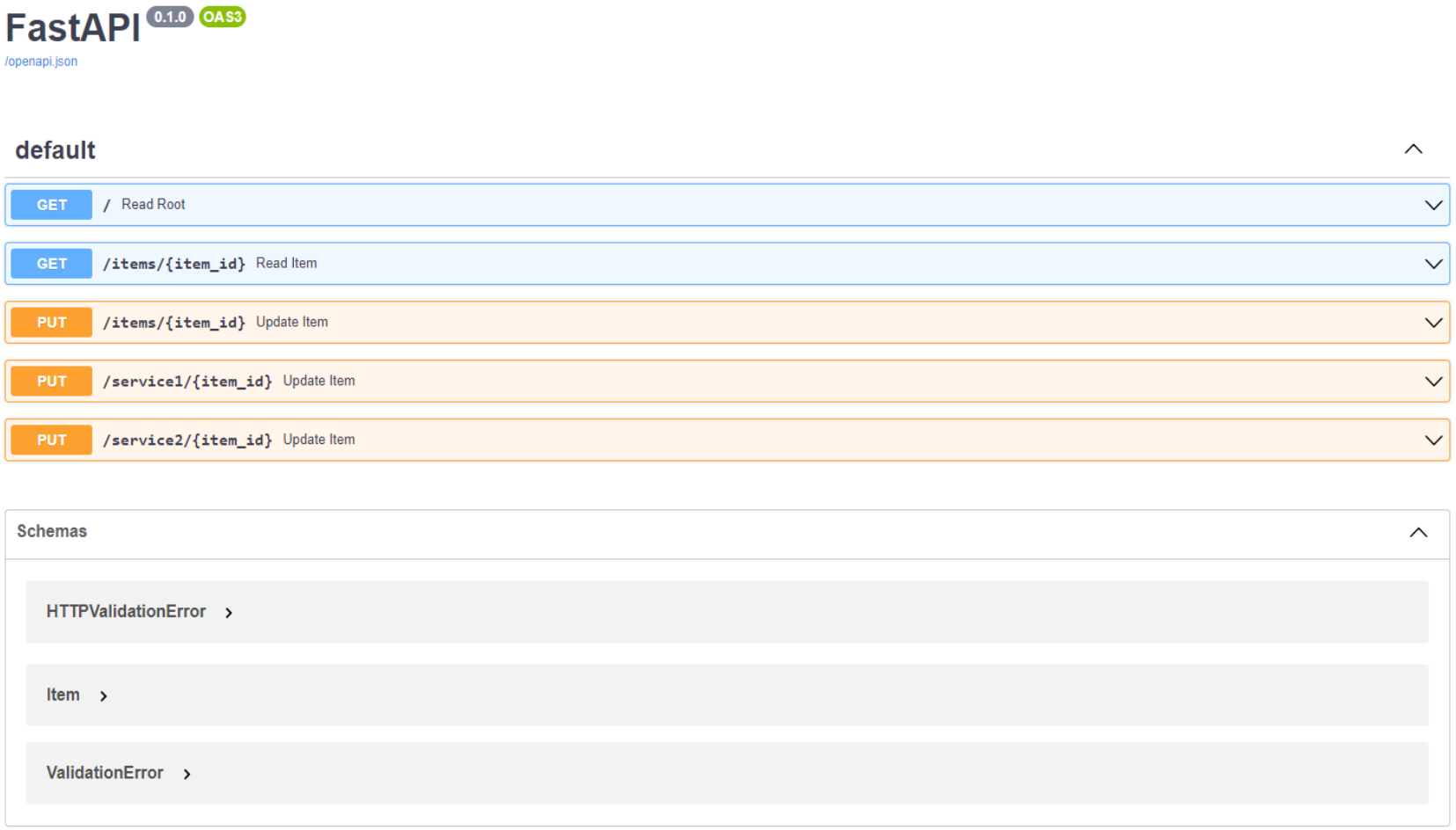

```
{
    "page": 2,
    "per_page": 6,
    "total": 12,
    "total_pages": 2,
    "data": [
        {
            "id": 7,
            "email": "michael.lawson@reqres.
            "first_name": "Michael",
            "last_name": "Lawson",
            "avatar": "https://reqres.in/img
        },
        {
            "id": 8,
            "email": "lindsay.ferguson@reqre
            "first_name": "Lindsay",
            "last_name": "Ferguson",
            "avatar": "https://reqres.in/img
        },
        {
            "id": 9,
            "email": "tobias.funke@reqres.in
            "first_name": "Tobias",
            "last_name": "Funke",
            "avatar": "https://reqres.in/img
        },
```

What is REST

As stated earlier, REST stands for Representational State Transfer. It is a simple way of sending and receiving data between client and server. It doesn't require any software or standards to transfer data. It has a predefined structure to do the API call. Developers just need to use the predefined way and pass their data as JSON payload.

https://gocoding.org/what-are-restful-web-services/

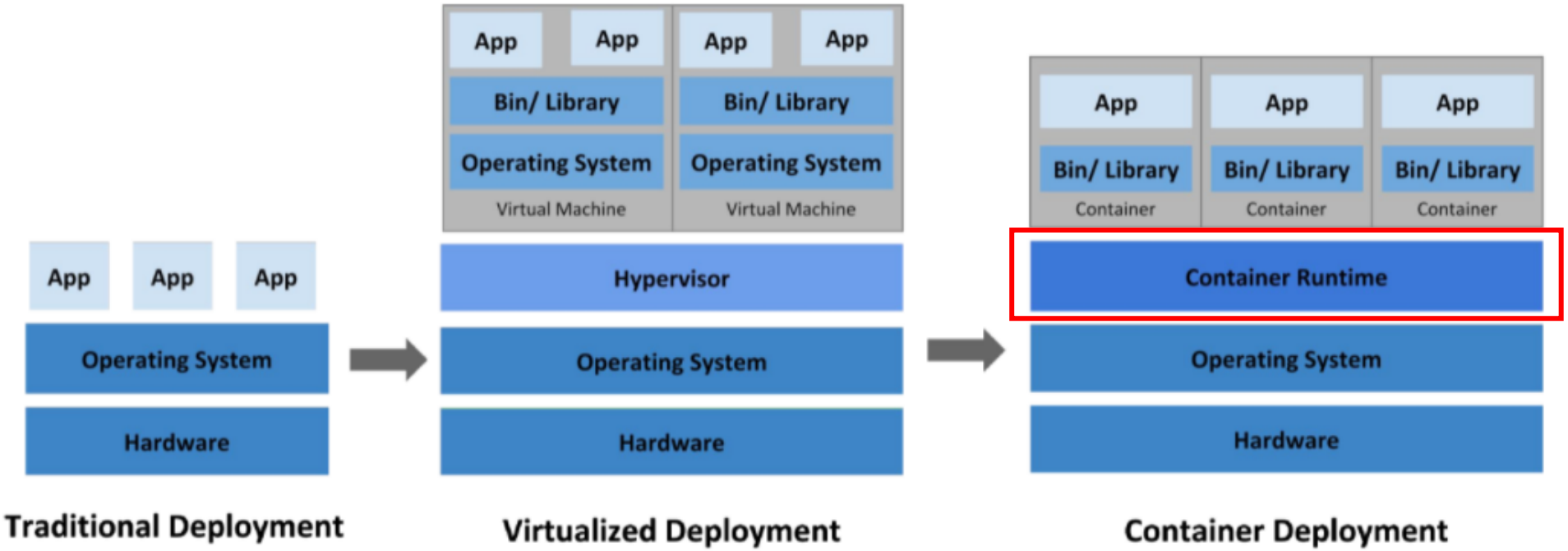

FastAPI
0.1.0
OAS3
/openapi.json
default
GET / Read Root
GET /items/{item_id} Read Item
PUT /items/{item_id} Update Item
PUT /service1/{item_id} Update Item
PUT /service2/{item_id} Update Item
Schemas
HTTPValidationError
Item
ValidationError

Modern Infrastructure

Docker *“Software Container”*

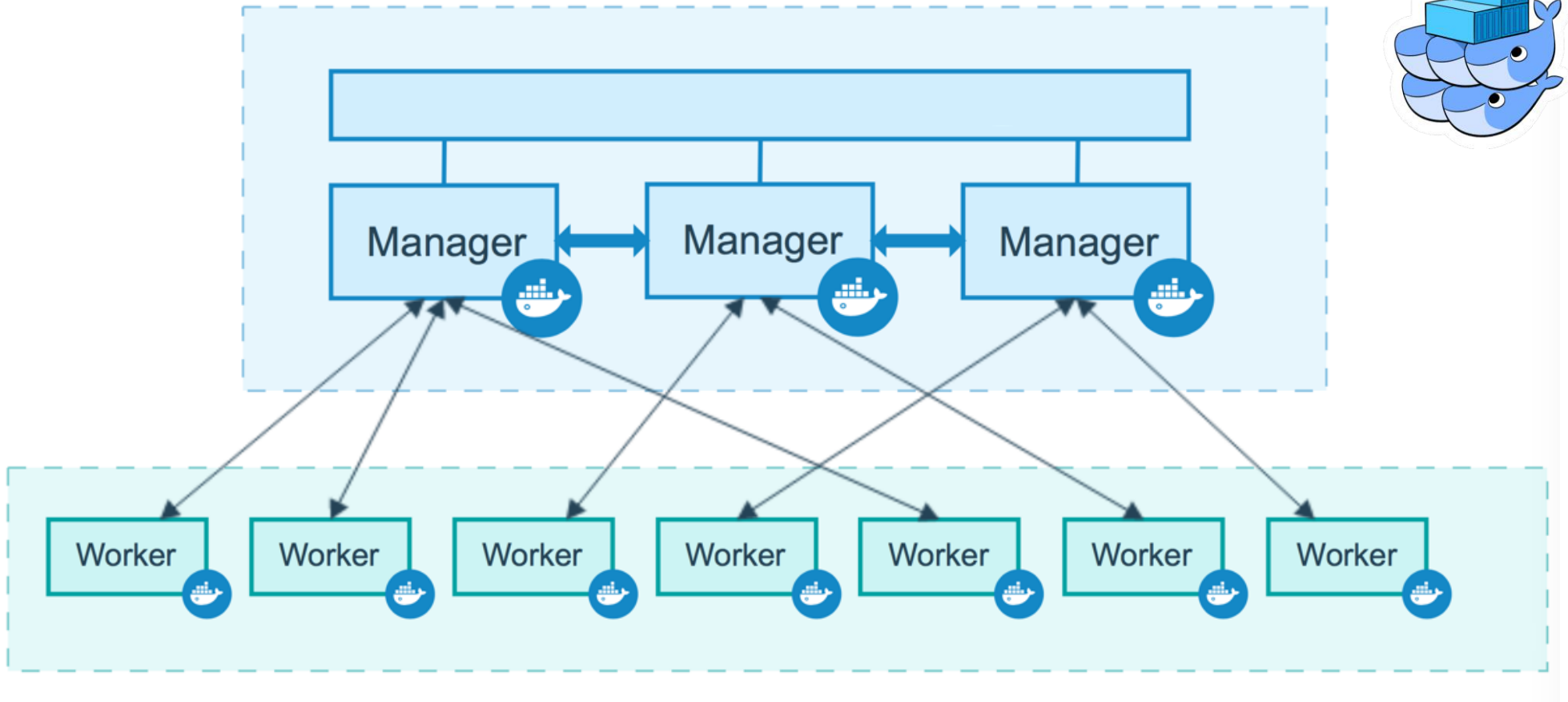
ช่วยสร้าง ทดสอบ และติดตั้งแอปพลิเคชัน ได้อย่างรวดเร็ว

“ Build - Ship - Run ”

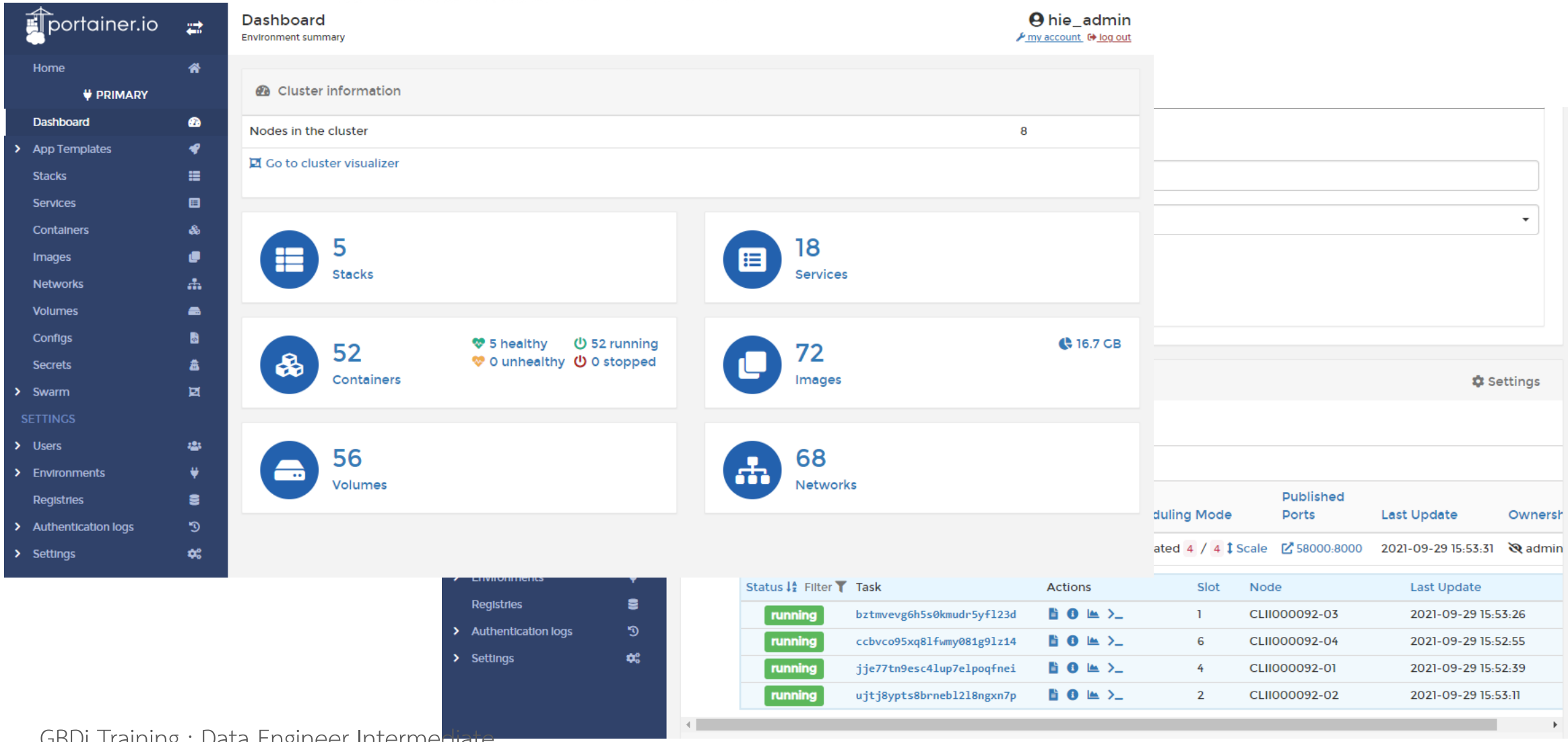


Docker Swarm

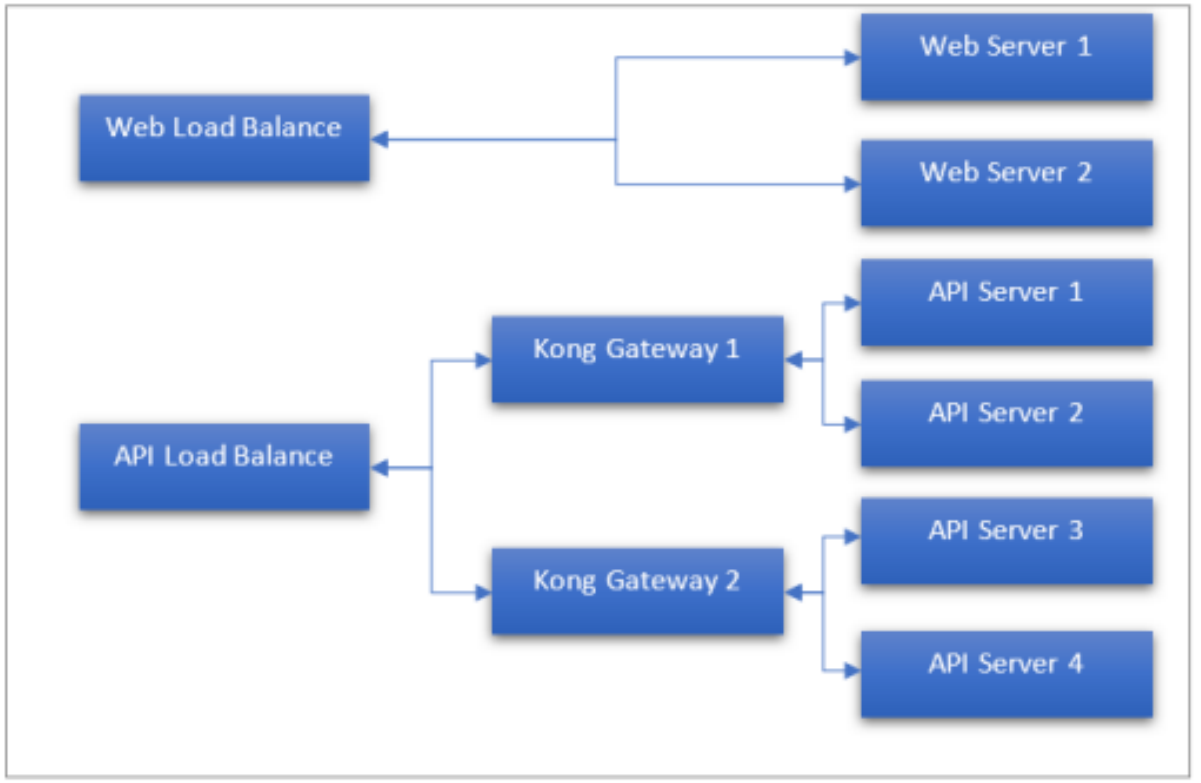
: Container-Orchestration



portainer.io

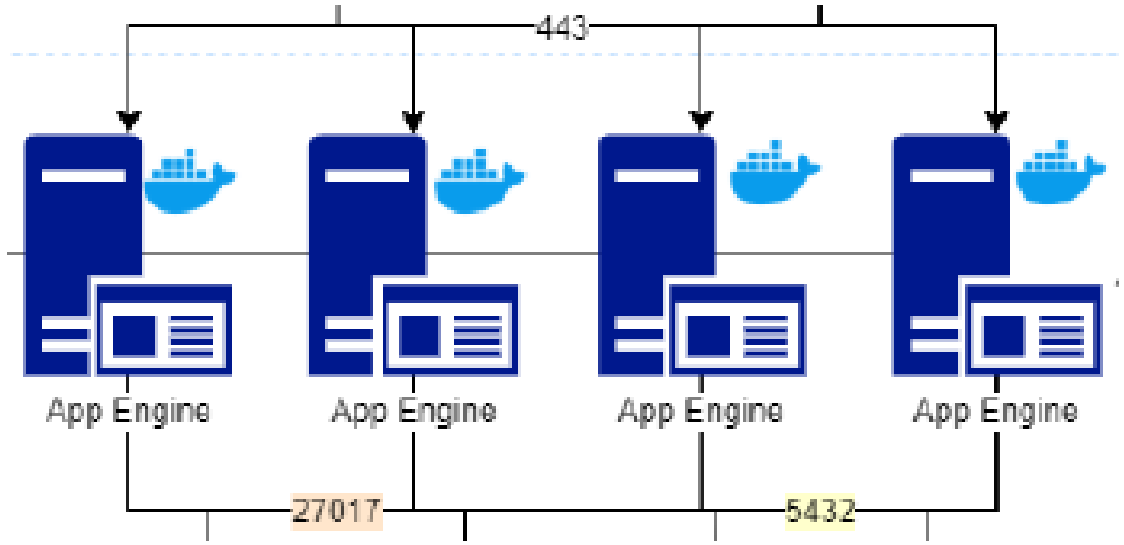


Loadbalance Architecture

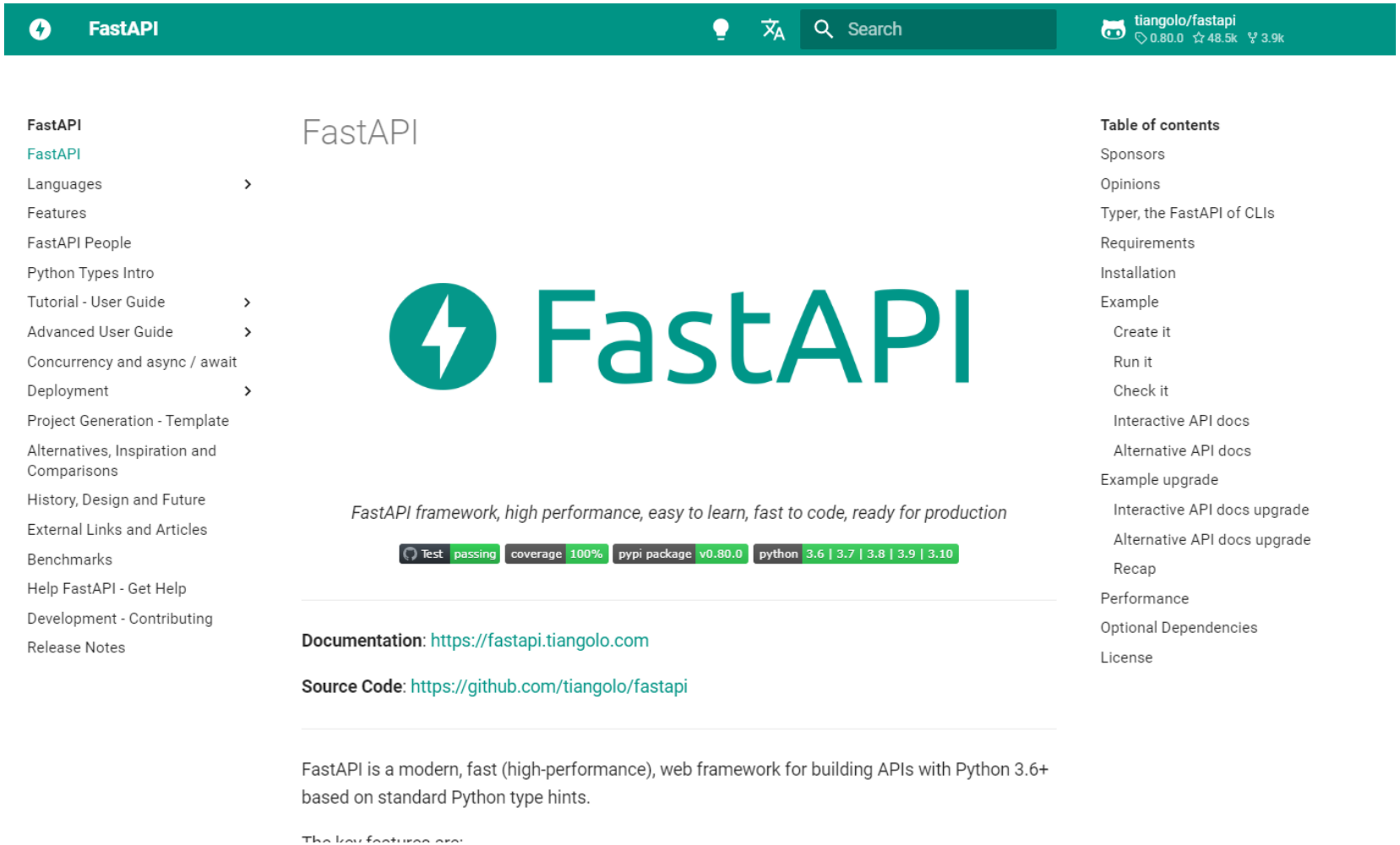


Deploy Backend Service

FastAPI



FastAPI Lab https://fastapi.tiangolo.com/



BIG DATA
THAILAND

Government
Big Data
Institute
สถาบันส่งเสริมการวิเคราะห์และบริหารข้อมูลขนาดใหญ่ภาครัฐ

Thank You