# Cloud Native Final Project
# Smart Manufacture

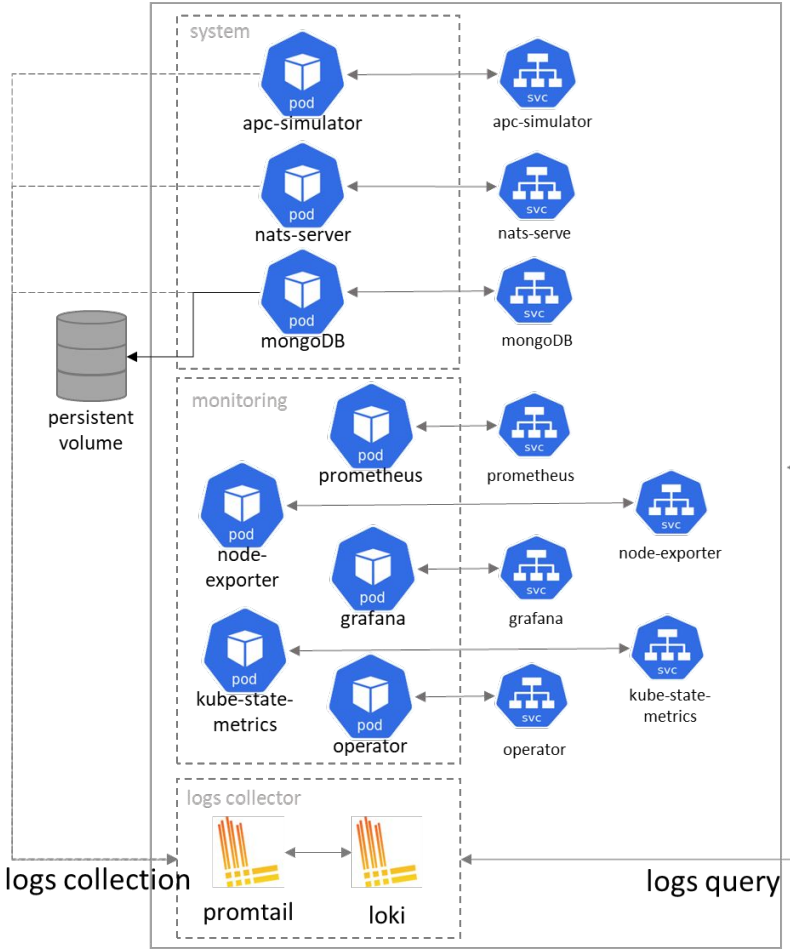
## 第 8 組

成員: 許蔓萍、王麒銘、林和俊、陳淞榆、游朝荃、葉詠富

# Outline

- System architecture & DigitalOcean Cloud Platform
- Deployment
  - [Requirements] apc-simulator & nats-server
  - [Configurable + Encryption] MongoDB & MongoDB Secret
  - [Application Mnoitor] Prometheus operator
  - [Application Mnoitor] Grafana
  - [Code Quality] CI/CD
  - [Application Mnoitor] Logging: Grafana + Loki + Promtail
- Development
  - [Code Optimization] APC Service strategy extension
  - [Code Optimization] Measure Service can easily extend more types
  - [Code Quality] User DB - mongoCache
  - [Code Optimization] Metrics transmission to Prometheus
  - [Application Mnoitor] Logger: custom json format
- Testing
  - [Code Quality] Test coverage enhancement
  - [Code Quality] Package Usage - Mock Nats and Supertest
- Demo
- Q & A

# System architecture



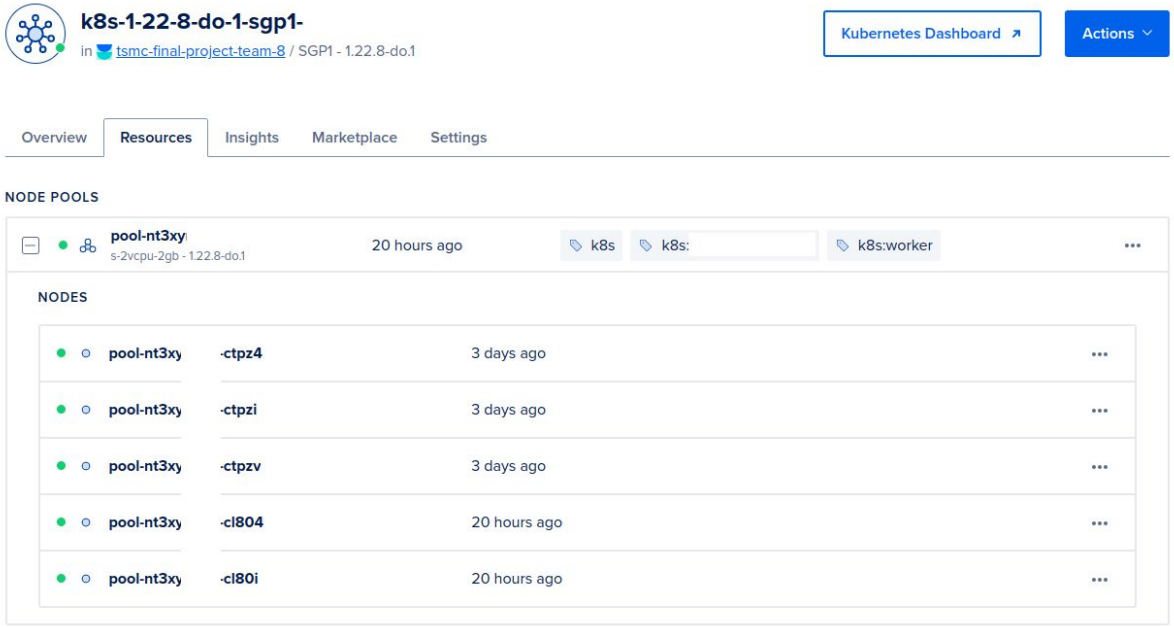

pull metrics

Prometheus

Grafana

# DigitalOcean

- Kubernetes Clusters (5 nodes)
- Container Registry



# Deployment

# Deploy apc-simulator & nats-server

- 在 apc yaml 增加環境變數給 MongoDB 連接使用

```yaml
env:
  - name: PORT
    value: '3030'
  - name: NATS_SERVICE_CONNECTION
    value: 'nats-server:4222'
  - name: MONGO_HOST
    value: 'mongo-svc:27017'
  - name: MONGO_INITDB_ROOT_USERNAME
    valueFrom:
      secretKeyRef:
        name: mongo-creds
        key: username
  - name: MONGO_INITDB_ROOT_PASSWORD
    valueFrom:
      secretKeyRef:
        name: mongo-creds
        key: password
  - name: MONGO_CACHE_DB_NAME
    value: mongo_cache
  - name: MONGO_CACHE_COLLECTION_NAME
    value: bucket_1
```

- 部屬apc-simulator
  - **kubectl apply -f deployment.yaml**
- 部屬nats-server
  - **kubectl apply -f deployment.yaml**
  - **kubectl apply -f service.yaml**

# Deploy MongoDB

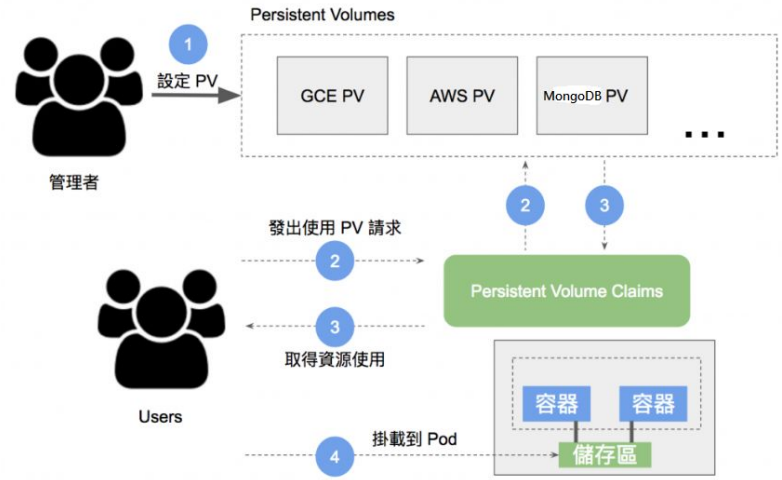
- **Secret** 是用於向 container 提供敏感訊息，數據以 base64 編碼格式儲存
- 使用 Secrets 將我們想要的Mongodb密碼 load 到容器中(apc-simulator)
- 使用完 secret yaml 檔後可刪除或加入.gitignore

```yaml
apiVersion: v1
data:
  password: cGFzc3dvcmQxMjM=
  username: YWRtaW51c2Vy
kind: Secret
metadata:
  creationTimestamp: null
  name: mongo-creds
```

- 部屬 MongoDB secret
  - **kubectl apply -f MongoDB-secrets.yaml**

# Deploy MongoDB(cont.)

- 我們需要 **Volumns** 儲存永久資料，即使pods 掛掉，資料仍然存在
- PersistentVolumes (PV): 由管理者設定儲存位置、容量的PV
- Persistent Volume Claims (PVC): 使用者使用 PV 動態配置 PV
- 部屬 PV
  - **kubectl apply -f MongoDB-pv.yaml**
  - **kubectl apply -f MongoDB-pvc.yaml**




MongoDB-pv.yaml

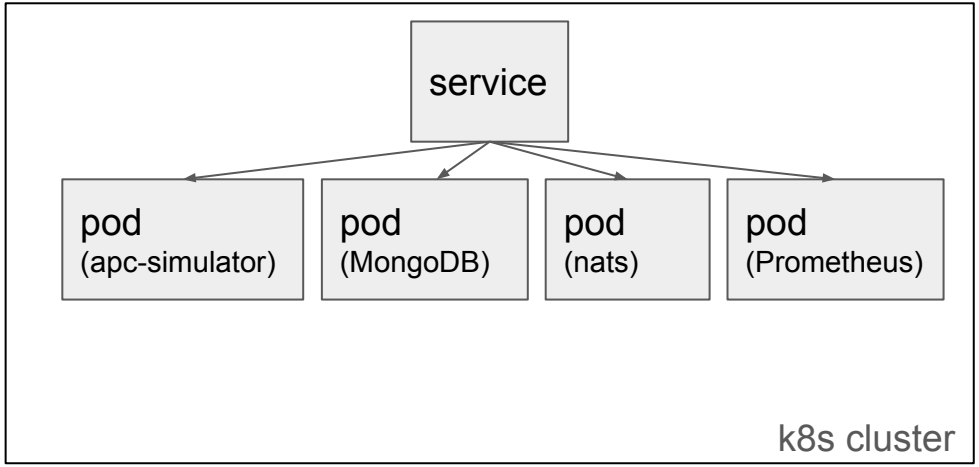
```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mongo-data-pv
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 1Gi
  hostPath:
    path: /data/mongo
```

MongoDB-pvc.yaml

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mongo-data
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

# Deploy MongoDB(cont.)

- 我們需要 **Services** 在 k8s 用來與其他 pod 建立通訊
- ClusterIP 通常用來做內部 pods 間的通訊，所以我們設定 port 27017 開放給同 cluster 中 apc-simulator 連
- 部屬 service
  - **kubectl apply -f MongoDB-svc.yaml**




MongoDB-svc.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: mongo-svc
spec:
  selector:
    app: mongo
  ports:
    - protocol: TCP
      port: 27017
      targetPort: 27017
```

# Deploy MongoDB(cont.)

- 最後依照前面設定部屬MongoDB
  - **kubectl apply -f MongoDB-deployment.yaml**

MongoDB-deployment.yaml

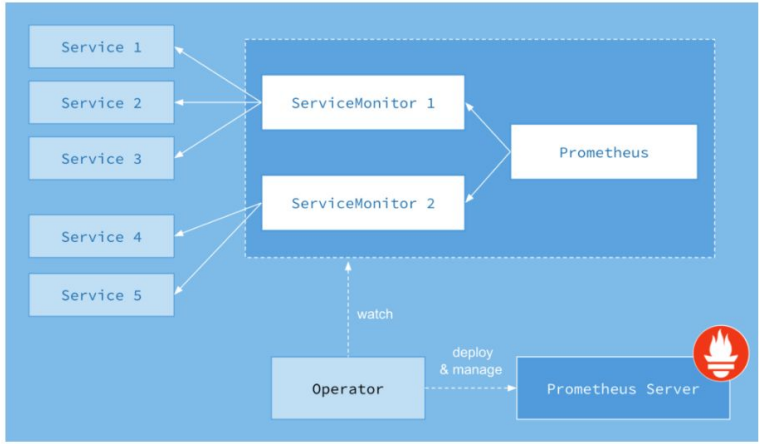
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: mongo
  name: mongo
```

...

```yaml
  env:
  - name: MONGO_INITDB_ROOT_USERNAME
    valueFrom:
      secretKeyRef:
        name: mongo-creds
        key: username
  - name: MONGO_INITDB_ROOT_PASSWORD
    valueFrom:
      secretKeyRef:
        name: mongo-creds
        key: password
  volumeMounts:
  - name: "mongo-data-dir"
    mountPath: "/data/db"
volumes:
- name: "mongo-data-dir"
  persistentVolumeClaim:
    claimName: "mongo-data"
```

# Deploy Prometheus Operator

- Helm: 更容易管理設定檔
  - 為一個服務中，各種元件裡的 yaml 檔統一打包成 chart，然後透過給參數的方式，去同時管理與設定 yaml 檔案
  - e.g: chart -> yaml file -> apply kubectl
- Prometheus Operator : 簡化部署與維護 Prometheus 上的事情



# Deploy Prometheus Operator(cont.)

1. 加入 Prometheus Community Helm repo
   - helm repo add Prometheus-community https://Prometheus-community.GitHub.io/helm-charts
2. 利用 Helm 來建立與管理 Prometheus Operator
   - Prometheus-community/kube-Prometheus-stack
3. 若是想要改動裡面的資訊，可以直接寫另外一個新的 yaml 檔去 apply
   - **helm install kube-Prometheus-stack Prometheus-community/kube-Prometheus-stack --namespace monitoring --create-namespace --values values.yaml**

# Deploy Prometheus Operator(cont.)

- Prometheus / operator / alertmanager / Grafana
- node-exporter: k8s cluster 中有 五個 node，所以會有五個 node-exporter (藍色框)
- kube-state-metrics: 收集 api server 獲得集群內部的資料(ex: pod state, container state, endpoints, service)，並 expose 出來

```
NAME                                                     READY   STATUS    RESTARTS   AGE
pod/alertmanager-kube-prometheus-stack-alertmanager-0    2/2     Running   0          23h
pod/kube-prometheus-stack-grafana-85567dd97f-86s5g       3/3     Running   0          23h
pod/kube-prometheus-stack-kube-state-metrics-d699cc95f-dmht8   1/1     Running   0          23h
pod/kube-prometheus-stack-operator-7c9699d9f4-s94gj      1/1     Running   0          23h
pod/kube-prometheus-stack-prometheus-node-exporter-7dj2q 1/1     Running   0          20h
pod/kube-prometheus-stack-prometheus-node-exporter-b8vf6 1/1     Running   0          20h
pod/kube-prometheus-stack-prometheus-node-exporter-hbw7r 1/1     Running   0          23h
pod/kube-prometheus-stack-prometheus-node-exporter-kc8d2 1/1     Running   0          23h
pod/kube-prometheus-stack-prometheus-node-exporter-n8f29 1/1     Running   0          23h
pod/prometheus-kube-prometheus-stack-prometheus-0        2/2     Running   0          23h
```

# Deploy Prometheus Operator(cont.)

- 在 apc-simulator 中，我們有把 moisture、thickness 的資訊丟進 apc-simulator /metrics 中
- 在 values.yaml 中新增以下 code

```yaml
prometheus:
  prometheusSpec:
    additionalScrapeConfigs:
      - job_name: app
        scrape_interval: 15s
        kubernetes_sd_configs:
        - role: service
          namespaces:
            names:
              - default
        relabel_configs:
        - source_labels: [__meta_kubernetes_service_name]
          action: replace
          target_label: service
        - source_labels: [__meta_kubernetes_service_name]
          action: keep
          regex: apc-simulator
```

```
PS C:\Users\Hochun\Desktop\cloud-native-final-project> kubectl get service  --kubeconfig=../k8s-1-22-8-do-1-sgp1-1654240
572373-kubeconfig.yaml
NAME            TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)     AGE
apc-simulator   ClusterIP   10.245.223.70    <none>        8080/TCP    4d1h
kubernetes      ClusterIP   10.245.0.1       <none>        443/TCP     4d22h
mongo-svc       ClusterIP   10.245.205.226   <none>        27017/TCP   2d14h
nats-server     ClusterIP   10.245.252.143   <none>        4222/TCP    4d17h
```

# Deploy Prometheus Operator(cont.)

- 使用 put forwarding 從 local 存取 K8S 內的 Prometheus Service
  - **kubectl port-forward -n=monitoring svc/kube-Prometheus-stack-Prometheus 9090:9090**
  - 執行完成後，可從 **http://localhost:9090/** 進入 Prometheus

# Deploy Prometheus Operator(cont.)

Prometheus Alerts Graph Status Help

app (1/1 up) show less

| Endpoint | State | Labels | Last Scrape | Scrape Duration | Error |
|---|---|---|---|---|---|
| http://apc-simulator.default.svc:8080/metrics | UP | instance="apc-simulator.default.svc:8080" job="app" service="apc-simulator" | 15.383s ago | 18.105ms | |

serviceMonitor/monitoring/kube-prometheus-stack-alertmanager/0 (1/1 up) show more

serviceMonitor/monitoring/kube-prometheus-stack-apiserver/0 (1/1 up) show more

serviceMonitor/monitoring/kube-prometheus-stack-coredns/0 (2/2 up) show more

serviceMonitor/monitoring/kube-prometheus-stack-kube-proxy/0 (5/5 up) show more

serviceMonitor/monitoring/kube-prometheus-stack-kube-state-metrics/0 (1/1 up) show more

serviceMonitor/monitoring/kube-prometheus-stack-kubelet/0 (5/5 up) show more

serviceMonitor/monitoring/kube-prometheus-stack-kubelet/1 (5/5 up) show more

serviceMonitor/monitoring/kube-prometheus-stack-kubelet/2 (5/5 up) show more

serviceMonitor/monitoring/kube-prometheus-stack-operator/0 (1/1 up) show more

serviceMonitor/monitoring/kube-prometheus-stack-prometheus-node-exporter/0 (5/5 up) show more

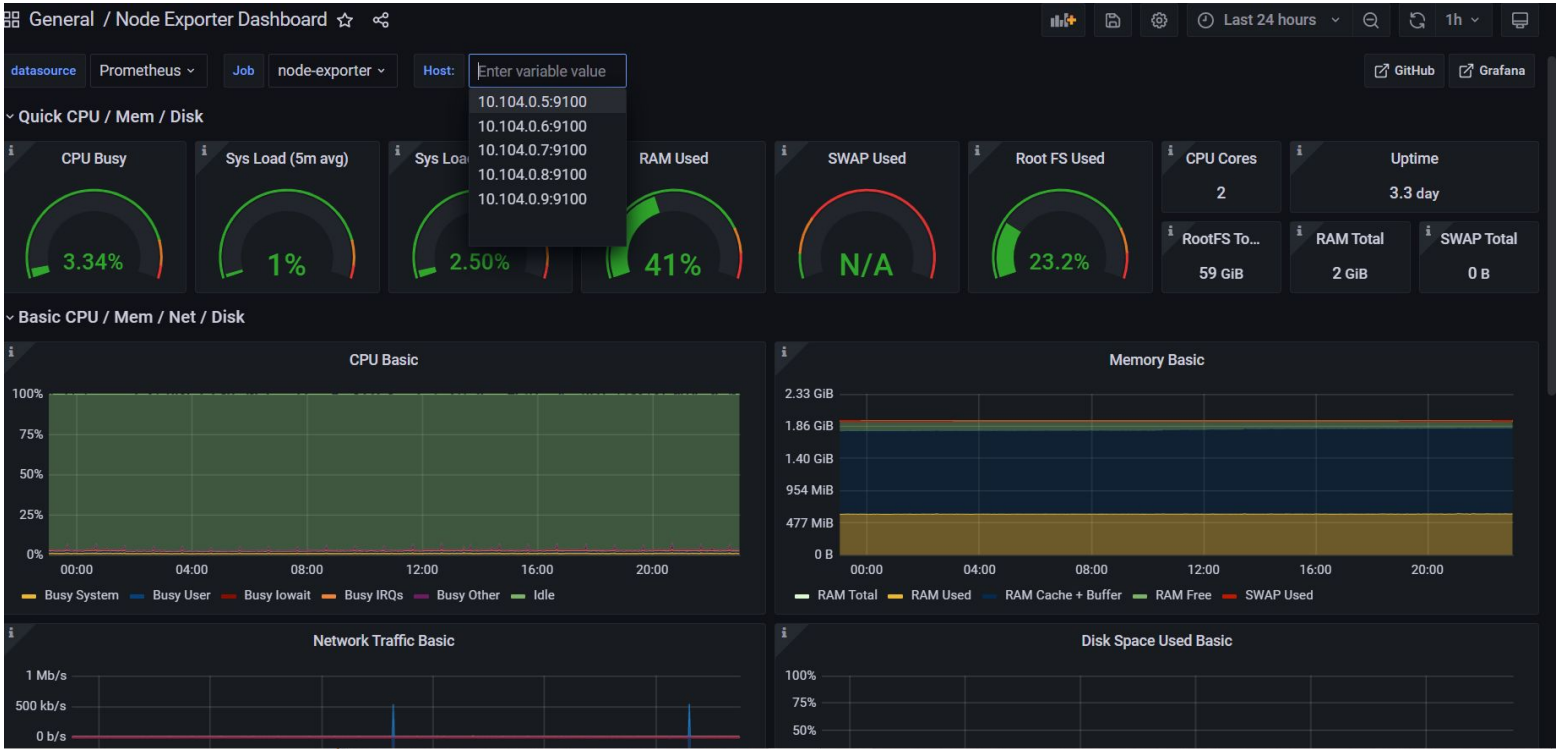
serviceMonitor/monitoring/kube-prometheus-stack-prometheus/0 (1/1 up) show more

# Deploy Grafana(cont.)

- 使用 put forwarding 從 local 存取 K8S 內的 Grafana Service
  - **kubectl port-forward -n=monitoring svc/kube-Prometheus-stack-Grafana 8080:80**
  - 執行完成後，可從 **http://localhost:8080/** 進入 Grafana
- 我們建立了三個 dashboard 來觀察此 k8s cluster
  - node exporter 整體狀況
  - apc-simulator service 整體狀況
  - apc-simulator service factor 傳出參數
- 我們額外有做 Loki，利用 query 取得指定 log 資訊

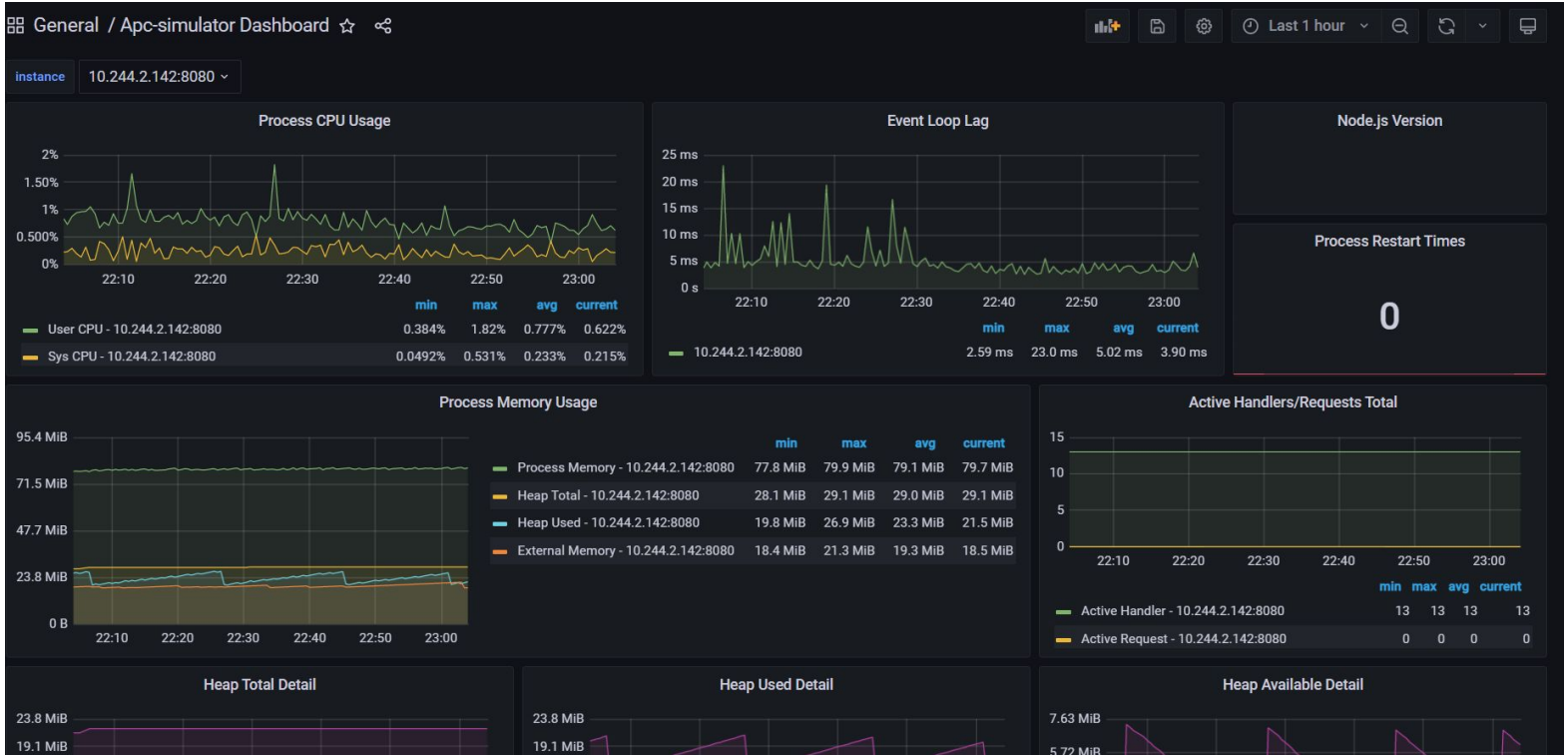
# Deploy Grafana(cont.)

- node exporter 整體狀況



# Deploy Grafana(cont.)

- apc-simulator service 整體狀況

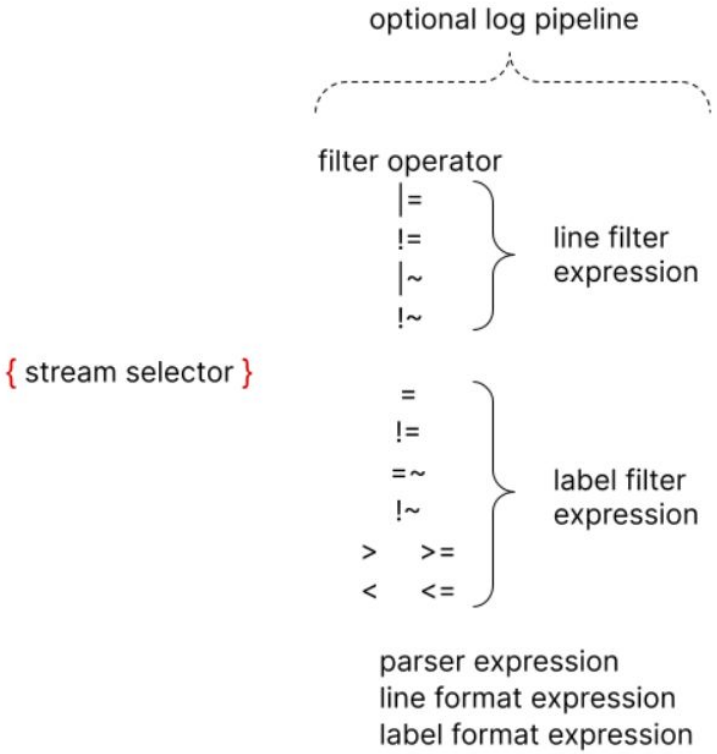


# Deploy Grafana(cont.)

- apc-simulator service 傳出參數

# Deploy Grafana(cont.)

- Loki: 查找 log 有兩個方法
  - line filter expression: 直接查找 log 中是否有相同的 substring
  - label filter expression: 可利用 label 查找想要的資訊
- Example: 想查詢 moisture factor 大於 0.35 的值有哪些

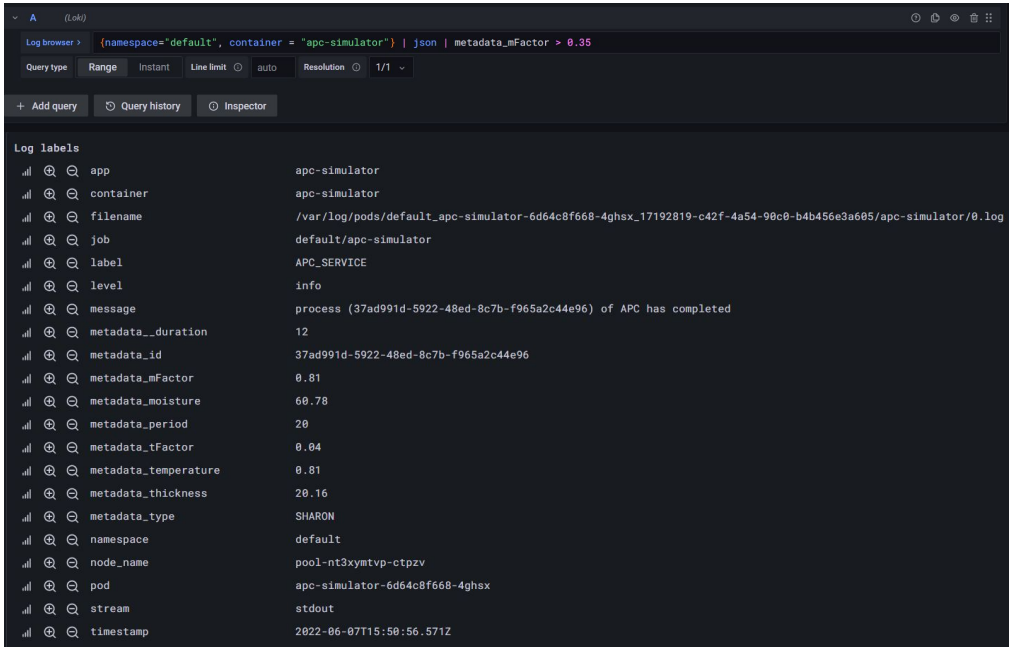


# Deploy Grafana(cont.)

- Before: 使用line filter expression, log 格式無法抓取想要的資訊

# Deploy Grafana(cont.)

- After: 修改 log 成 json 輸出後，利用 label filter expression，用 label vector 抓取我們想要的數值區域

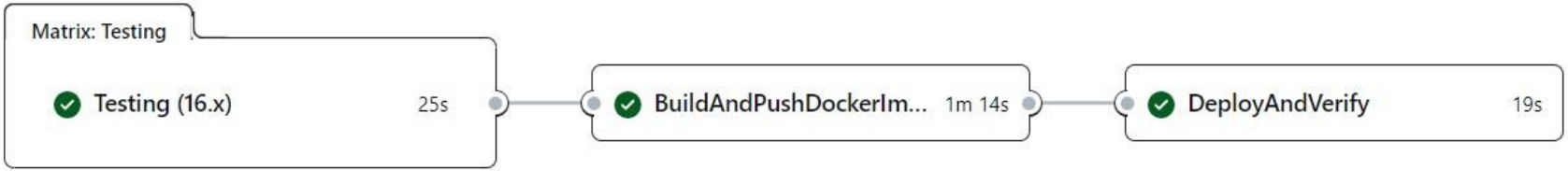


# Deploy Grafana(cont.)


```
> 2022-06-07 23:50:56 {"label":"APC_SERVICE","timestamp":"2022-06-07T15:50:56.571Z","level":"info","message":"process (37ad991d-5922-48ed-8c7b-f965a2c44e96) of APC has completed","m
etadata":{"_duration":12,"id":"37ad991d-5922-48ed-8c7b-f965a2c44e96","type":"SHARON","thickness":"20.16","moisture":"60.78","tFactor":"0.04","mFactor":"0.8
1","period":20,"temperature":"0.81"}}
> 2022-06-07 23:50:26 {"label":"APC_SERVICE","timestamp":"2022-06-07T15:50:26.569Z","level":"info","message":"process (8351923d-aa30-40fe-b589-d5a830bb1080) of APC has completed","m
etadata":{"_duration":10,"id":"8351923d-aa30-40fe-b589-d5a830bb1080","type":"SHARON","thickness":"20.47","moisture":"60.48","tFactor":"0.52","mFactor":"0.4
6","period":20,"temperature":"10.64"}}
> 2022-06-07 23:49:56 {"label":"APC_SERVICE","timestamp":"2022-06-07T15:49:56.563Z","level":"info","message":"process (e458fb61-d2f3-4079-af36-332749906b51) of APC has completed","m
etadata":{"_duration":8,"id":"e458fb61-d2f3-4079-af36-332749906b51","type":"SHARON","thickness":"20.97","moisture":"60.67","tFactor":"0.42","mFactor":"0.64","p
eriod":20,"temperature":"8.81"}}
> 2022-06-07 23:49:46 {"label":"APC_SERVICE","timestamp":"2022-06-07T15:49:46.563Z","level":"info","message":"process (c066e4b0-57dc-4d73-8b2b-8d6293e423bc) of APC has completed","m
etadata":{"_duration":10,"id":"c066e4b0-57dc-4d73-8b2b-8d6293e423bc","type":"RIB_EYE","thickness":"20.54","moisture":"60.53","tFactor":"0.13","mFactor":"0.7
9","period":"47.82","temperature":100}}
> 2022-06-07 23:49:36 {"label":"APC_SERVICE","timestamp":"2022-06-07T15:49:36.562Z","level":"info","message":"process (b9631f39-3f44-40bb-8c76-7cec3c21a91e) of APC has completed","m
etadata":{"_duration":8,"id":"b9631f39-3f44-40bb-8c76-7cec3c21a91e","type":"RIB_EYE","thickness":"20.31","moisture":"60.47","tFactor":"0.13","mFactor":"0.7
9","period":"47.77","temperature":100}}
> 2022-06-07 23:49:26 {"label":"APC_SERVICE","timestamp":"2022-06-07T15:49:26.563Z","level":"info","message":"process (6f65fcb0-5a1c-4dff-ba8c-6bef8fc4669b) of APC has completed","m
etadata":{"_duration":10,"id":"6f65fcb0-5a1c-4dff-ba8c-6bef8fc4669b","type":"SHARON","thickness":"20.29","moisture":"60.12","tFactor":"0.11","mFactor":"0.7
8","period":20,"temperature":"2.23"}}
> 2022-06-07 23:49:16 {"label":"APC_SERVICE","timestamp":"2022-06-07T15:49:16.567Z","level":"info","message":"process (df16ed81-2e4f-4206-bb86-c910606fddad) of APC has completed","m
etadata":{"_duration":14,"id":"df16ed81-2e4f-4206-bb86-c910606fddad","type":"RIB_EYE","thickness":"20.74","moisture":"60.32","tFactor":"0.24","mFactor":"0.4
6","period":"27.75","temperature":100}}
> 2022-06-07 23:49:06 {"label":"APC_SERVICE","timestamp":"2022-06-07T15:49:06.579Z","level":"info","message":"process (8ddc8f85-73e8-4a96-8f74-95c9c0433c68) of APC has completed","m
etadata":{"_duration":11,"id":"8ddc8f85-73e8-4a96-8f74-95c9c0433c68","type":"RIB_EYE","thickness":"20.38","moisture":"60.84","tFactor":"0.24","mFactor":"0.4
6","period":"27.99","temperature":100}}
> 2022-06-07 23:48:56 {"label":"APC_SERVICE","timestamp":"2022-06-07T15:48:56.549Z","level":"info","message":"process (facdc1bf-ec4f-4b97-818a-041e87145832) of APC has completed","m
etadata":{"_duration":16,"id":"facdc1bf-ec4f-4b97-818a-041e87145832","type":"SHARON","thickness":"20.68","moisture":"60.45","tFactor":"0.21","mFactor":"0.7
5","period":20,"temperature":"4.34"}}
> 2022-06-07 23:48:46 {"label":"APC_SERVICE","timestamp":"2022-06-07T15:48:46.542Z","level":"info","message":"process (ab78e138-be8c-4bc8-8ce4-dae2466c5ff5) of APC has completed","m
etadata":{"_duration":8,"id":"ab78e138-be8c-4bc8-8ce4-dae2466c5ff5","type":"RIB_EYE","thickness":"20.35","moisture":"60.99","tFactor":"0.85","mFactor":"0.5
1","period":"31.10","temperature":100}}
> 2022-06-07 23:48:36 {"label":"APC_SERVICE","timestamp":"2022-06-07T15:48:36.550Z","level":"info","message":"process (3e1e4ea0-a4e3-4dbd-8422-a09d7df38e1c) of APC has completed","m
etadata":{"_duration":15,"id":"3e1e4ea0-a4e3-4dbd-8422-a09d7df38e1c","type":"SHARON","thickness":"20.18","moisture":"60.06","tFactor":"0.85","mFactor":"0.5
1","period":20,"temperature":"17.15"}}
> 2022-06-07 23:48:26 {"label":"APC_SERVICE","timestamp":"2022-06-07T15:48:26.544Z","level":"info","message":"process (94ff1269-3e57-4137-901a-b5a0e2559754) of APC has completed","m
etadata":{"_duration":10,"id":"94ff1269-3e57-4137-901a-b5a0e2559754","type":"RIB_EYE","thickness":"20.77","moisture":"60.51","tFactor":"0.57","mFactor":"0.5
0","period":"30.25","temperature":100}}
```

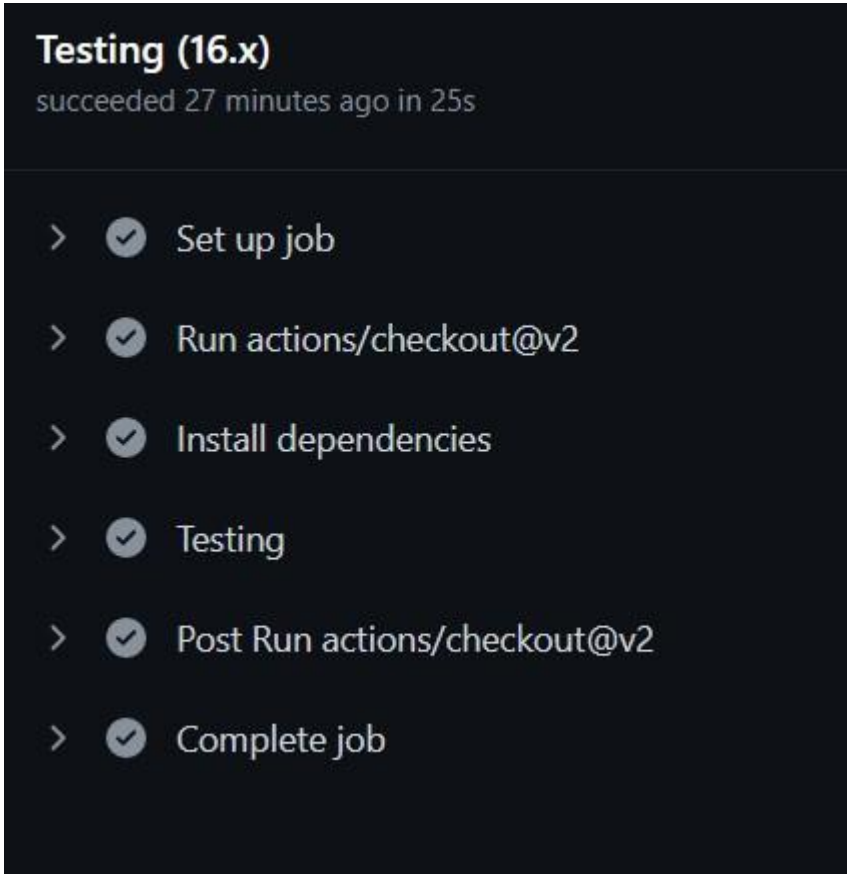
# CI/CD

1. Testing
2. Build and push Docker Image
3. Deploys to our DigitalOcean Kubernetes cluster



# CI/CD-Testing

1. node-version 安裝
2. actions/checkout@v2 讓我們訪問到倉庫
3. 安裝 application 所需的 dependencies
4. npm run test -- --coverage 可以在測試過程中計算我們的測試覆蓋率



# GitHub secrets setting

1. 設定 GitHub Actions secrets 環境變數

## Actions secrets

New repository secret

Secrets are environment variables that are **encrypted**. Anyone with **collaborator** access to this repository can use these secrets for Actions.

Secrets are not passed to workflows that are triggered by a pull request from a fork. Learn more.

| | | | |
|---|---|---|---|
| CLUSTER_NAME | Updated yesterday | Update | Remove |
| DIGITALOCEAN_ACCESS_TOKEN | Updated 2 days ago | Update | Remove |
| REGISTRY_ENDPOINT | Updated yesterday | Update | Remove |

# GitHub secrets setting(cont.)

```yaml
# Install doctl
- name: Install doctl
  uses: digitalocean/action-doctl@v2
  with:
    token: ${{ secrets.DIGITALOCEAN_ACCESS_TOKEN }}
```

```yaml
# Save DigitalOcean kubeconfig with short-lived credentials
- name: Save DigitalOcean kubeconfig with short-lived credentials
  run: doctl kubernetes cluster kubeconfig save --expiry-seconds 600 ${{ secrets.CLUSTER_NAME }}
```

# GitHub secrets setting(cont.)

**apc-simulator**
13 Images

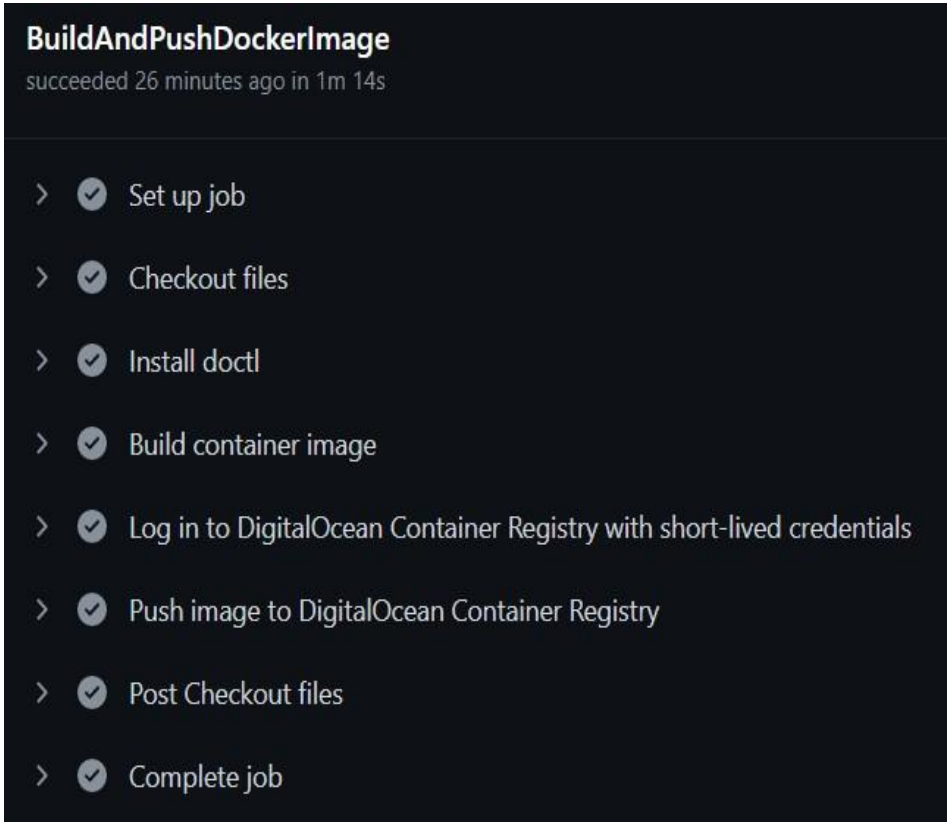
Recent image tag: de8ae43 | 40 minutes ago

IMAGES (13) | Delete

| Digest | Tags | Size | Last Pushed |
|---|---|---|---|
| sha256:7bd0cc514e2d3ff81c7c8 | de8ae43 | 361.21 MB | 40 minutes ago |
| sha256:bbc8253da73a5ba638a01 | 6d5deb4 | 361.21 MB | 1 hour ago |
| sha256:a108a068bc754c05e087b | eaabd61 | 361.21 MB | 1 hour ago |
| sha256:a82c4106bf78ec2dbcdca | a086d98 | 361.21 MB | 1 hour ago |
| sha256:9a7e639f662728ac0cae8 | 646a2bd | 361.21 MB | 1 hour ago |
| sha256:1526c98b4fbee4c849c0d | cf8b280 | 361.21 MB | 1 hour ago |
| sha256:a4ff66c3456e7d0251476 | 6ba0cd7 | 361.21 MB | 1 hour ago |
| sha256:8013b7a443e49a9842335 | c218b00 | 361.21 MB | 1 hour ago |
| sha256:5556717fa40f9ad5aeb32 | 5bf8612 | 361.21 MB | 2 hours ago |
| sha256:6e9f3a28224474e4d4568 | 1c20f53 | 361.21 MB | 2 hours ago |
| sha256:dda0d050cc5cf3b2e70f2 | 4b880c5 | 361.21 MB | 2 hours ago |
| sha256:821d8d3d648c570bc9815 | 5fda417 | 361.21 MB | 2 hours ago |
| sha256:c3b30544677785c27ad8d | 4b1e88f | 359.02 MB | 6 hours ago |

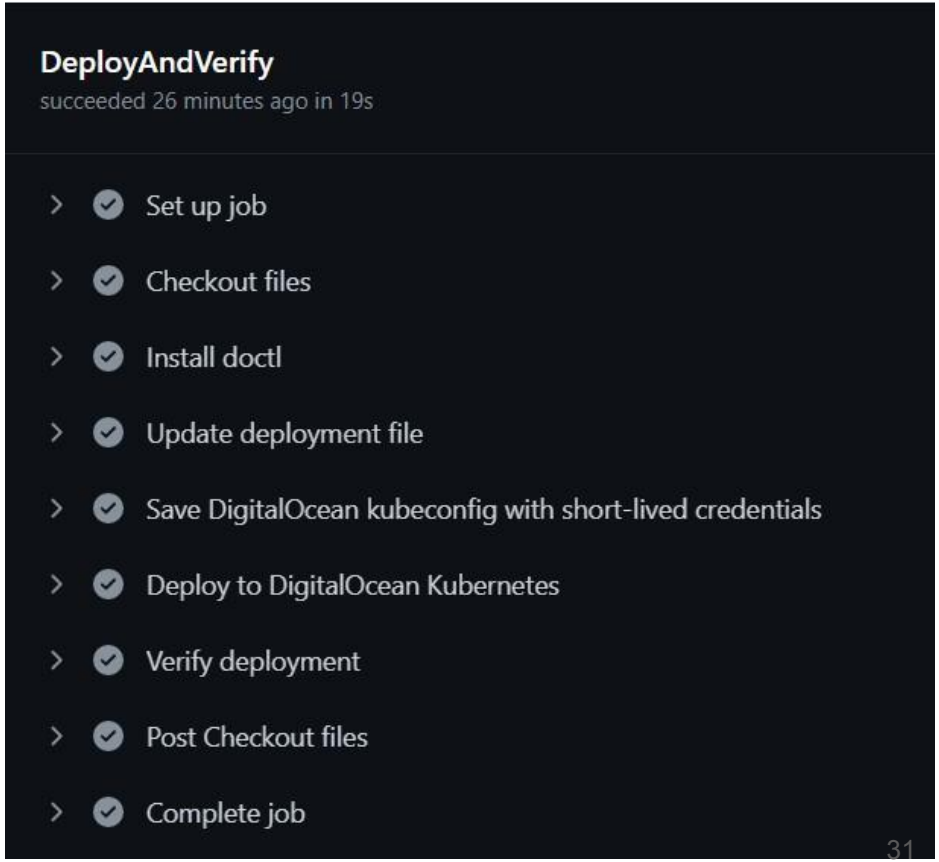
# CI/CD-Build and push Docker Image

1. 安裝 doctl
2. 利用 $GITHUB_SHA tag 建立 Docker Image
3. 透過 `--expiry-seconds` flag 生成短期憑證讓我們登錄 DigitalOcean Container Registry
4. 最後，再把我們的 Docker Image push 上去



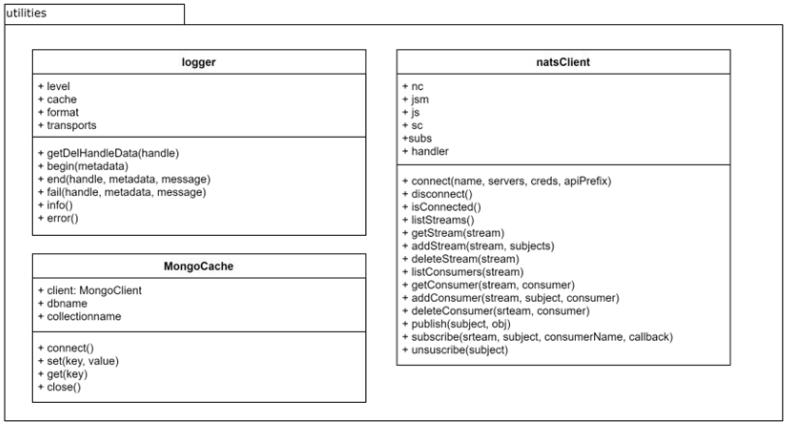
# CI/CD-Deploys to our DigitalOcean Kubernetes cluster

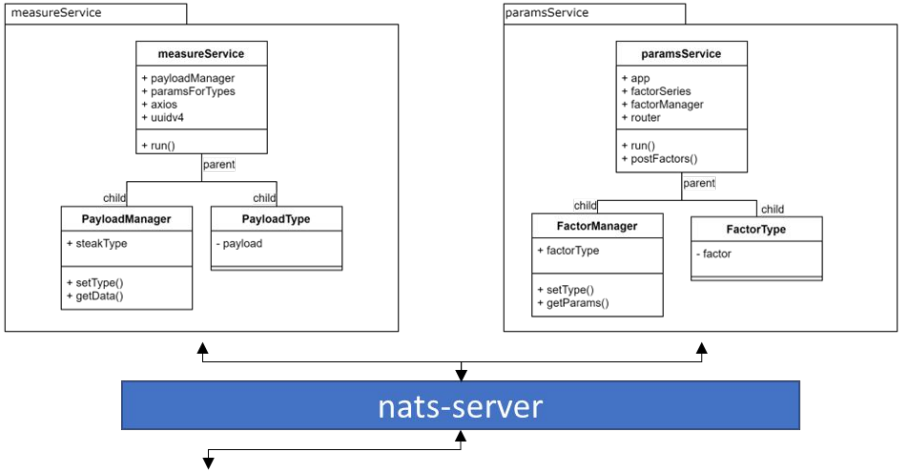
1. 將 push 後的 Docker Image 更新，並更新 deployment.yml file
2. 儲存特定 cluster 的憑證到 kubeconfig
3. 部屬 DigitalOcean Kubernetes
4. 最後，驗證以確認我們是否已成功部屬

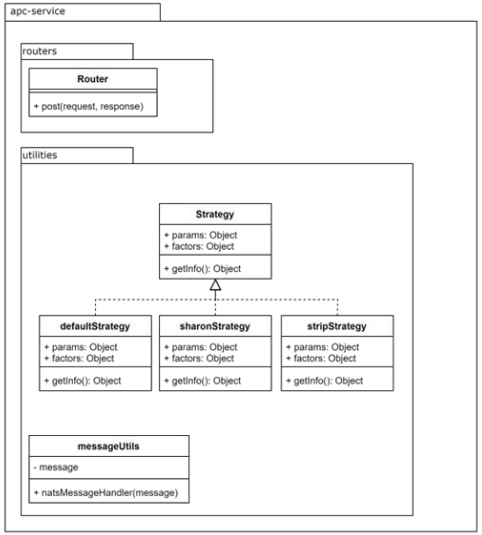


# Development

# UML diagram


utilities
logger
+ level
+ cache
+ format
+ transports
+ getDelHandleData(handle)
+ begin(metadata)
+ end(handle, metadata, message)
+ fail(handle, metadata, message)
+ info()
+ error()
MongoCache
+ client: MongoClient
+ dbname
+ collectionname
+ connect()
+ set(key, value)
+ get(key)
+ close()
natsClient
+ nc
+ jsm
+ js
+ sc
+subs
+ handler
+ connect(name, servers, creds, apiPrefix)
+ disconnect()
+ isConnected()
+ listStreams()
+ getStream(stream)
+ addStream(stream, subjects)
+ deleteStream(stream)
+ listConsumers(stream)
+ getConsumer(stream, consumer)
+ addConsumer(stream, subject, consumer)
+ deleteConsumer(srteam, consumer)
+ publish(subject, obj)
+ subscribe(srteam, subject, consumerName, callback)
+ unsuscribe(subject)


measureService
measureService
+ payloadManager
+ paramsForTypes
+ axios
+ uuidv4
+ run()
parent
child
child
PayloadManager
+ steakType
+ setType()
+ getData()
PayloadType
- payload
paramsService
paramsService
+ app
+ factorSeries
+ factorManager
+ router
+ run()
+ postFactors()
parent
child
child
FactorManager
+ factorType
+ setType()
+ getParams()
FactorType
- factor
nats-server


apc-service
routers
Router
+ post(request, response)
utilities
Strategy
+ params: Object
+ factors: Object
+ getInfo(): Object
defaultStrategy
+ params: Object
+ factors: Object
+ getInfo(): Object
sharonStrategy
+ params: Object
+ factors: Object
+ getInfo(): Object
stripStrategy
+ params: Object
+ factors: Object
+ getInfo(): Object
messageUtils
- message
+ natsMessageHandler(message)

# APC Service strategy extension

```
class Strategy {
  constructor(params, factors) {
      this.params = params;
      this.factors = factors;

  };
  getInfo(){

  }
}
```

```
class defaultStrategy extends Strategy{

  getInfo() {
    const period = (this.params.moisture * this.factors.mFactor).toFixed(2);

    return {
      period: period,
      temperature: 100,
    };
  }
}

class sharonStrategy extends  Strategy{

  getInfo() {
    const temperature = (this.params.thickness * this.factors.tFactor).toFixed(2);

    return {
      period: 20,
      temperature,
    };
  }
}

class stripStrategy extends  Strategy{

  getInfo() {
    const temperature = (this.params.thickness * this.factors.tFactor).toFixed(2);
    const period = (this.params.moisture * this.factors.mFactor + 20).toFixed(2);

    return {
      period,
      temperature,
    };
  }
}
```

params: {
“moisture”: moisture,
“thickness”: thickness
}

factors: {
“tFactor”: tFactor,
“mFactor”: mFactor
}

New strategy with multiple parameters and factors

# Refactor functions

```
const run = async () => {
    const handler = setInterval(async () => {
        const index = Math.floor((Math.random() * 10) % types.length);
        const id = uuidv4();

        const payload = {
            id,
            type: types[index],
            thickness: 2 + Math.random().toFixed(2),
            moisture: 6 + Math.random().toFixed(2),
        };

        global.thickness_metric.set(parseFloat(paramsForPayloadType.thickness));
        global.moisture_metric.set(parseFloat(paramsForPayloadType.moisture));

        const {
            data
        } = await axios.post(`${domainService.apc.endpoint}/api/v1/process`, payload);
    }, cron.measurePeriod);

    return handler;
};
```

# Measure Service can easily extend more types

- use an array to store the more types, adjust the neccessary params in the array

```
const payload = {
  id,
  type: types[index],
  thickness: 2 + Math.random().toFixed(2),
  moisture: 6 + Math.random().toFixed(2),
};
```

```
const paramsForTypes = (id) => ([
  {
    id,
    type: 'SHARON',
    thickness: 2 + Math.random().toFixed(2),
    moisture: 6 + Math.random().toFixed(2),
  },
  {
    id,
    type: 'RIB_EYE',
    thickness: 2 + Math.random().toFixed(2),
    moisture: 6 + Math.random().toFixed(2),
  },
])
```

# Refactor functions

```
const run = async () => {
    const handler = setInterval(async () => {
        const index = Math.floor((Math.random() * 10) % types.length);
        const id = uuidv4();

        const payload = {
            id,
            type: types[index],
            thickness: 2 + Math.random().toFixed(2),
            moisture: 6 + Math.random().toFixed(2),
        };

        global.thickness_metric.set(parseFloat(paramsForPayloadType.thickness));
        global.moisture_metric.set(parseFloat(paramsForPayloadType.moisture));

        const {
            data
        } = await axios.post(`${domainService.apc.endpoint}/api/v1/process`, payload);
    }, cron.measurePeriod);

    return handler;
};
```

# Refactor functions

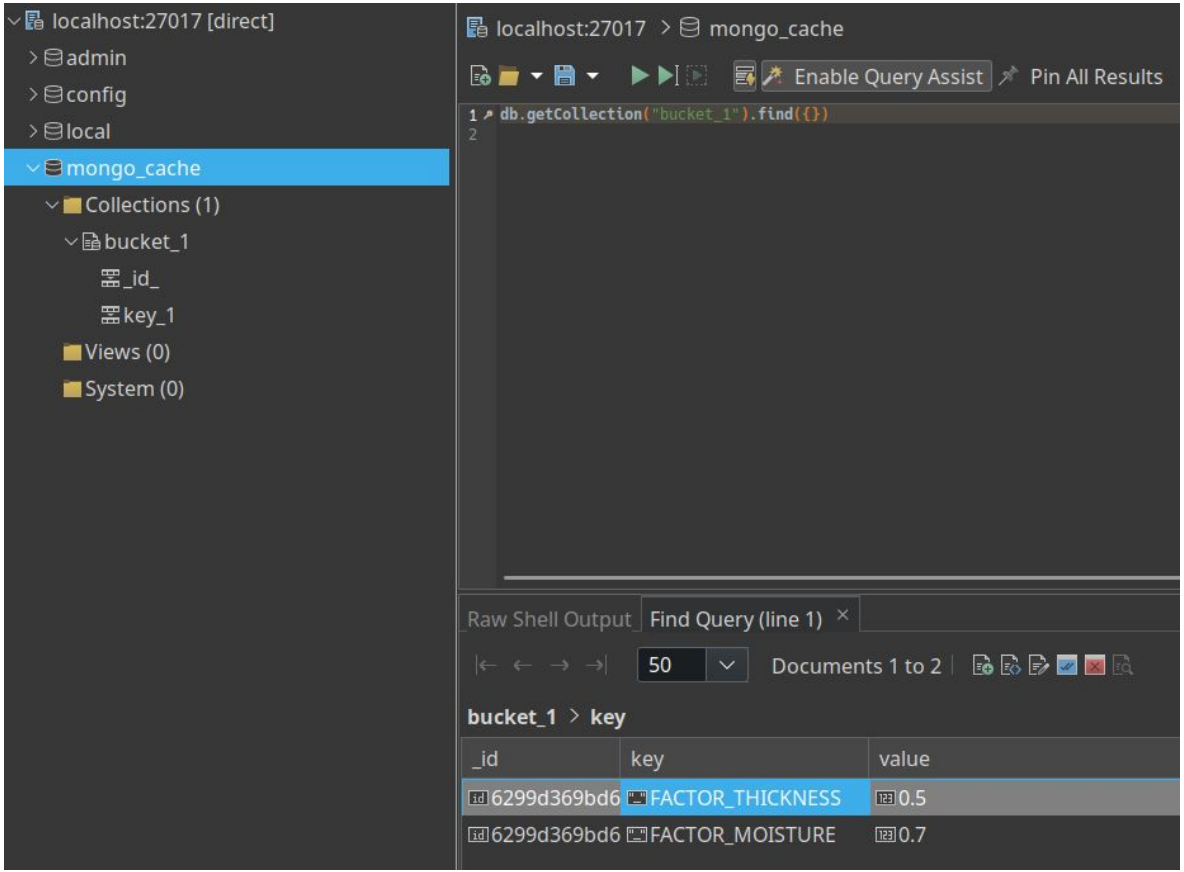
```
const run = async () => {
    const handler = setInterval(async () => {
        const {
            payload,
            paramsForPayloadType
        } = setPayload();

        setMetric(paramsForPayloadType);

        const {
            data
        } = await axios.post(`${domainService.apc.endpoint}/api/v1/process`, payload);
    }, cron.measurePeriod);

    return handler;
};
```

# MongoDB: mongoCache (取代 node-cache)

- 將 MongoDB 抽象為 Key-Value Database
  - key field 增加 Unique Indexes
- mongoCache 實作 node-cache 介面 (get / set)

```
const initGlobalCache = async () => {
    // Before
    // global.cache = new NodeCache();

    // After
    global.cache = new MongoCache(
        `mongodb://${process.env.MONGO_INITDB_ROOT_US
hSource=admin`,
        `${process.env.MONGO_CACHE_DB_NAME}`,
        `${process.env.MONGO_CACHE_COLLECTION_NAME}`
    );
```

```
class MongoCache {
    async set(key, value) {
        const query = { key: key };
        const update = { $set: { key: key, value: value }};
        const options = { upsert: true };

        this.client.db(this.dbName).collection(this.collectionName).updateOne(query, update, options);
    }

    async get(key) {
        const result = await this.client.db(this.dbName).collection(this.collectionName).findOne( { key: key } );

        if (!result) {
            return undefined;
        }

        if ('value' in result) {
            return result.value;
        }

        return undefined;
    }
```


localhost:27017 [direct]
admin
config
local
mongo_cache
Collections (1)
bucket_1
_id_
key_1
Views (0)
System (0)
localhost:27017 > mongo_cache
Enable Query Assist
Pin All Results
1 db.getCollection("bucket_1").find({})
2
Raw Shell Output
Find Query (line 1)
50
Documents 1 to 2
bucket_1 > key
_id
key
value
6299d369bd6
FACTOR_THICKNESS
0.5
6299d369bd6
FACTOR_MOISTURE
0.7

# Metrics transmission to Prometheus

- package: prom-client
- HTTP endpoint: apc-simulator/metrics
- Gauge: thickness / moisture / thickness_factor /  moisture_factor


```
# HELP thickness_factor thickness_factor_metric
# TYPE thickness_factor gauge
thickness_factor{app="apc-simulator"} 0.56

# HELP moisture_factor moisture_factor_metric
# TYPE moisture_factor gauge
moisture_factor{app="apc-simulator"} 0.37

# HELP thickness thickness_metric
# TYPE thickness gauge
thickness{app="apc-simulator"} 20.26

# HELP moisture moisture_metric
# TYPE moisture gauge
moisture{app="apc-simulator"} 60.73
```

# Logger: custom json format

- Loki / Promtail / Grafana
- src/utilities/logger.js

```
const customFormat = printf(({ timestamp, label, message, level, ...metadata }) => {
    // Before
    // return `[${label}] | ${timestamp} | ${level} | ${message} | ${JSON.stringify(metadata)}`;

    // After
    let ret = {
        label: label,
        timestamp: timestamp,
        level: level,
        message: message,
        metadata: metadata,
    }

    return JSON.stringify(ret);
});
```


```
{"label":"NATSClient","timestamp":"2022-06-07T18:08:34.308Z","level":"info","message":"Successfully connect
{"label":"MongoCache","timestamp":"2022-06-07T18:08:34.436Z","level":"info","message":"Successfully connect
{"label":"INDEX","timestamp":"2022-06-07T18:08:34.456Z","level":"info","message":"FACTOR_THICKNESS: 0.73","m
{"label":"INDEX","timestamp":"2022-06-07T18:08:34.456Z","level":"info","message":"FACTOR_MOISTURE\": 0.71","
{"label":"APC_SERVICE","timestamp":"2022-06-07T18:08:44.505Z","level":"info","message":"process (d971f83a-61
duration":10,"id":"d971f83a-61fc-4ab6-9de6-7c42ed5e7a5a","type":"SHARON","thickness":"20.84","moisture":"60.
:"15.21"}}
{"label":"INDEX","timestamp":"2022-06-07T18:08:49.468Z","level":"info","message":"0.75","metadata":{}}
{"label":"INDEX","timestamp":"2022-06-07T18:08:49.471Z","level":"info","message":"0.96","metadata":{}}
{"label":"PARAMS_SERVICE","timestamp":"2022-06-07T18:08:49.489Z","level":"info","message":"complete the proc
```

# Testing

# Testing

| | |
|---|---|
| apc Service | messageUtil:<br>1. Test for the response of natsMessageHandler<br>2. Test different type of factor for the if else branch in natsMessageHandler<br>3. Test when the type of factor is fake |
| | strategyUtil:<br>1. Test if the improvement of Strategy Model works properly |
| measureService | 1. Test for the parameters passed into POST /api/v1/process, should be properly ranged<br>2. Test for functions for expected results |
| paramsService | factor.js<br>1. use Supertest to test for the HTTP request<br>2. mock nats |

# Test coverage enhancement

| File | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s |
|---|---|---|---|---|---|
| All files | 60.37 | 25 | 55.55 | 62.74 | |
| apcService/utilities | 84.21 | 66.66 | 100 | 84.21 | |
| messageUtil.js | 75 | 66.66 | 100 | 75 | 5,14-16 |
| strategyUtil.js | 100 | 100 | 100 | 100 | |
| utilities | 47.05 | 0 | 33.33 | 50 | |
| logger.js | 47.05 | 0 | 33.33 | 50 | 29-35,39-42,46-51,55-60 |

| File | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s |
|---|---|---|---|---|---|
| All files | 84.87 | 60 | 82.6 | 86.2 | |
| config | 100 | 100 | 100 | 100 | |
| default.js | 100 | 100 | 100 | 100 | |
| src/apcService/utilities | 100 | 100 | 83.33 | 100 | |
| messageUtil.js | 100 | 100 | 100 | 100 | |
| strategyUtil.js | 100 | 100 | 80 | 100 | |
| src/measureService | 83.33 | 100 | 77.77 | 82.75 | |
| index.js | 83.33 | 100 | 77.77 | 82.75 | 67-75 |
| src/paramsService/routers/v1 | 80.64 | 50 | 100 | 80.64 | |
| factor.js | 80.64 | 50 | 100 | 80.64 | 21,34-36,50,63-65 |
| src/utilities | 80 | 40 | 83.33 | 84.84 | |
| logger.js | 80 | 40 | 83.33 | 84.84 | 39,63-68 |

# Usage Testing Package

- Supertest
  - 用來模擬 HTTP request 所需要用到的套件
  - 以 POST 為例:

```
const responseData = await request(app)
  .post('/api/v1/factor/moisture')
  .send({ factor: Math.random().toFixed(2) })
  .expect(200);
```

```
expect(responseData.ok).toBe(true);
```

# Use Testing Package

- Mock-nats-client
  - 以利於在跑 unit test 時, 可以模擬對 Nats 的行為
  - 與真實 Nats 在使用 API 上的呼叫相同
  - 並非真的對 Nats 做 publish, subscribe 等的行為, 而是在 memory 之中模擬出來

```
beforeEach(() => {
  global.natsClient = new MockNatsClient({ json: true });
});
```

```
await global.natsClient.publish(`test`, {
  type: 'FACTOR_MOISTURE',
  factor,
});
```

# Demo

- kubectl logs apc-simulator
- Prometheus
  - Service Discovery
  - target
- Grafana
  - node exporter dashboard
  - apc-simulator service dashboard
  - factor dashboard
- Loki
  - Example: 查詢 moisture factor 大於0.35 的 log 有哪些
  - Before: Line filter expression
  - After: Label filter expression

# Q & A

Thanks for listening