XREX

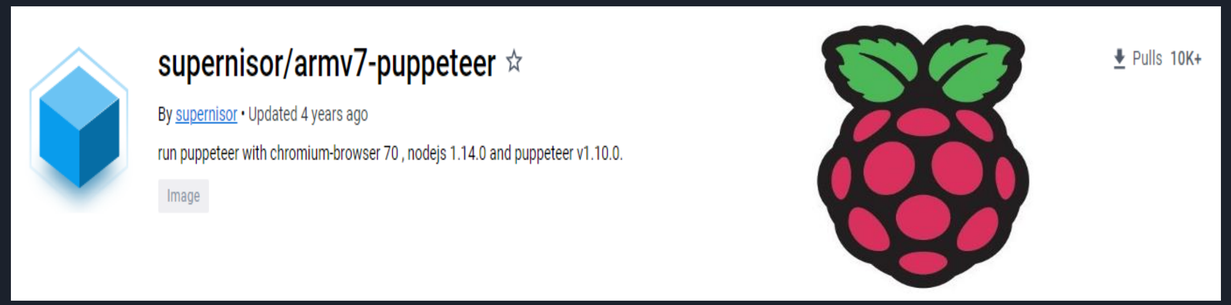
從單體到容器化導入之路

李太毓
Danny Lee
XREX SRE

XREX

我的第一個容器化專案 - puppeteer on raspberry pi 2

https://hub.docker.com/r/supernisor/armv7-puppeteer/tags



註:puppeteer是一個nodejs上的爬蟲套件，是一個運用chromium browser進行爬蟲的工具。

容器化導入流程 - 從測試到生產環境

Dockerfile建置
CI/CD Pipeline建置
部署服務至測試環境
EKS

壓力測試
CPU,Memory
Monitoring
Log collection
penetration Testing

部署服務至生產環境EKS
Monitoring,Alert

容器化工具 - Docker

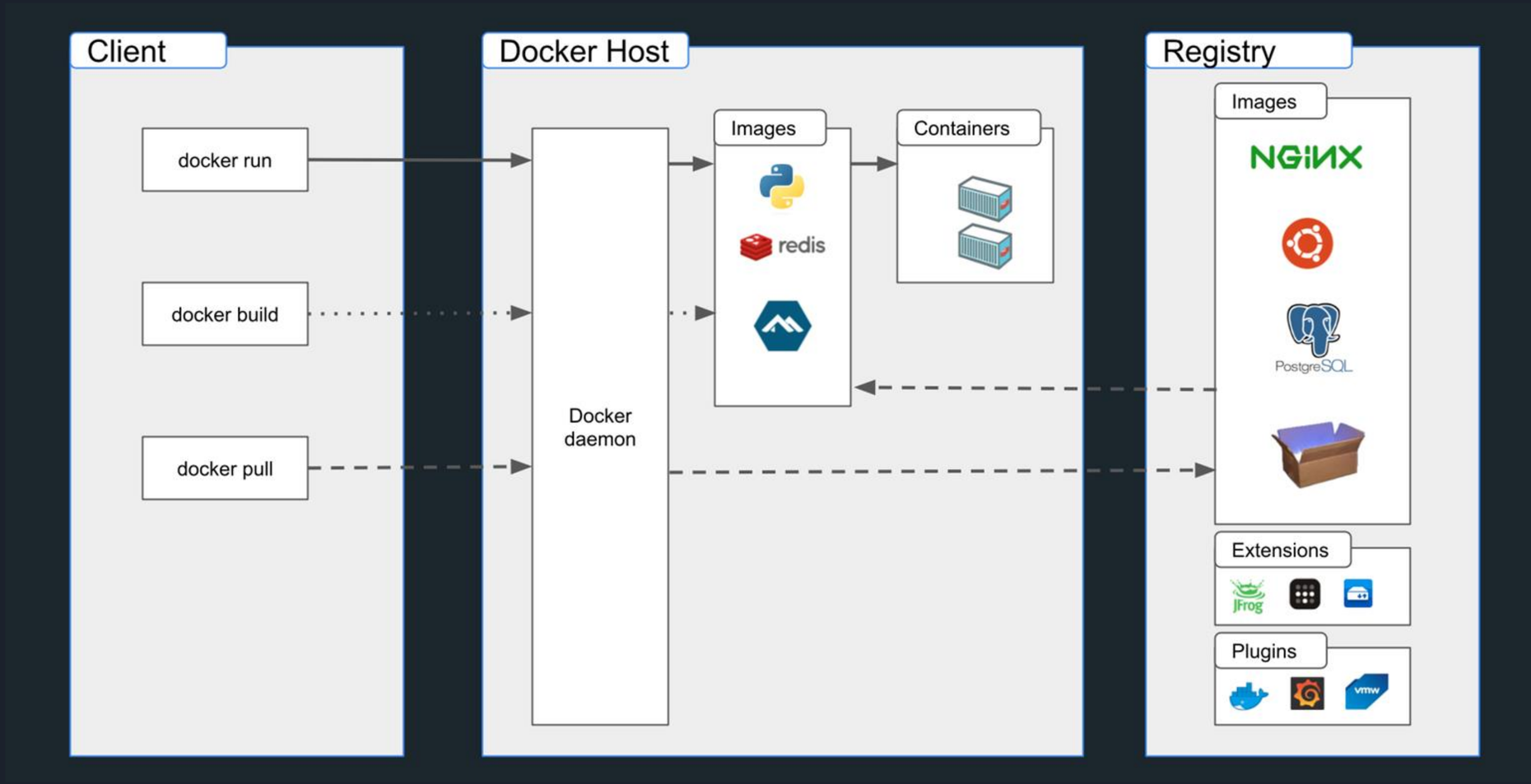
What is a container?

Simply put, a container is a sandboxed process on your machine that is isolated from all other processes on the host machine. That isolation leverages kernel namespaces and cgroups, features that have been in Linux for a long time. Docker has worked to make these capabilities approachable and easy to use. To summarize, a container:

- Is a runnable instance of an image. You can create, start, stop, move, or delete a container using the DockerAPI or CLI.
- Can be run on local machines, virtual machines or deployed to the cloud.
- Is portable (can be run on any OS).
- Is isolated from other containers and runs its own software, binaries, and configurations.

https://docs.docker.com/get-started/

Docker **Architecture**



https://docs.docker.com/get-started/overview/

Dockerfile建置

1.簡單生成一個Dockerfile

```
FROM tomcat:9-jdk17

# copy war
COPY test.war /usr/local/tomcat/webapps/

# run
CMD ["/usr/local/tomcat/bin/catalina.sh","run"]
```

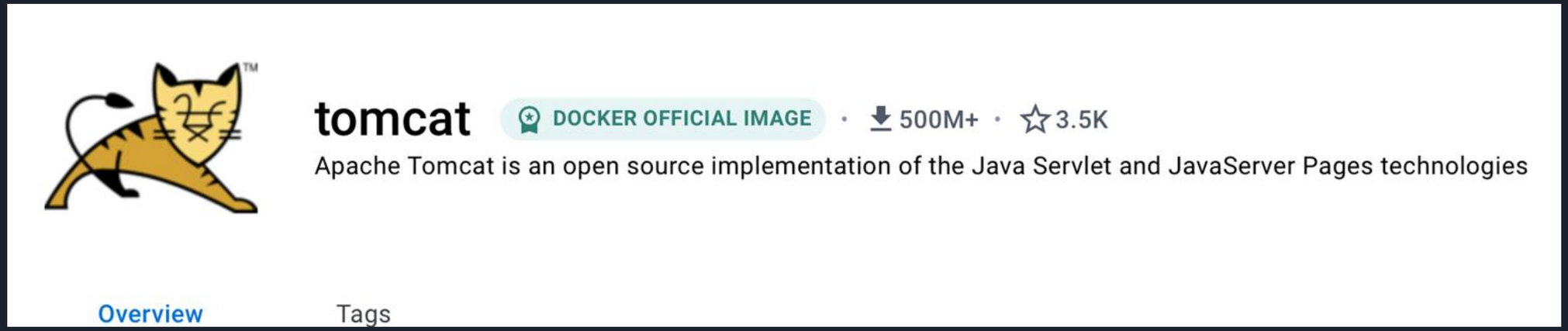
使用Official Image

複製war檔到server目錄

2. 在同一目錄跑docker build -t application:test -f Dockerfile .
3. docker run -itd application:test -p 8080:8080
4. docker ps 可以看容器運作狀態

https://adamsanalysis.com/docker/docker-cheatsheet

Dockerfile建置 - Official Image



https://github.com/docker-library/tomcat/blob/9fd0c865c16751a144216186720dae7cfc9113bd/9.0/jdk17/temurin-jammy/Dockerfile

Dockerfile建置 - non-root

```
FROM tomcat:9-jdk17

# Set a non-root user
RUN useradd -u 1001 -g 0 myuser

# Create a working directory and set the ownership to the non-root user
RUN  chown -R myuser:root /usr/local/tomcat/webapps/

# Copy your application files to the working directory
COPY test.war /usrlocal/tomcat/webapps/

USER myuser
CMD ["/usr/local/tomcat/bincatalina.sh", "run"]
```

關於Docker的基本安全

1. Dockerfile的User不要用root，不然會有許多潛在的Linux提權風險。
2. Docker的port不要對外打開。
3. 不要把config檔案直接放在Dockerfile內。

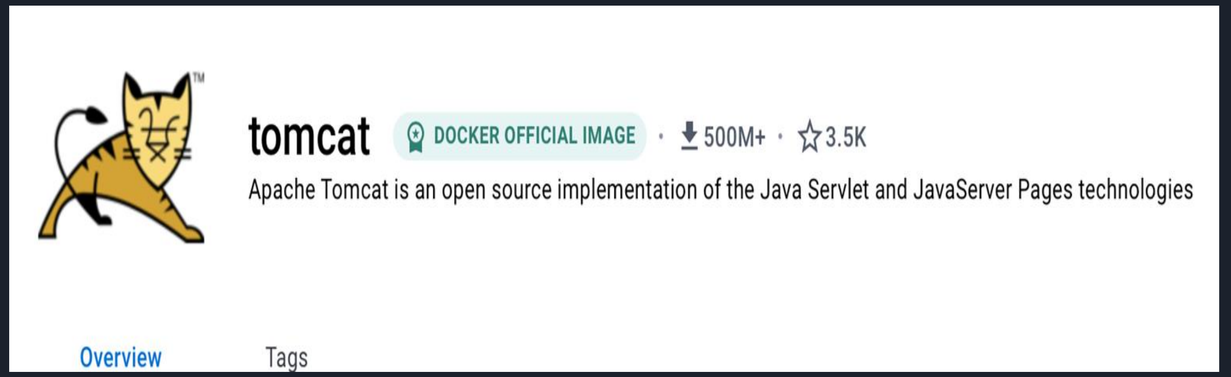
參考Docker安全筆記:
https://github.com/SunWeb3Sec/Kubernetes-security#docker-attacks

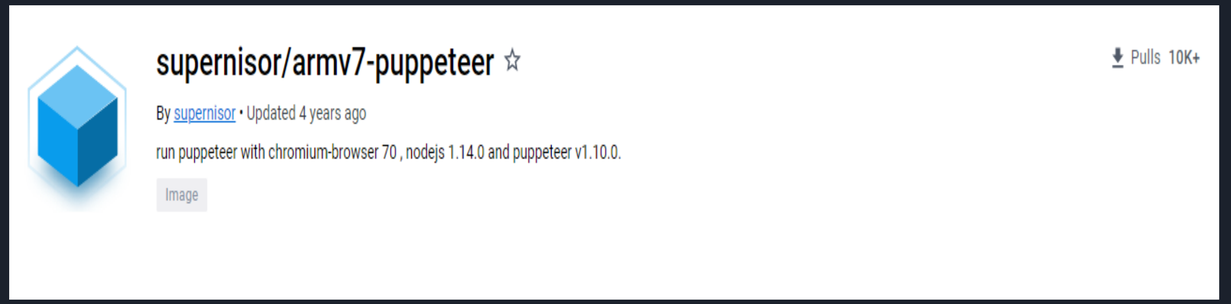
運作Container時

本地環境核心可能與虛擬主機核心不同，導致Container無法正常啟動

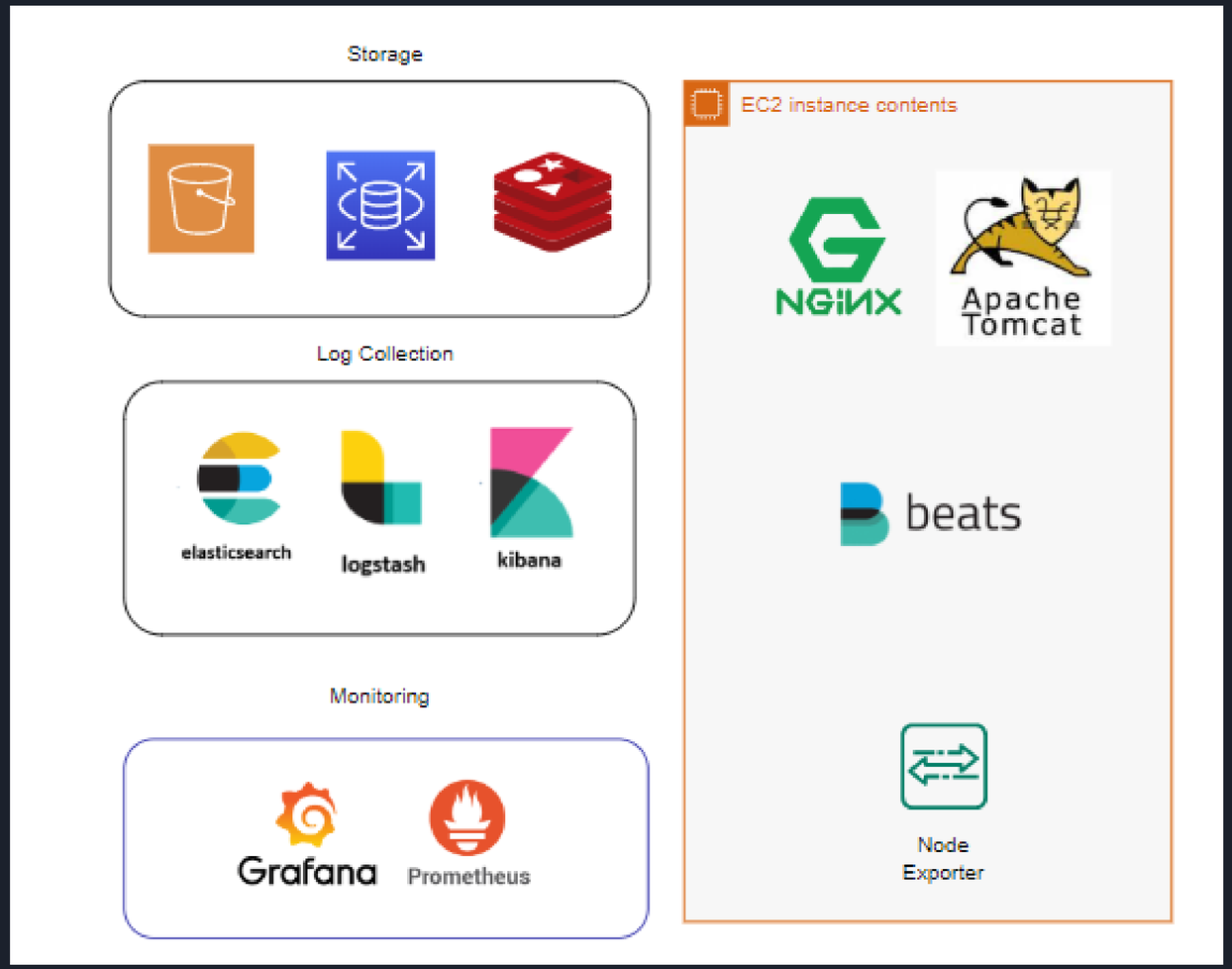
x86

arm

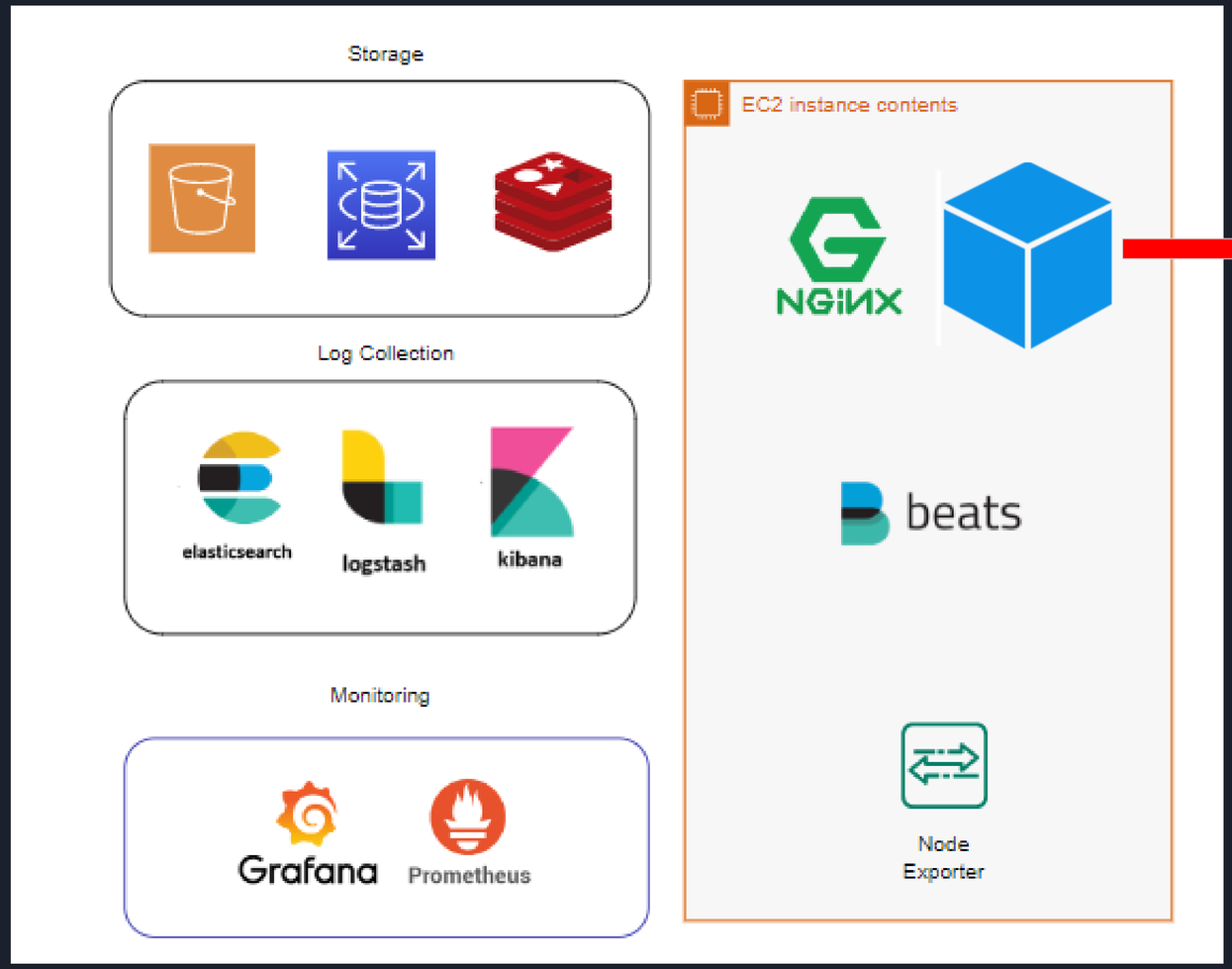




測試環境 - 還沒跑Container

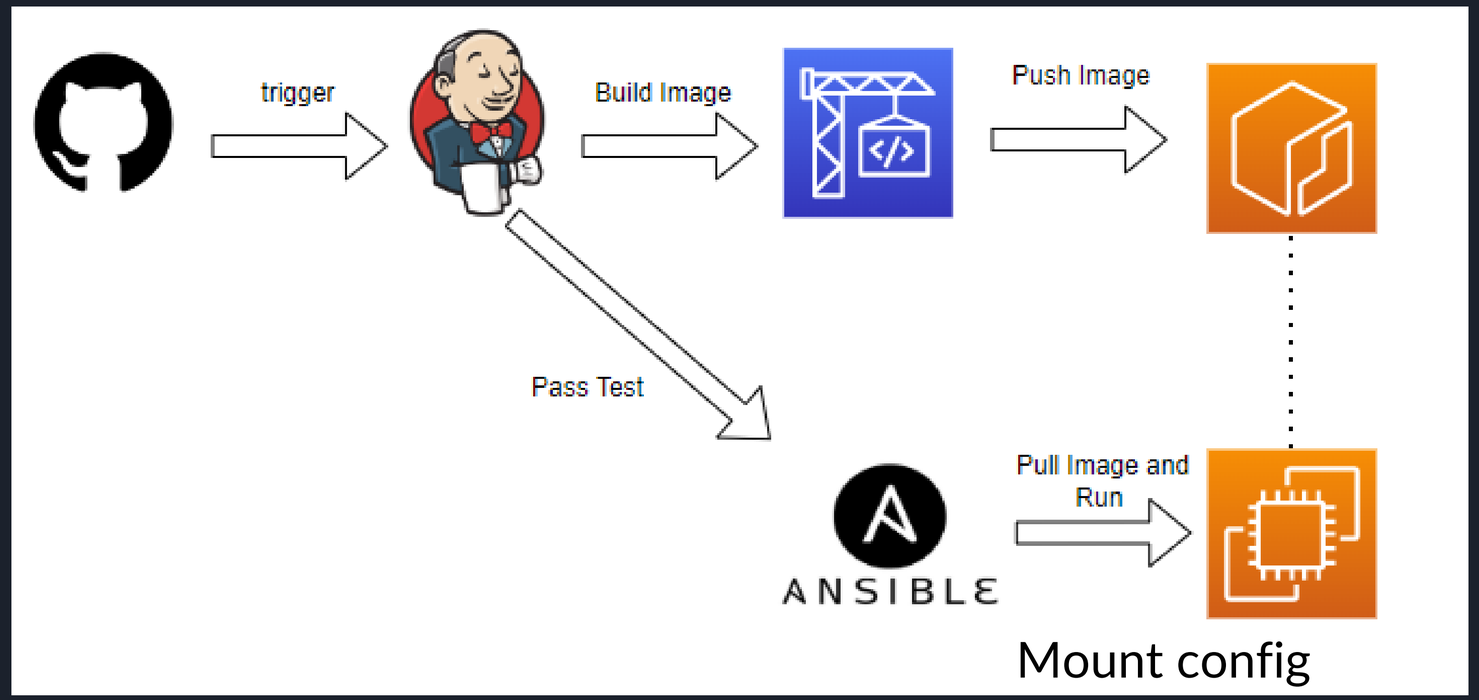


測試環境 - 運作container後



VM的Server被放置
在 Container內執行

CI/CD

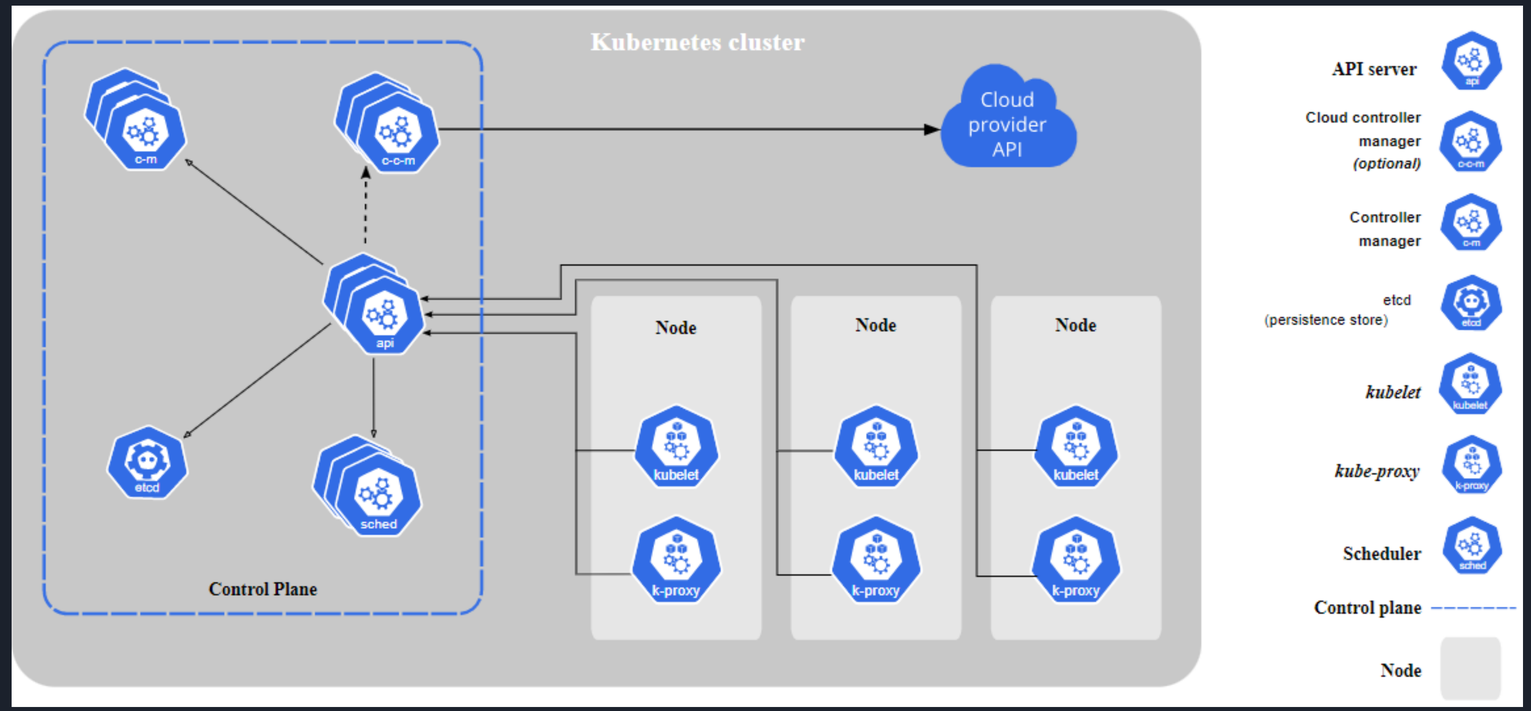

trigger
Build Image
Push Image
Pass Test
Pull Image and Run
ANSIBLE
Mount config

容器化優勢與劣勢

優勢：
1.容易移植 - 可以移植到相同核心不同系統環境上運行。
2.快速部署 - 可以很快建制部署到虛擬主機當中。

劣勢：
1.容器安全性 - 容器化的權限需要特別注意，若設置不當可能會有容器逃逸、數據洩漏等問題。
2.比較高的學習成本 - 容器化的控制元件、工作元件更新的速度很快，維運上會面臨更新、deprecate等問題。

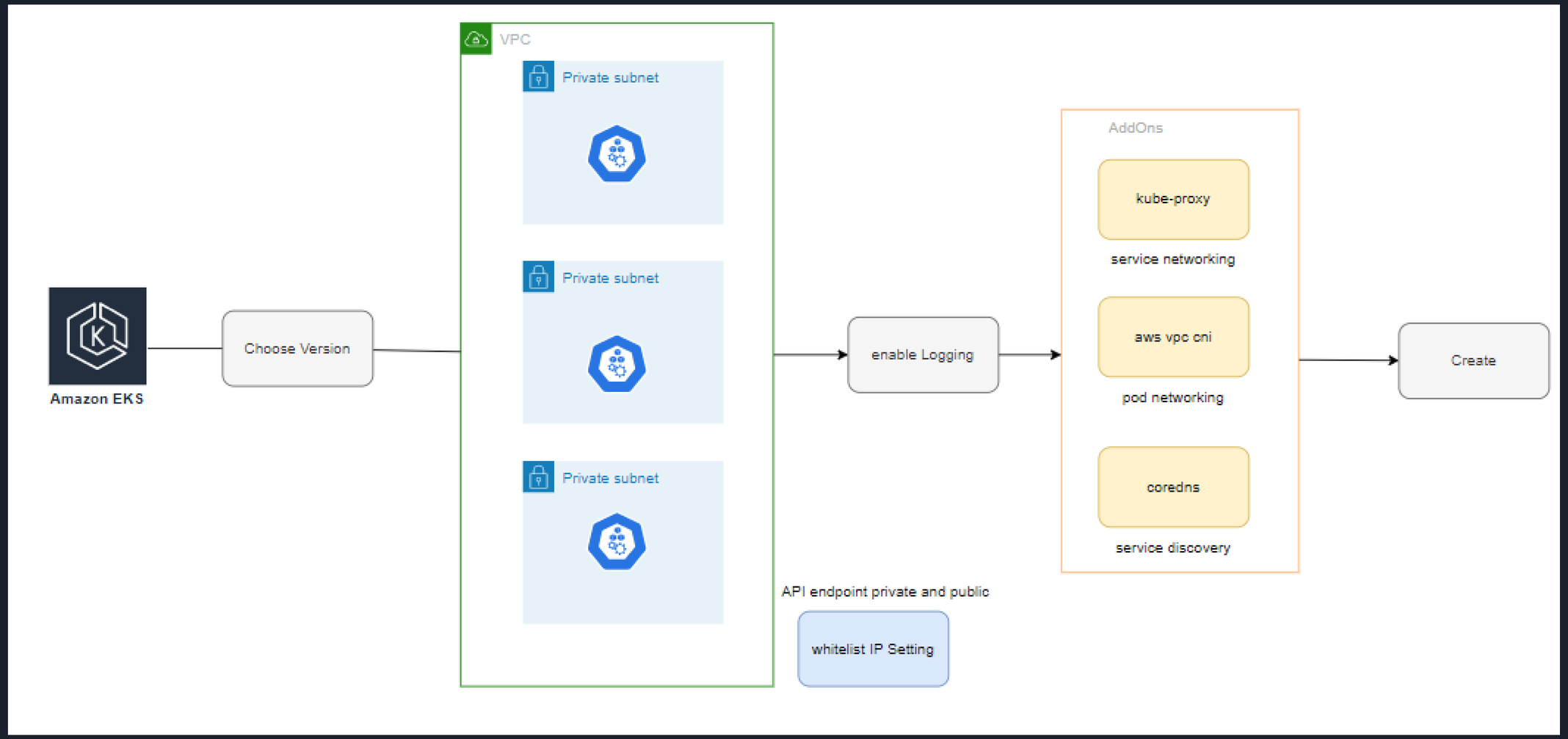
容器化管理工具 - Kubernetes



https://kubernetes.io/docs/concepts/overview/components/

A Kubernetes cluster consists of a set of worker machines, called nodes, that run containerized applications. Every cluster has at least one worker node.

建立EKS Cluster



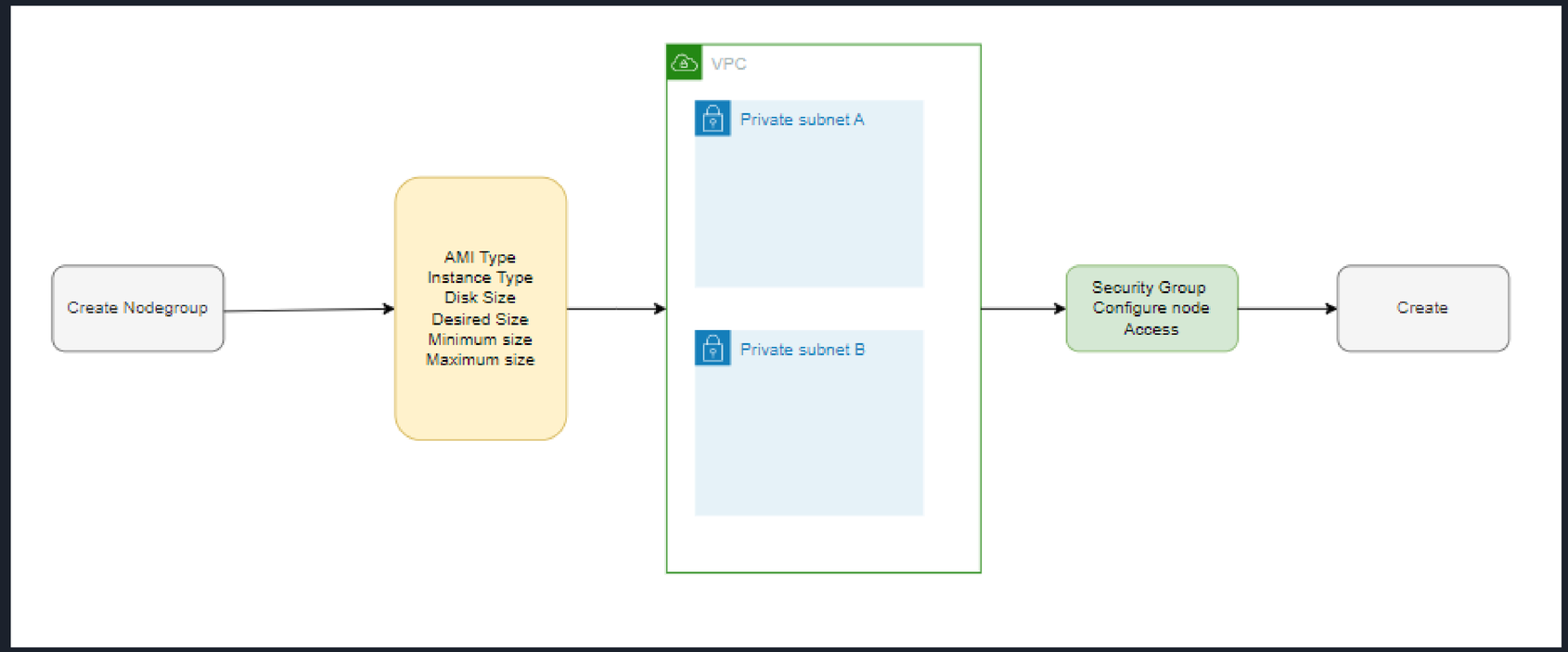
建立EKS Cluster - 注意

VPC網路 : EKS 會透過aws cni plugin進行IP配發，在建立Cluster時，如果服務不多，但有服務要用較大的機器類型運作，會有IP浪費的問題。
https://aws.github.io/aws-eks-best-practices/networking/vpc-cni/

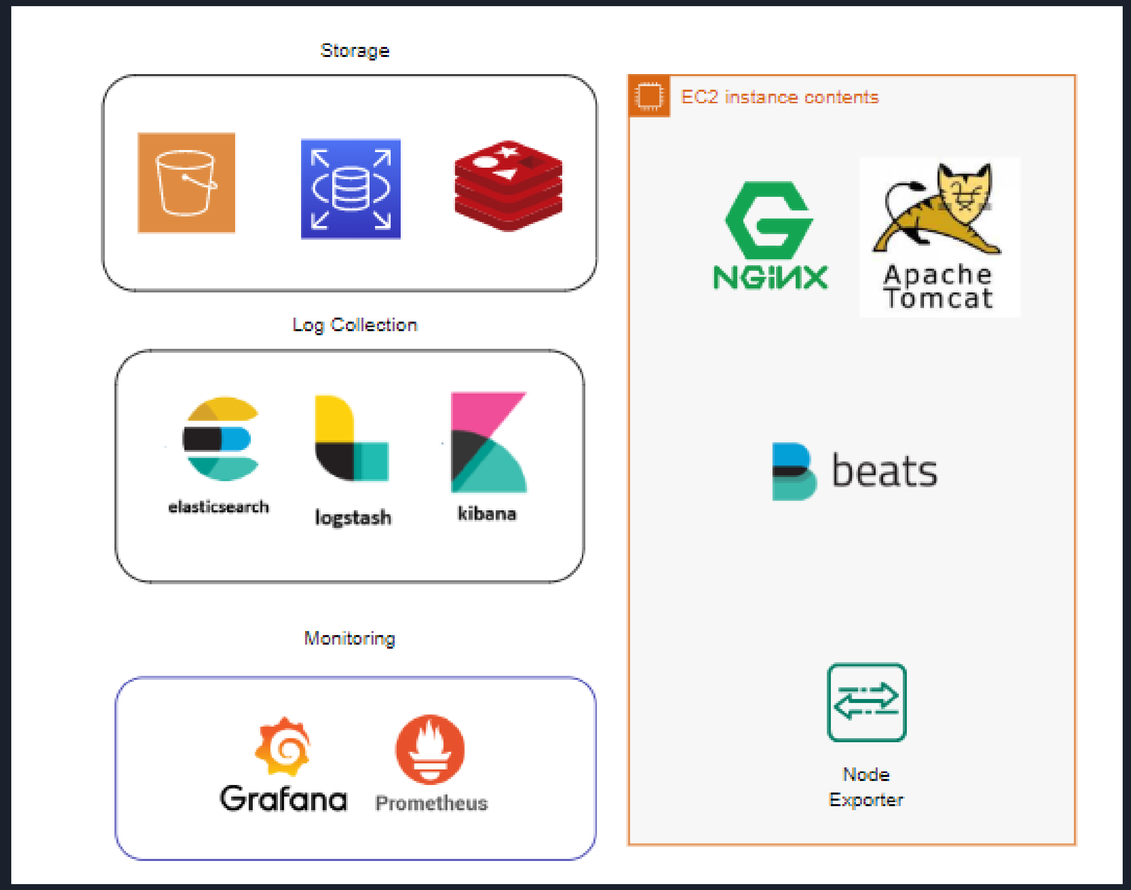
RBAC權限: 有些在AWS上的使用權限可以沿用到Kubernetes上，權限配置可以容易進行，但針對需要使用的AWS資源，需要從IAM建置。(roles 和 rolebindings 或 clusterroles 和 clusterrolebindings)
https://docs.aws.amazon.com/zh_tw/eks/latest/userguide/add-user-role.html

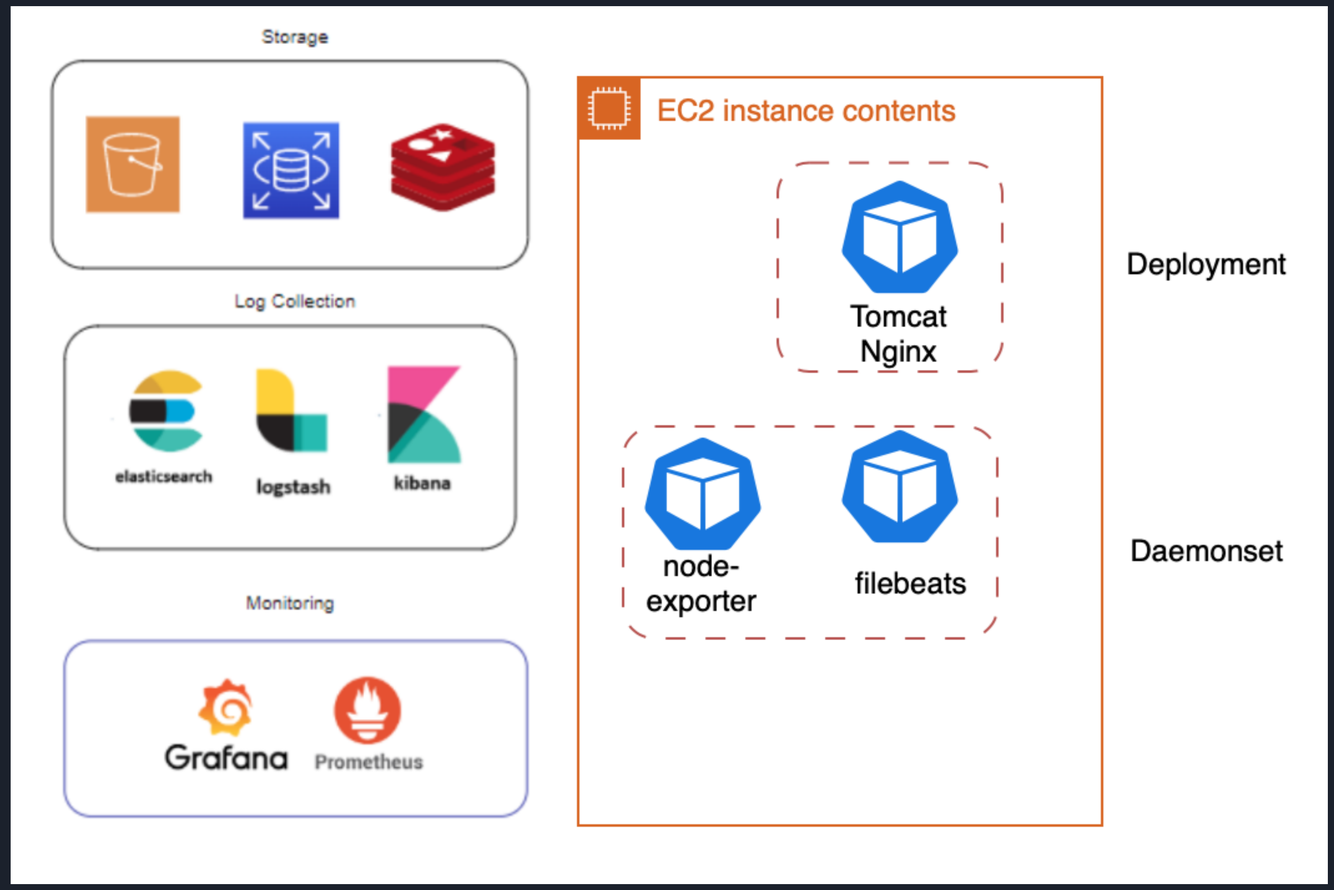
Kubernetes版本：Kubernetes版本迭代很快，版本如果過舊，有些功能可能會無法使用。
https://kubernetes.io/blog/2020/12/02/dont-panic-kubernetes-and-docker/

AWS EKS - 建立NodeGroup

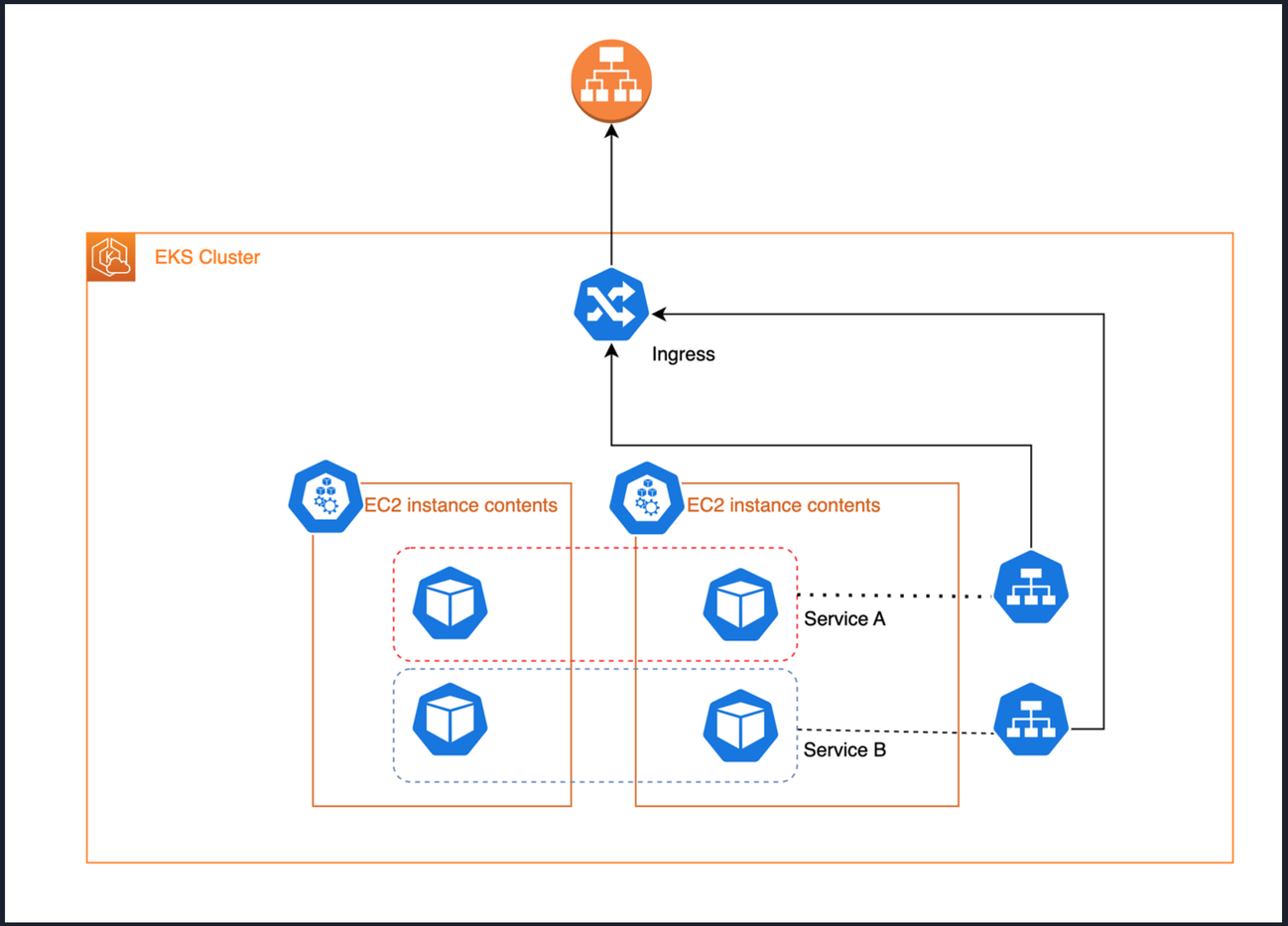


容器化


Storage
Log Collection
elasticsearch
logstash
kibana
Monitoring
Grafana
Prometheus
EC2 instance contents
NGINX
Apache Tomcat
beats
Node Exporter

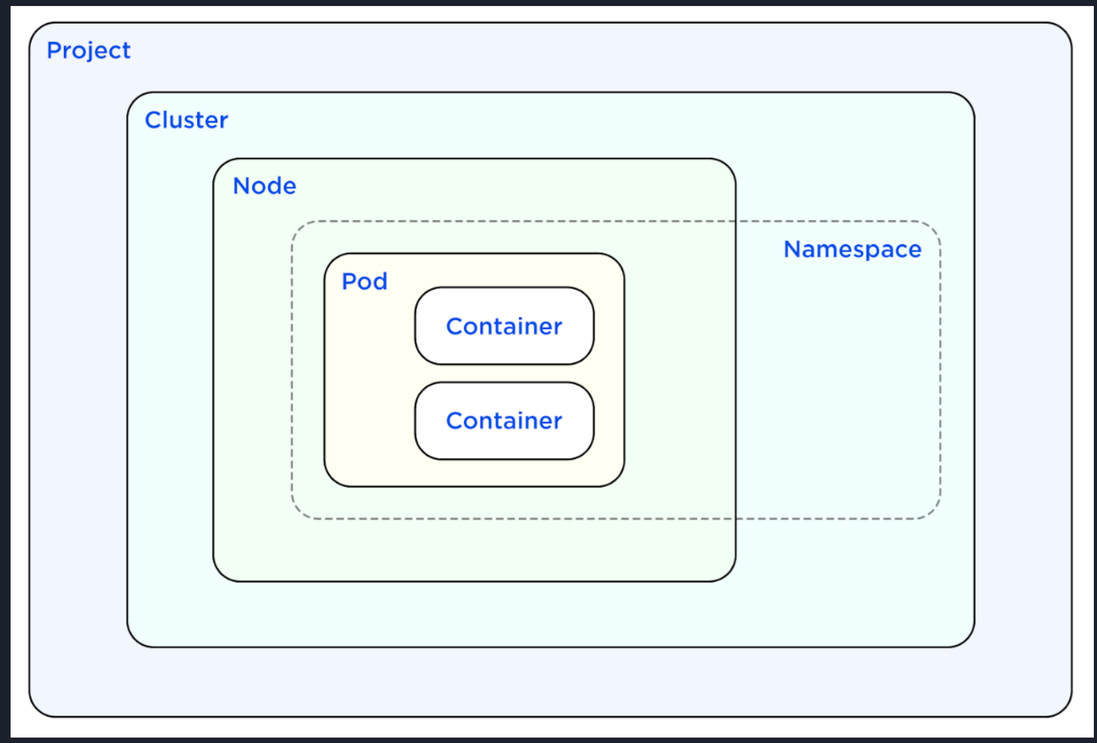

Storage
Log Collection
elasticsearch
logstash
kibana
Monitoring
Grafana
Prometheus
EC2 instance contents
Tomcat
Nginx
node-
exporter
filebeats
Deployment
Daemonset

EKS Cluster - 基本規格建置



POD、Node、Container

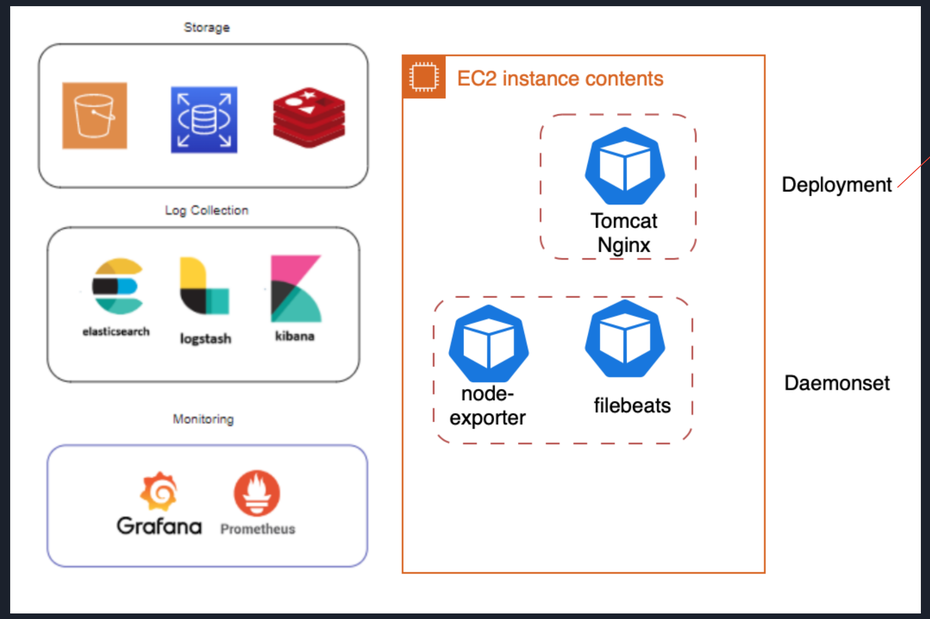
Node : 虛擬機器(VM)
Pod:乘載數個容器的元件
Container: 運作程式的容器單位



https://www.armosec.io/blog/kubernetes-security-best-practices/

Deployment

Container - tomcat & nginx
- 同一個Pod內，connection屬於localhost。
- replicas可以設成2以上for scale。
- 透過nginx container forward port 8080。
- mount nginx config給nginx container使用。




```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tomcat-nginx-deployment
  labels:
    app: tomcat-nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: tomcat-nginx
  template:
    metadata:
      labels:
        app: tomcat-nginx
    spec:
      containers:
      - name: tomcat
        image: tomcat:9.0
        ports:
        - containerPort: 8080
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
        volumeMounts:
        - name: nginx-config
          mountPath: /etc/nginx/nginx.conf
          subPath: nginx.conf
      volumes:
      - name: nginx-config
        configMap:
          name: nginx-configmap
```

ConfigMap - nginx

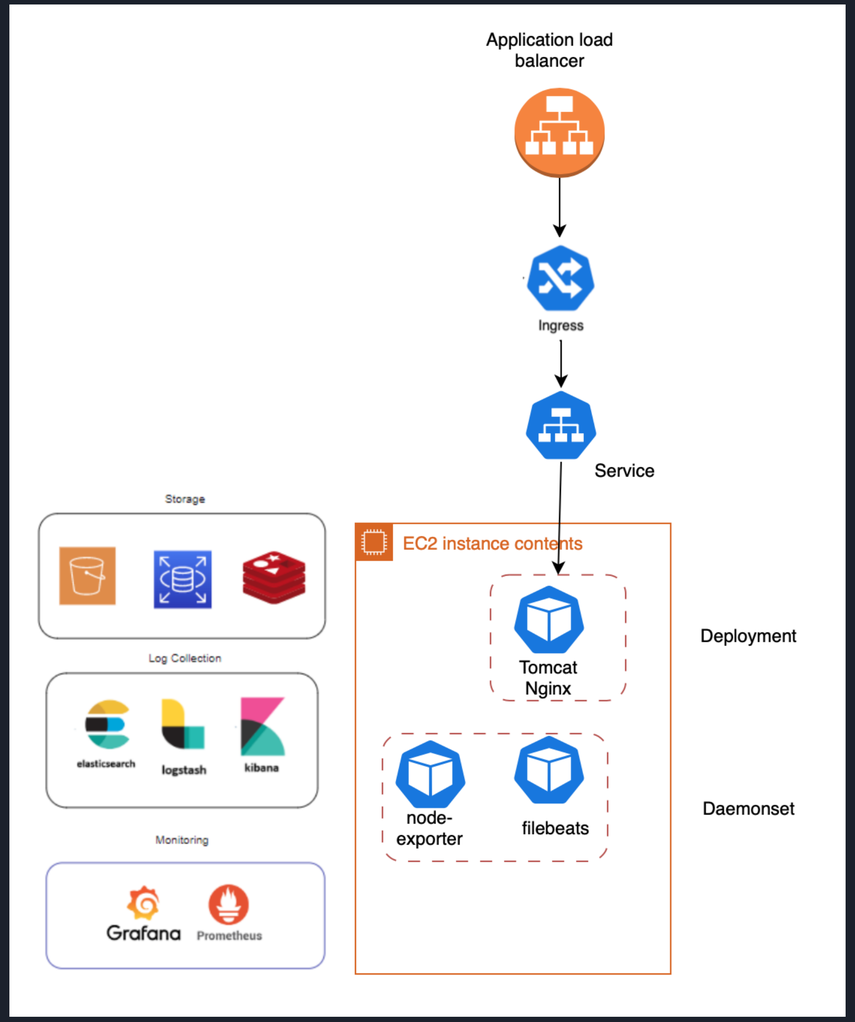
Nginx config檔

掛載給nginx使用。

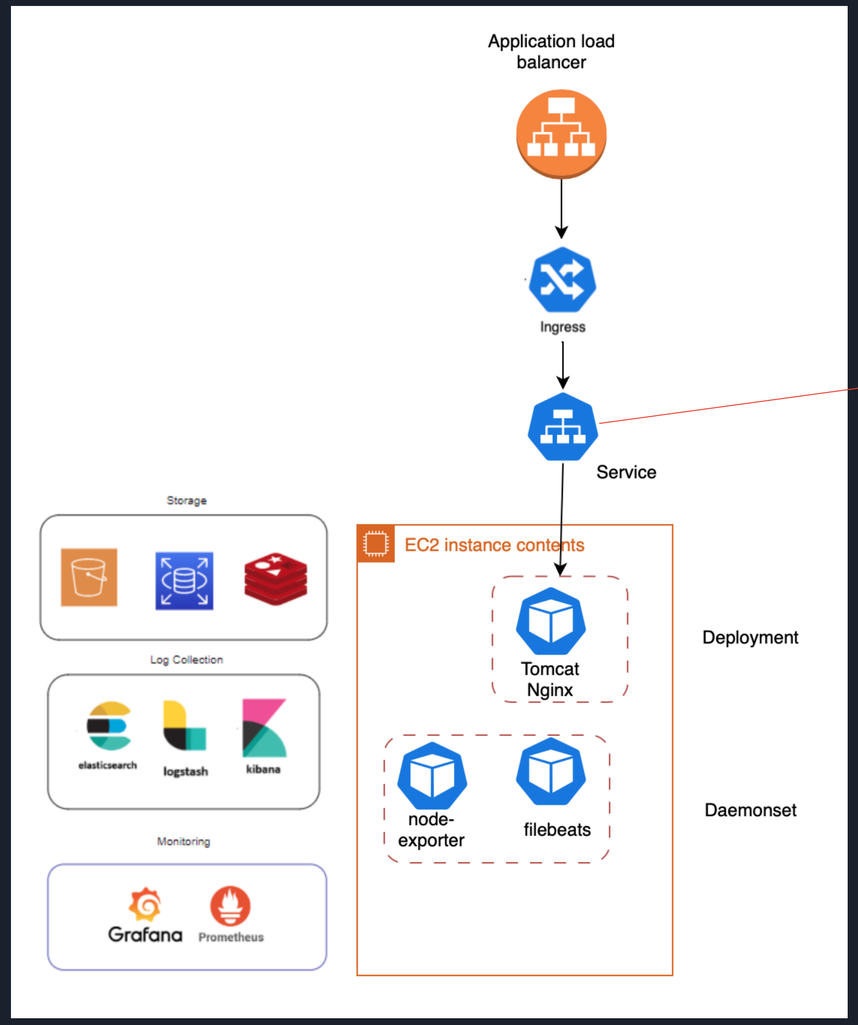
```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: nginx-configmap
data:
  nginx.conf: |
    events {
      worker_connections  1024;
    }

    http {
      server {
        listen 80;
        server_name localhost;

        location / {
          proxy_pass http://localhost:8080/;
          proxy_set_header Host $host;
          proxy_set_header X-Real-IP $remote_addr;
          proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
          proxy_set_header X-Forwarded-Proto $scheme;
        }
      }
    }
```

容器化架構 - 更多元件組合

Kubernetes可以透過Service、Ingress和ALB Controller，進行對外連線。

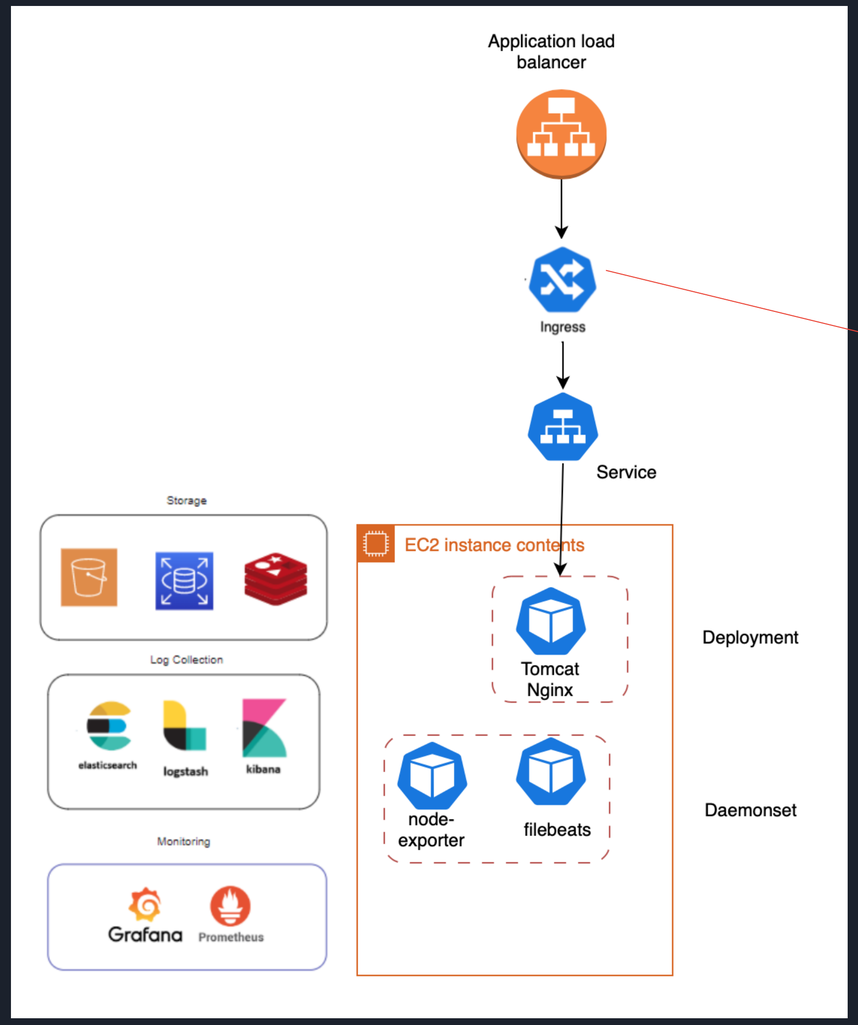


Service




```yaml
apiVersion: v1
kind: Service
metadata:
  name: tomcat-service
spec:
  selector:
    app: tomcat
  ports:
    - name: http
      port: 8080
      targetPort: 8080
  type: ClusterIP
```

Ingress



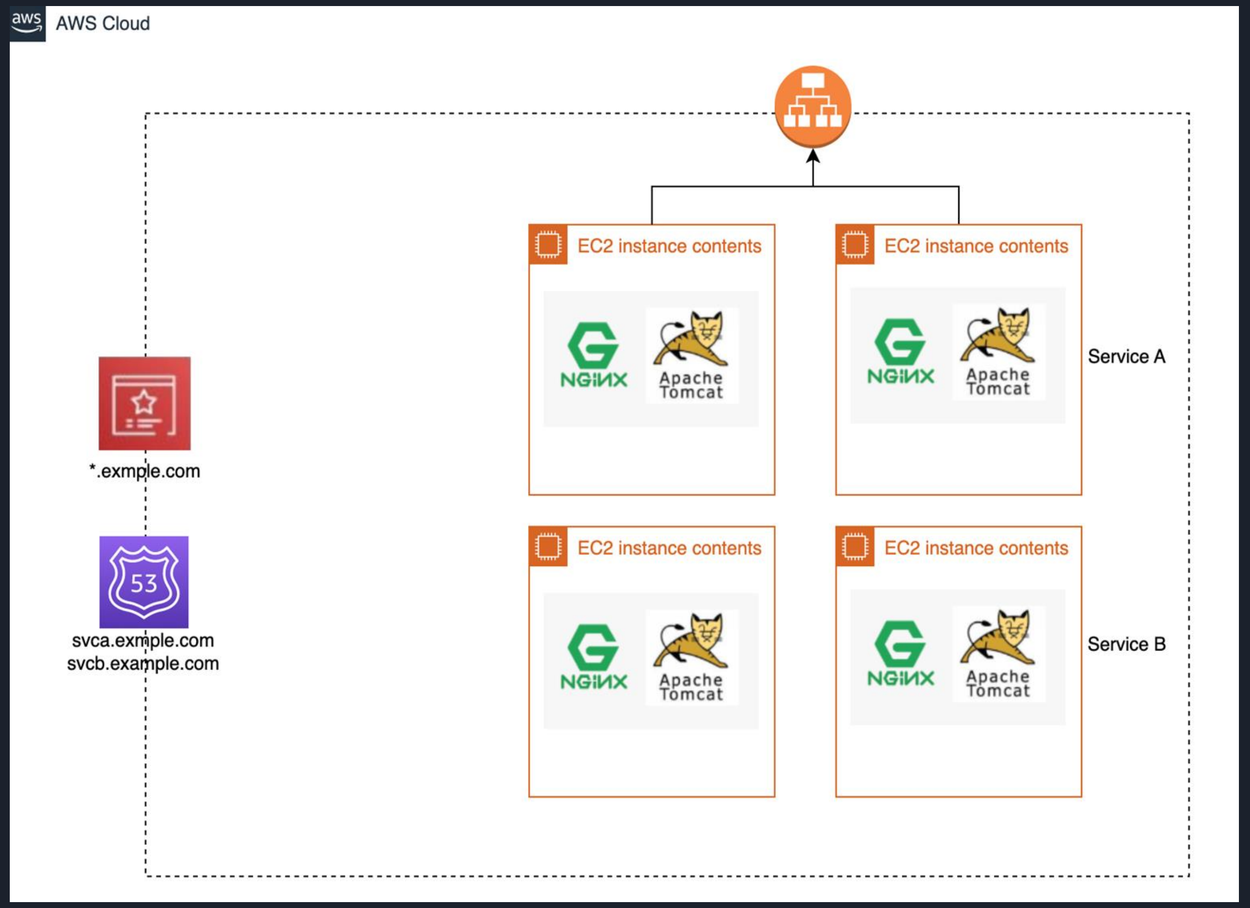

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: tomcat-ingress
  annotations:
    kubernetes.io/ingress.class: alb
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80}, {"HTTPS": 443}]'
    alb.ingress.kubernetes.io/actions.ssl-redirect: |
      {
        "type": "redirect",
        "redirectConfig": {
          "protocol": "HTTPS",
          "port": "443",
          "statusCode": "HTTP_301"
        }
      }
spec:
  rules:
    - host: svc1.example.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: tomcat-service
                port:
                  name: http
```

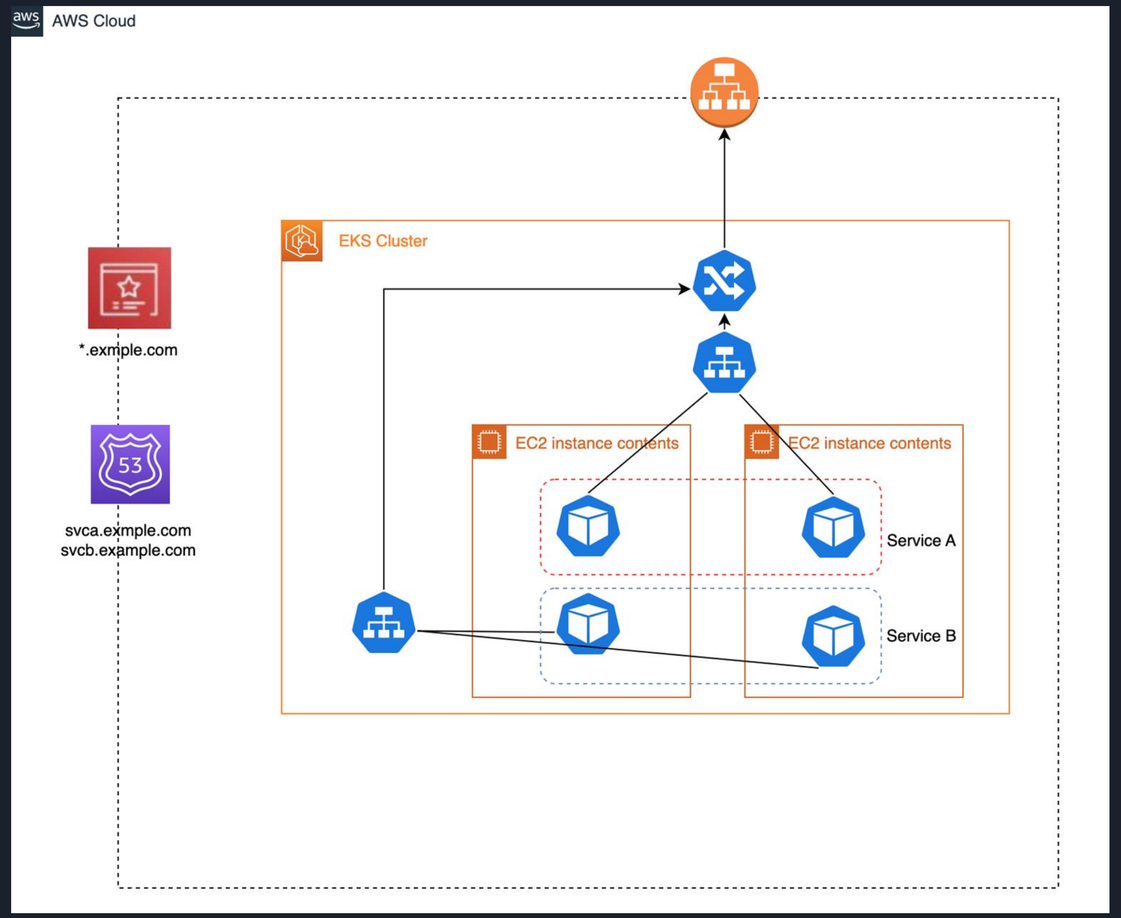
Kubectl - 與Kubernetes 交互指令

kubectl是一個命令行接口，用於對Kubernetes集群運行命令，完成對k8s集群連接、查看資源、部署等基本操作。

https://kubernetes.io/docs/reference/kubectl/cheatsheet/

轉容器化（多個Service）





技術困難 - 太多know how

需要用到什麼學什麼
- Application - Pod,Deployment, Daemonset,ConfigMap
- Network - Ingress,Service

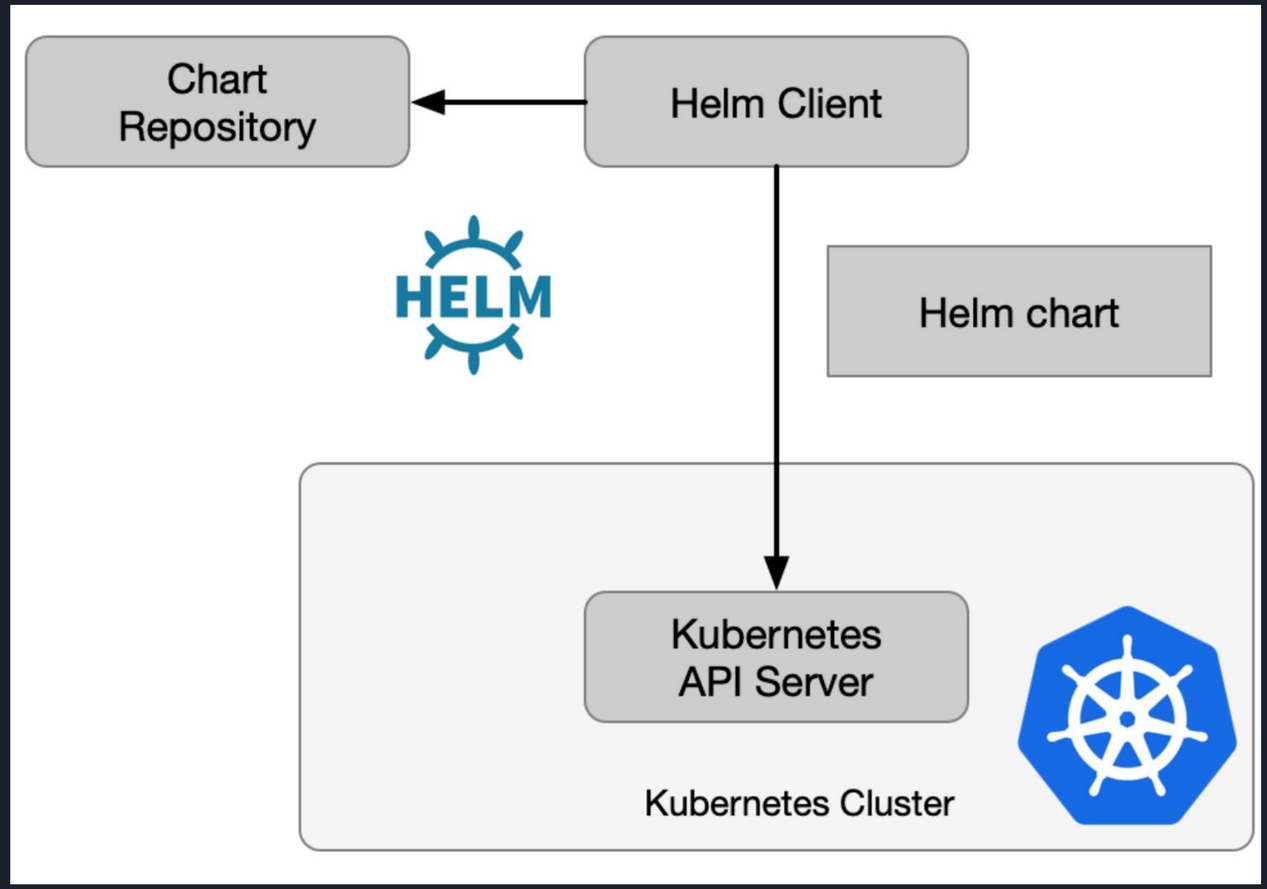
https://kubernetes.io/docs/concepts/workloads/pods/
https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/
https://kubernetes.io/docs/concepts/workloads/controllers/deployment/
https://kubernetes.io/docs/concepts/services-networking/ingress/
https://kubernetes.io/docs/concepts/services-networking/service/
75個常見問題和答案 https://medium.com/@bubu.tripathy/top-75-kubernetes-questions-and-answers-d677a0b87d79

K8s物件管理工具 - Helm

Helm 是管理 Kubernetes yaml的工具，
用來整合許多Kubernetes物件的運作。

由於許多開源工具都有用Helm來打包部署k8s，Helm是幾乎是必學的管理工具。

https://helm.sh/zh/docs/intro/quickstart/



来自 IBM Developer Blog

工具 - Monitor, Log Collection

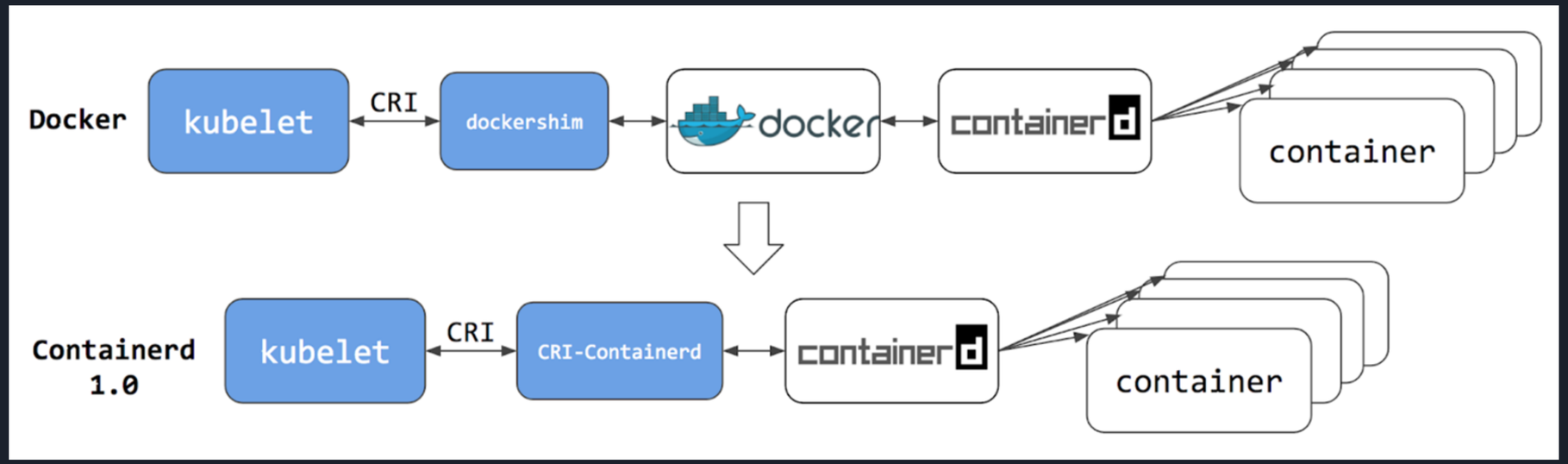
Monitoring (Prometheus,Grafana,Node-exporter)
kube-prometheus-stack: https://github.com/prometheus-community/helm-charts/tree/main/charts/kube-prometheus-stack

Log Collection (Filebeat):
https://www.elastic.co/guide/en/beats/filebeat/current/running-on-kubernetes.html
AWS Load balancer controller:
https://docs.aws.amazon.com/zh_tw/eks/latest/userguide/aws-load-balancer-controller.html

External-DNS:
https://github.com/kubernetes-sigs/external-dns
Cluster AutoScaler
https://docs.aws.amazon.com/zh_tw/eks/latest/userguide/autoscaling.html

額外補充 - Kubernetes棄用Dockershim

在EKS v1.24之後，dockershim被棄用，仍然可以使用docker build指令進行建置，主要影響
是在Node裡下docker指令不會看到在kubernetes上執行的。
https://docs.aws.amazon.com/zh_tw/eks/latest/userguide/dockershim-deprecation.html
在Node如果要看容器運作等資訊，docker指令改用crictl。
https://kubernetes.io/docs/reference/tools/map-crictl-dockercli/



参考資料

Docker:
https://docs.docker.com/get-started/
Kubernetes :
https://kubernetes.io/
Helm:
https://helm.sh/
Kubernetes Best Practices
https://www.armosec.io/blog/kubernetes-security-best-practices/

XREX

THANK YOU!