~~AWS CDK 推坑大會~~

堆疊架構時該思考的事

思考的點

- 如何把 CDK 導入至現有的環境？
- 如何對 CDK 做模組化？
- 如何解決 Stack 跟 Stack 間的協調問題？

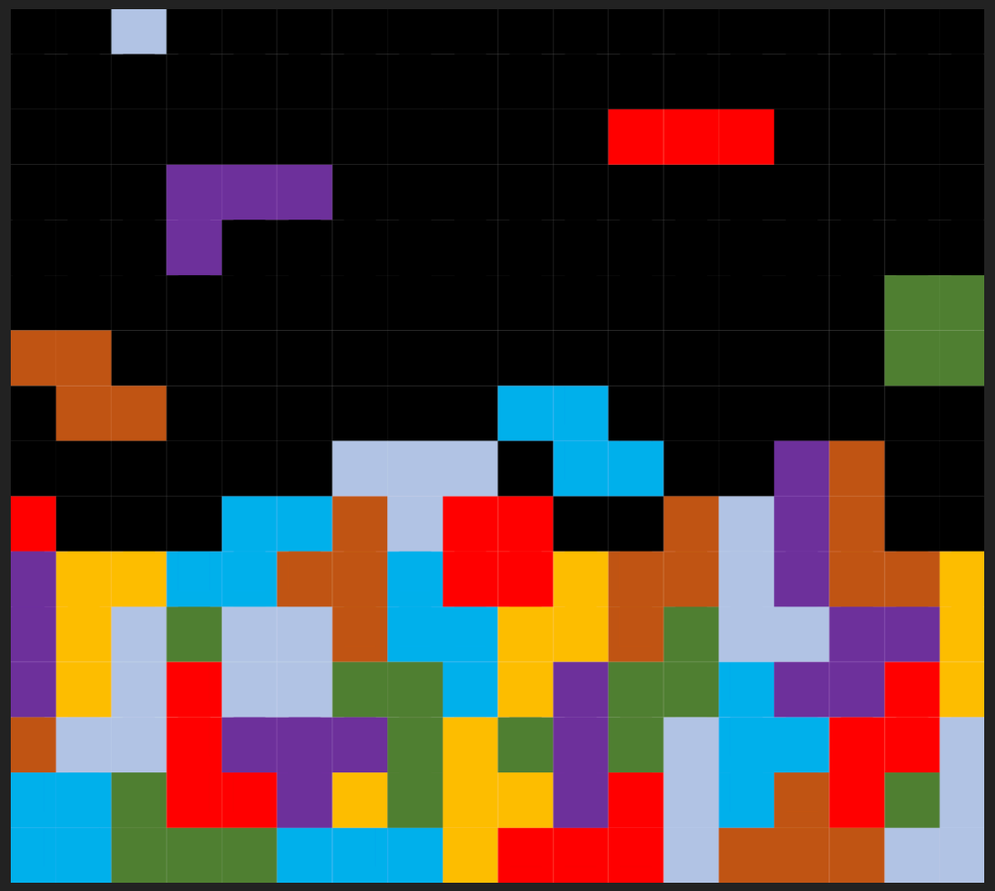
如何把 CDK 導入至現有的環境？

問題點

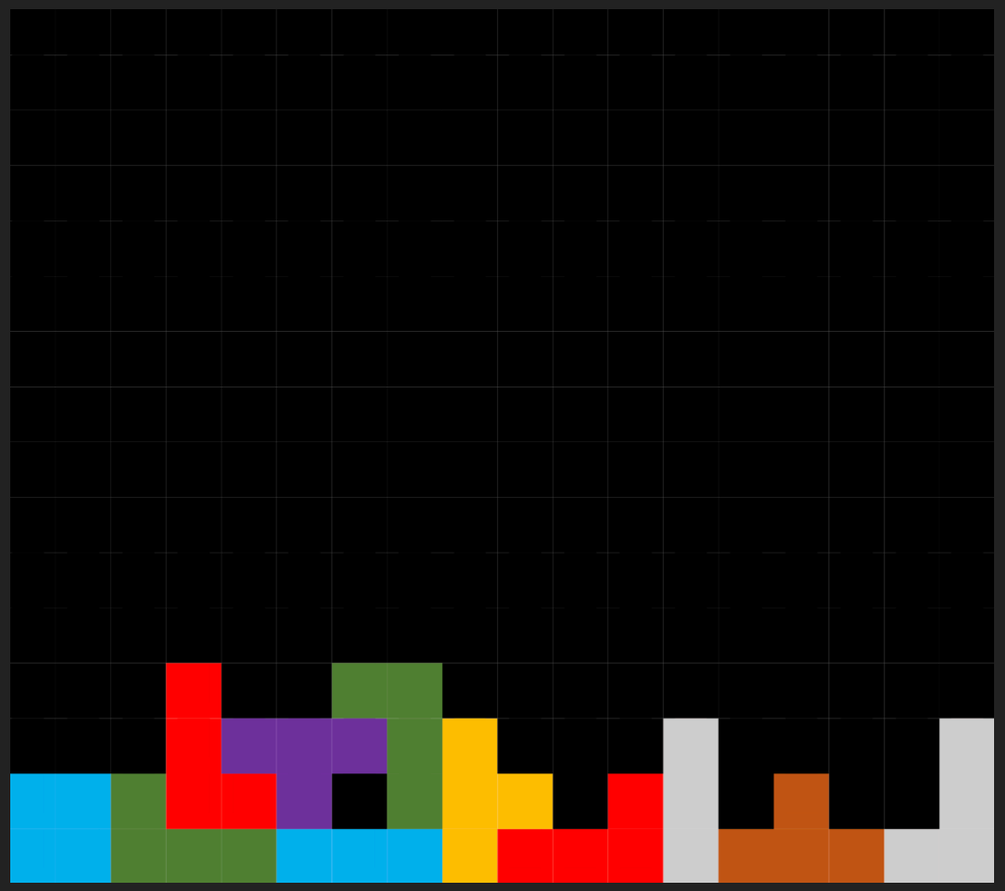
- 可能 Resource 原先不是透過 Cloudformation 建立
- 可能不同 Region, Resource 的長相或配置會不一樣
- 可能又同時存在 Cross Account 跟 Cross Region
- 短期內公司不鼓勵低商業價值的重建
  - 伴隨高風險、低效益

現有環境的真實樣貌

Region A



Region B



My Approach

可能的解法

- 可以找一個空的 Region 直接導入 CDK
- 設計 Config
- 針對每一個 AWS Resource 作開關
- 擴充 CDK Props Interface 來達到目的
- 開關範圍視情境設計
- 須思考如何同時支援跨帳號、跨區域

可能的解法

- 可以找一個空的 Region 直接導入 CDK
- 設計 Config
- 針對每一個 AWS Resource 作開關
- 擴充 CDK Props Interface 來達到目的
- 開關範圍視情境設計
- 須思考如何同時支援跨帳號、跨區域

可以找一個尚未建立任何東西的 Region

可能的解法

- 可以找一個空的 Region 直接導入 CDK
- 設計 Config
- 針對每一個 AWS Resource 作開關
- 擴充 CDK Props Interface 來達到目的
- 開關範圍視情境設計
- 須思考如何同時支援跨帳號、跨區域

擴充 CDK 的 Property

```ts
import * as s3 from '@aws-cdk/aws-s3';

export interface MyBucketProps extends s3.BucketProps {
  establish: boolean;
}
```

設計一個開關

每一個 AWS Resource 都想像成俄羅斯方塊

```
s3: {
  imageBucket: {
    establish: false,
    bucketName: `my-images`,
    versioned: false,
    encryption: s3.BucketEncryption.UNENCRYPTED,
    removalPolicy: cdk.RemovalPolicy.RETAIN,
    publicReadAccess: false,
    blockPublicAccess: new s3.BlockPublicAccess({
      blockPublicAcls: true,
      blockPublicPolicy: true,
      restrictPublicBuckets: true,
      ignorePublicAcls: true,
    }),
  },
  kopsBucket: {
    establish: true,
    bucketName: `my-kops-state`,
    versioned: true,
    encryption: s3.BucketEncryption.UNENCRYPTED,
    removalPolicy: cdk.RemovalPolicy.RETAIN,
    publicReadAccess: false,
    blockPublicAccess: new s3.BlockPublicAccess({
      blockPublicAcls: true,
      blockPublicPolicy: true,
      restrictPublicBuckets: true,
      ignorePublicAcls: true,
    }),
  },
},
```

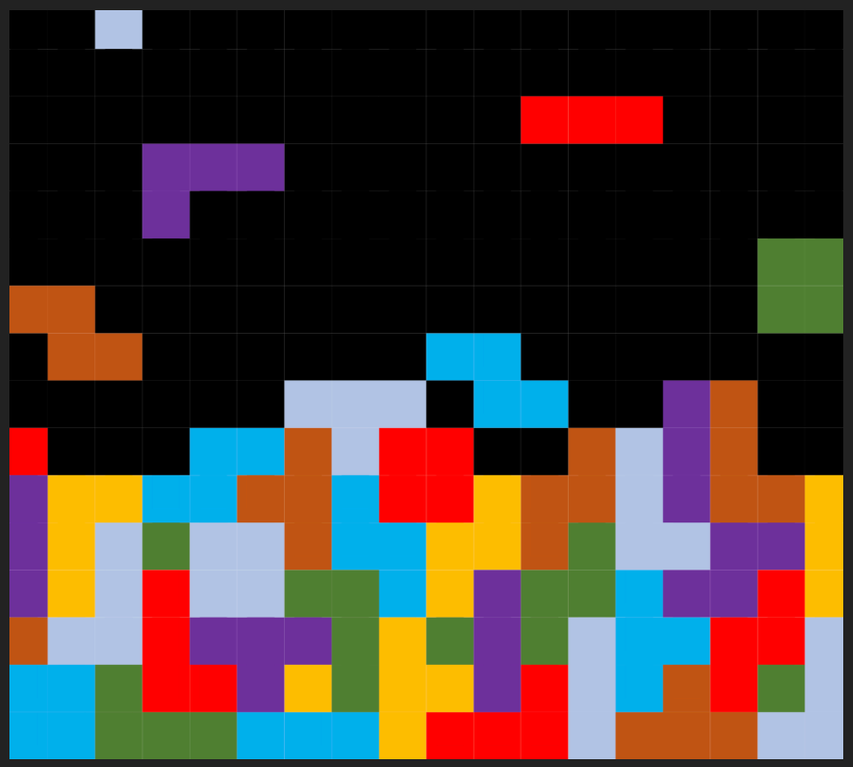
依照現實環境設定

透過 CDK 你可以任一堆疊你想像中的樣貌

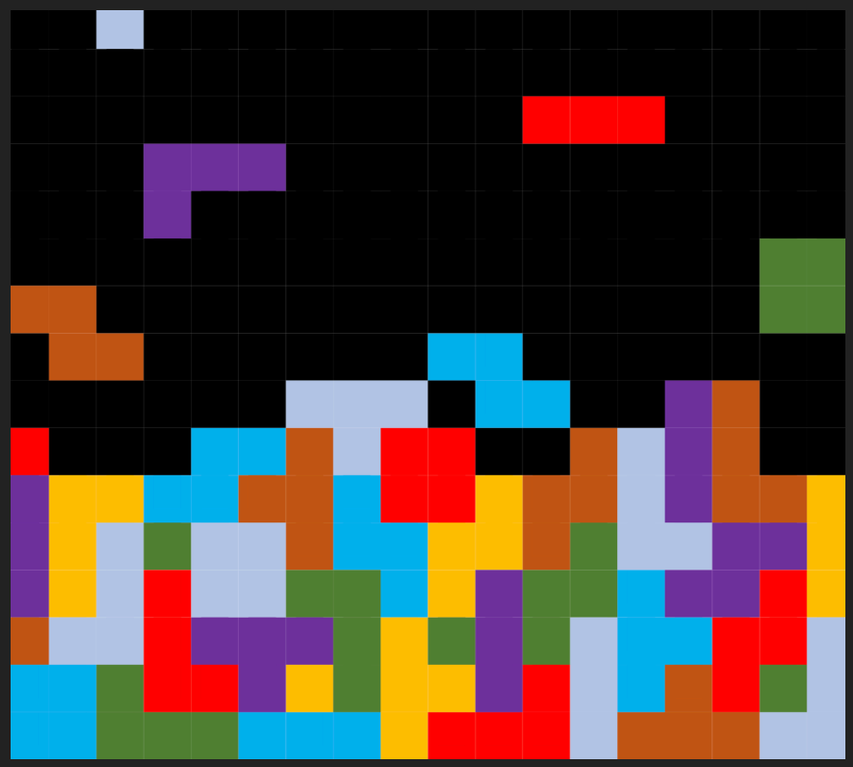
```
export class MyBucketStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props: MyBucketStackProps) {
    super(scope, id, props);

    Object.values(props.s3Props)
      .filter((bucketProps: MyBucketProps) => bucketProps.establish)
      .forEach((bucketProps: MyBucketProps) => {
        const bucket = new s3.Bucket(this, bucketProps.bucketName!, bucketProps);
      });
  }
}
```

最終所有 Region 環境一致

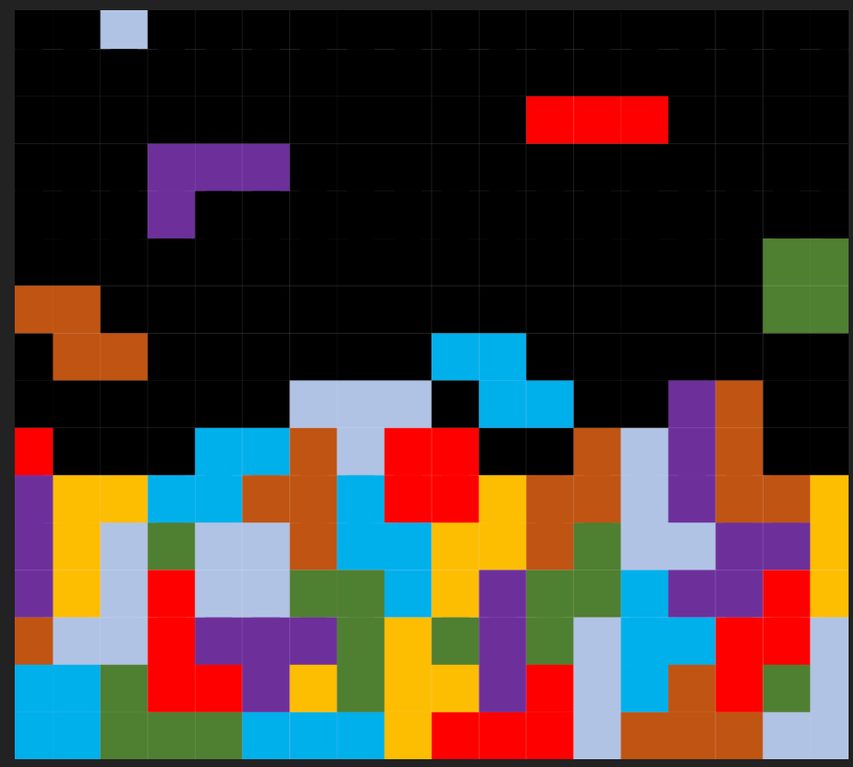
Region A



Region B



New Region



如何把 CDK 導入至現有的環境

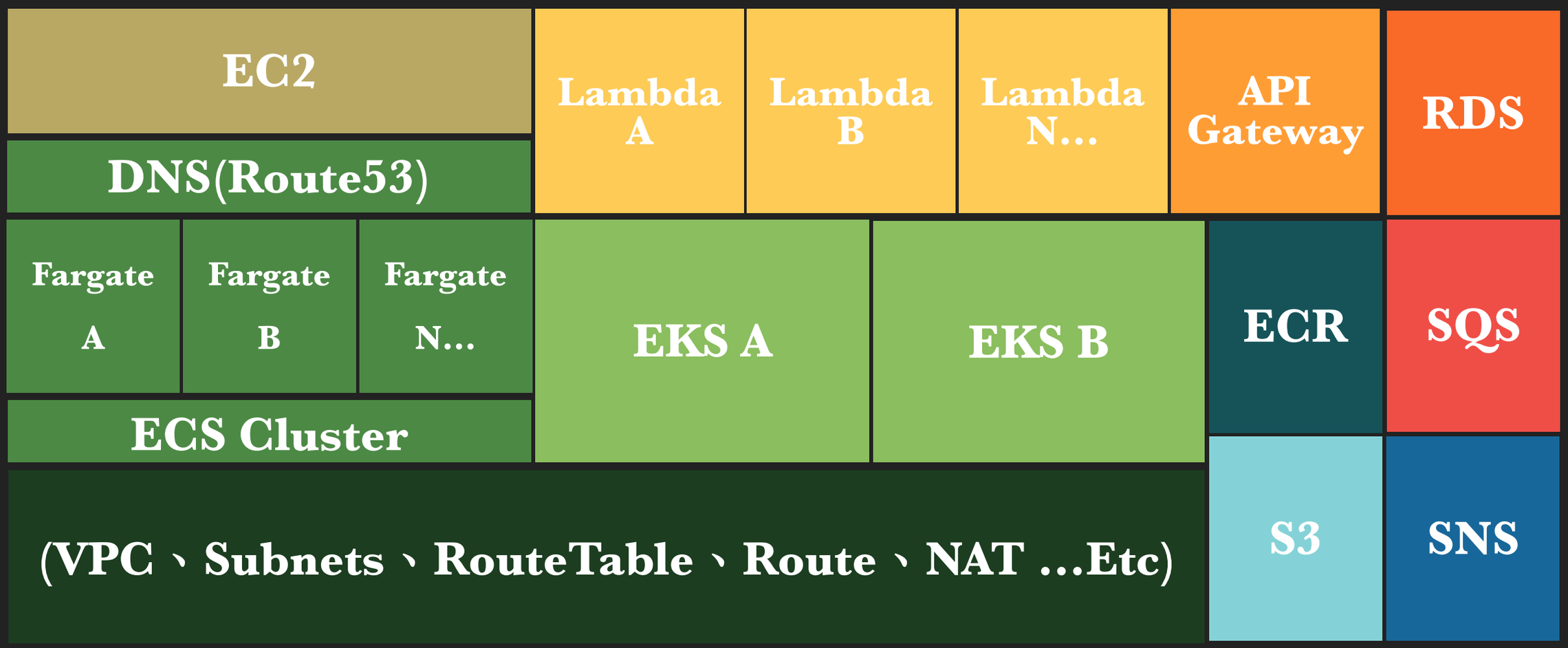
同時你的 DR(災難復原) 也完成了

如何對 CDK 做模組化？

可能的解法

- 必須要盤點現有的 Resource
- 歸納模組的類型
  - Network Stack
  - EKS Stack
  - Microservice Stacks
    - Fargate Service + Codebuild / Codepipeline
  - Route53 Stack
  - etc...
- 每個 CDK Stack 盡可能最小化
  - 搭配 Config 開關堆疊出跟原 Region 一樣的環境

盤點現有的 Resource 並歸納起來, 作模組的依據



可能的解法

- 必須要盤點現有的 Resource
- 歸納模組的類型
  - Network Stack
  - EKS Stack
  - **Microservice Stacks**
    - **Fargate Service + Codebuild / Codepipeline**
  - Route53 Stack
  - etc...
- 每個 CDK Stack 盡可能最小化
  - 搭配 Config 開關堆疊出跟原 Region 一樣的環境

Microservice 在 CDK 上的問題

- Microservices 有 N 組以上, 可能 50 組、100 組 .... 以上
- 每組 Microservices CDK Code 長得很像、卻又不一樣
- 不希望有大量重複的 Code, 但又希望每組 Service 保有客製彈性
- Cloudformation 單一 Stack 有 200 個 Resource 的限制

My Approach

Microservice 在 CDK 上的問題

- Microservices 有 N 組以上, 可能 50 組、100 組 .... 以上
- 每組 Microservices CDK Code 長得很像、卻又不一樣
- 不希望有大量重複的 Code, 但又希望每組 Service 保有客製彈性
- Cloudformation 單一 Stack 有 200 個 Resource 的限制

CDK 比 Cloudformation 更好的地方是

抽象化概念

Template Method Pattern

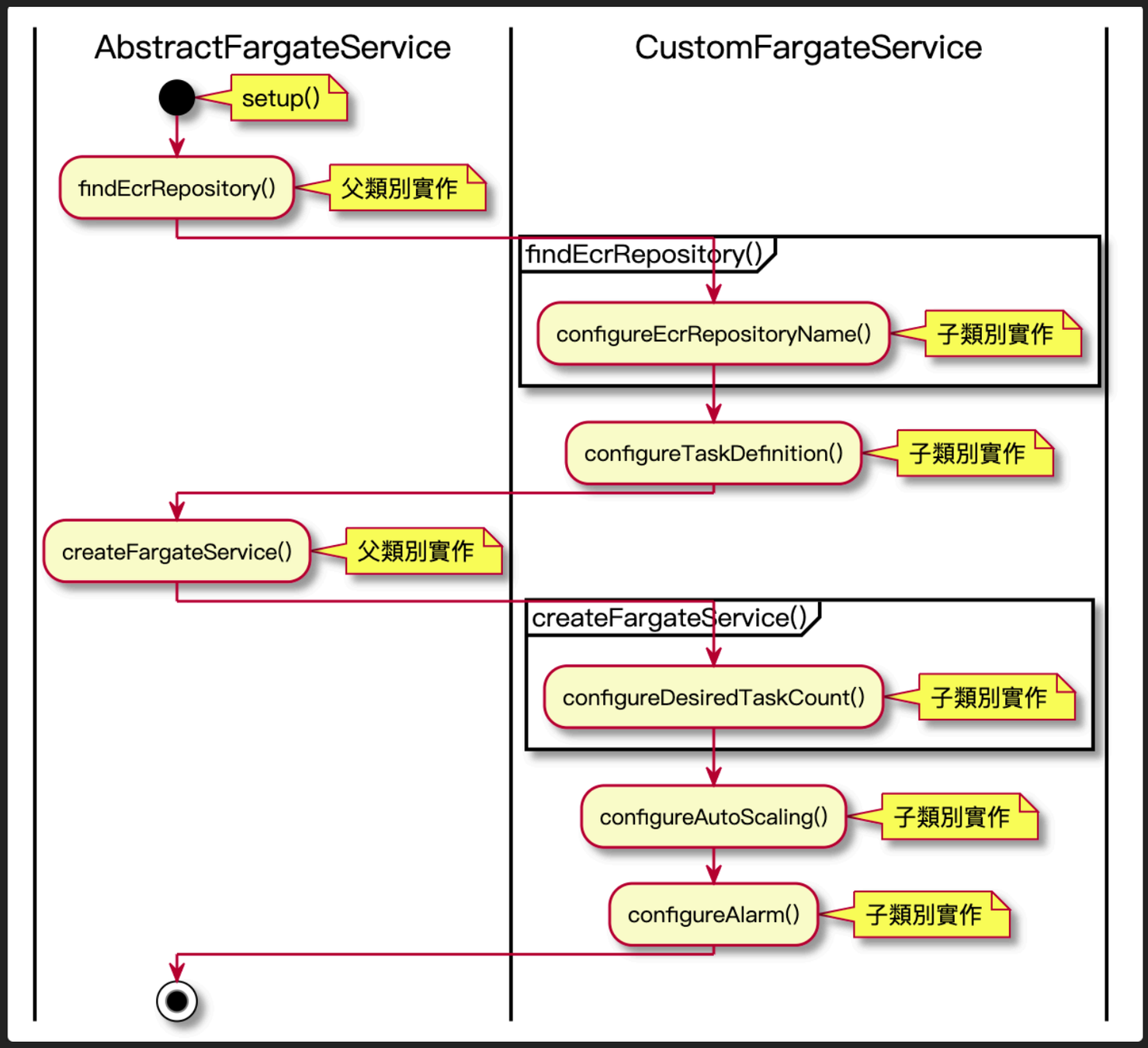
Template Method Pattern

- 原則是父類別把演算法的步驟定義好, 但是每一個步驟的細節由子類別實作跟執行
- 舉例: 『父親』 規定好 『孩子』 一天的行程
    - 吃早餐();
    - 去學校();
    - 上課();
    - 下課();
    - 回家();
    - 吃晚餐();
    - 睡覺();
- 但從 吃早餐() 、去學校()、上課() ...到 睡覺() 的細節就是由『孩子』執行

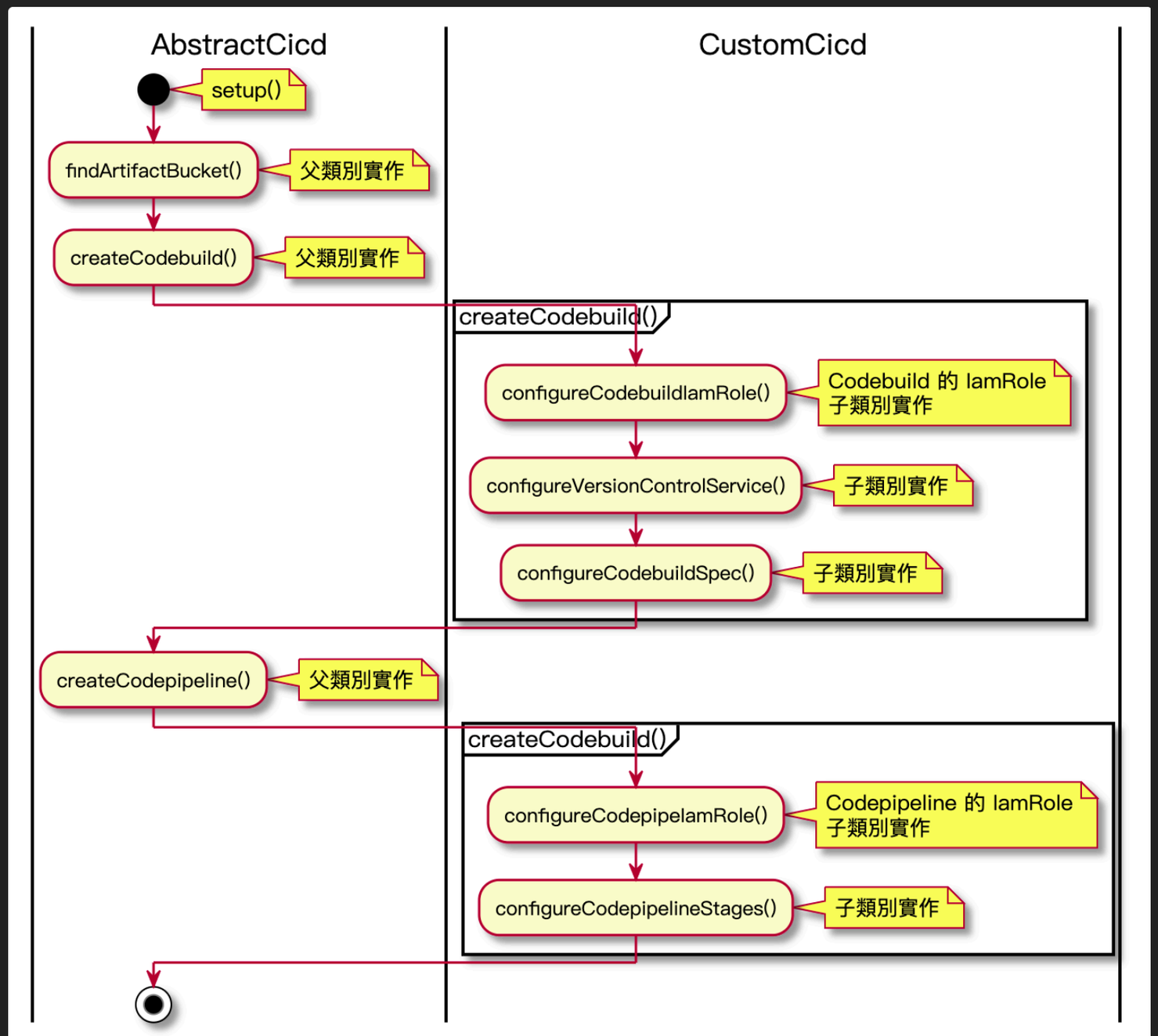
Template Method Pattern

- 找出每組 Microservice 共同的建立行為
- 可能產生不同的實作行為就委託給子類別實作
- 以 ECS Fargate 為例

找出每組 Fargate 共同的建立行為



找出每組 CICD 共同的建立行為

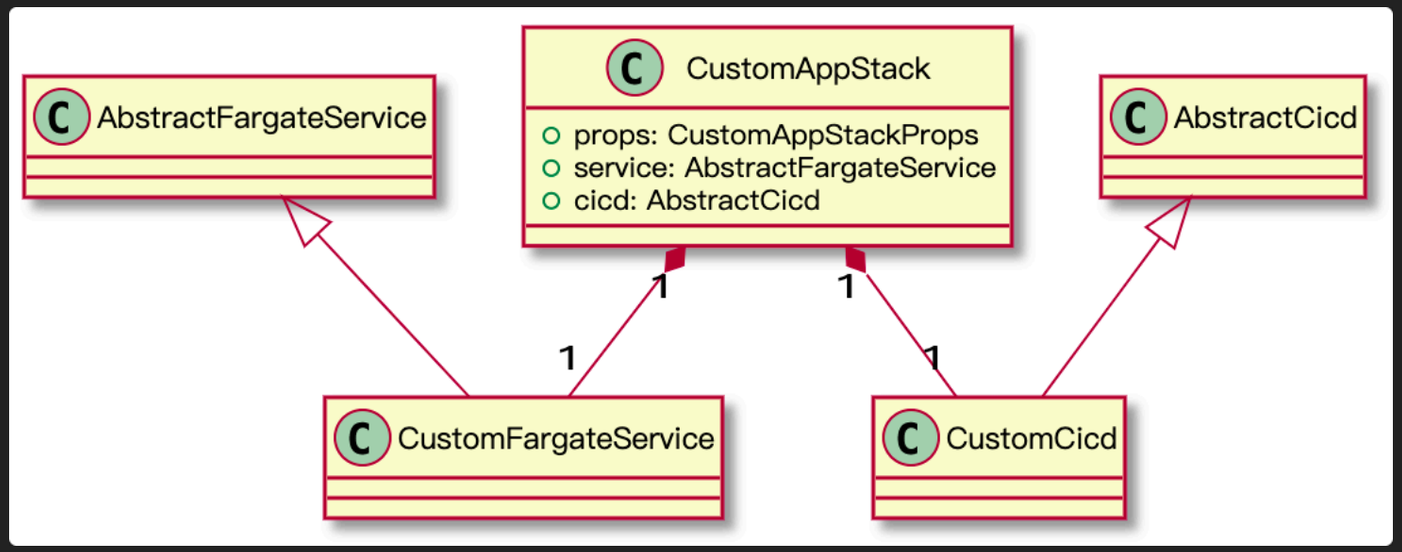


Abstract Class 示意圖

```
export abstract class AbstractFargate extends cdk.Construct {

  private setup(): Fargate {

    const repository = this.findEcrRepository();

    const definition = this.configureTaskDefinition();

    const fargate = this.createFargateService();

    this.configureAutoScaling();

    this.configureAlarm();
```

```
protected abstract configureTaskDefinition(): ecs.FargateTaskDefinition;
```

Class Diagram



好處

- 明確清晰的行為定義跟規範, 每組 Microservice 的行為一致
- 每一組 Microservice 都擁有自己獨立的 Stack
- 定義出的框架也允許讓各自 Project 的開發者動手寫自己的 CDK Code

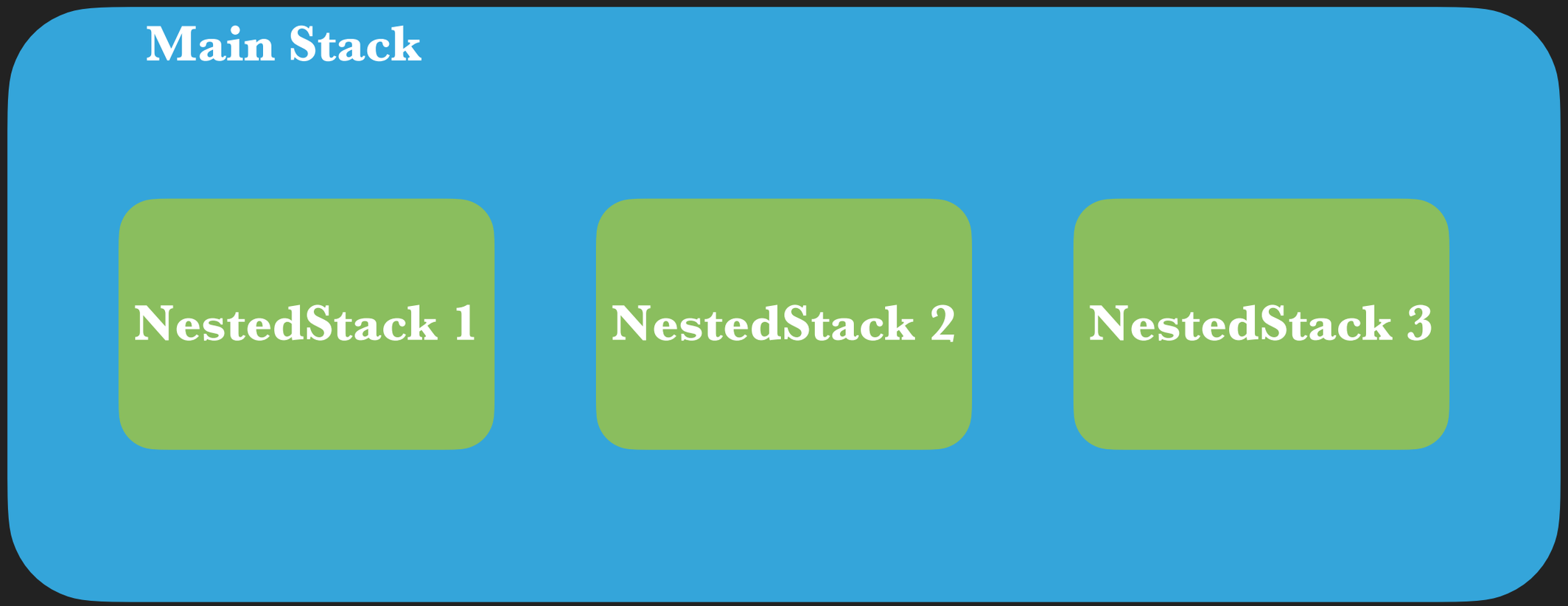
Microservice 在 CDK 上的問題

- Microservices 有 N 組以上, 可能 50 組、100 組 .... 以上
- 每組 Microservices CDK Code 長得很像、卻又不一樣
- 不希望有大量重複的 Code, 但又希望每組 Service 保有客製彈性
- Cloudformation 單一 Stack 有 200 個 Resource 的限制

Nested Stack vs Multiple Stack

Nested Stack 優點

- Stack 之間 Codebase 相同, CDK Code 裡面方法、變數都可以直接共用
- 只要操作 Main Stack 即可



Nested Stack 缺點

- cdk diff 不會顯示 Nested Stack 的任何變化
- cdk deploy 在部署的過程中不會顯示 Resource 建立狀況
- cdk deploy 在部署的過程中發生錯誤, 不會提示錯誤細節

Nested Stack 缺點

Nested Stack 缺點

- cdk diff 不會顯示 Nested Stack 的任何變化
- cdk deploy 在部署的過程中不會顯示 Resource 建立狀況
- cdk deploy 在部署的過程中發生錯誤, 不會提示錯誤細節

Show nested stack events when deploying #4489

Open **nakedible** opened this issue on 13 Oct 2019 · 1 comment

cdk diff does not work with nested stacks #5722

Open **cmckni3** opened this issue on 9 Jan · 6 comments · May be fixed by #7805

Multiple Stack 優點

- 相較 Nested Stack 無明顯優點

如何對 CDK 做模組化

但 Multiple Stack 解決我的問題

如何解決 STACK 跟 STACK 間的協調問題？

如何解決 Stack 跟 Stack 間的協調問題？

My Approach

Stack 定義好外部可存取但不能修改的 Properties

lib/cluster-stack.ts

```
export class ClusterStack extends cdk.Stack {
  readonly vpc: ec2.IVpc;
  readonly cluster: ecs.Cluster;

  constructor(scope: cdk.Construct, id: string, props: MyClusterStackProps) {
    super(scope, id, props);

    this.vpc = ec2.Vpc.fromLookup(this, `Vpc`, {
      isDefault: true,
    });

    this.cluster = new ecs.Cluster(this, `MyCluster`, {
      vpc: this.vpc,
      clusterName: `MyCluster`,
    });
  }
}
```

定義好 CDK Stack 名稱的 Pattern

bin/fargates.ts

```
const app = new cdk.App();

const cluster = new ClusterStack(app, `MyStagingClusterStack`, props);

const fargateA = new FargateAStack(app, `MyStagingFargateAStack`, {
  ...props,
  vpc: cluster.vpc,
  ecsCluster: cluster.cluster,
});
fargateA.addDependency(cluster);

const fargateB = new FargateBStack(app, `MyStagingFargateBStack`, {
  ...props,
  vpc: cluster.vpc,
  ecsCluster: cluster.cluster,
});
fargateB.addDependency(cluster);
```

定義好 CDK Stack 名稱的 Pattern

bin/fargates.ts

```ts
const app = new cdk.App();

const cluster = new ClusterStack(app, `MyStagingClusterStack`, props);

const fargateA = new FargateAStack(app, `MyStagingFargateAStack`, {
  ...props,
  vpc: cluster.vpc,
  ecsCluster: cluster.cluster,
});
fargateA.addDependency(cluster);

const fargateB = new FargateBStack(app, `MyStagingFargateBStack`, {
  ...props,
  vpc: cluster.vpc,
  ecsCluster: cluster.cluster,
});
fargateB.addDependency(cluster);
```

MyStaging*

執行 CDK Command 可以指定 名稱 Pattern*

```
#!/bin/bash

cdk diff MyStaging* \
--app="npx ts-node ./bin/my-staging-fargates.ts" \
--toolkit-stack-name=CDKToolkit
```

如何解決 STACK 跟 STACK 間的協調問題？

沒感覺嗎？來個小 Demo！

堆疊架構該思考的事

- 如何把 CDK 導入至現有的環境？
- 如何對 CDK 做模組化？
- 如何解決 Stack 跟 Stack 間的協調問題？

可能的方向

- 設計 Config, 並且針對每一個 AWS Resource 作開關
  - 開關範圍可以依照情境或狀況去設計
- 設計你的 CDK 模組之前
  - 盤點資源、歸納用途
    - 依照目前的情境或未來的展望做模組化
      - 依照團隊去切分模組
      - 依照功能去切分模組
    - 依需要引入軟體工法(Design Pattern)
- Stack 跟 Stack 之間的協調
  - 如果大家有更好的方法, 或是~~奇技淫巧~~, 歡迎一起交流

Thank you