AWS CDK Meetup Taipei

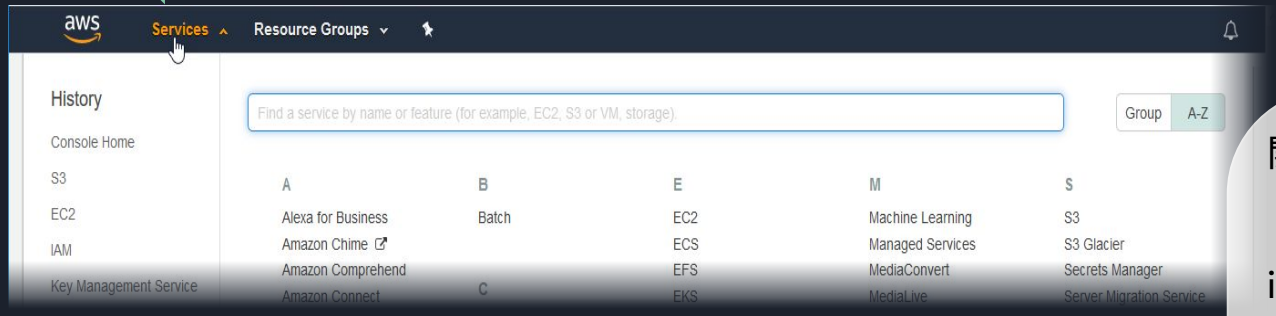
CDK build runner 翻玩 pipeline

CathayHoldings : Neil Kuan

About Me

```
(.env) [11:58]neilguan:~/aws-cdk-meetup[master !?] >>> cdk synth
Resources:
  AboutMEBD2499B9:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: cathayholdings.com.tw
        Version: "2012-10-17"
      Description: CathayHoldings DDT  Enginner
      RoleName: Neil-Guan
      Tags:
        - Key: a.NAME
          Value: Neil Guan
        - Key: b.EVENT
          Value: CDK Meetup Taipei
        - Key: c.SKILLS
          Value: AWS K8S OPENSHIFT
        - Key: d.EMAIL
          Value: neilguan@cathayholdings.com.tw
    Metadata:
      aws:cdk:path: Neil-Guan-Profile/About_ME/Resource
```

About why not use AWS CDK story ?!
(本故事純屬虛構，如有雷同實屬巧合)



開發 : 最近 AWS 有新出服務誒，
想要測試看看

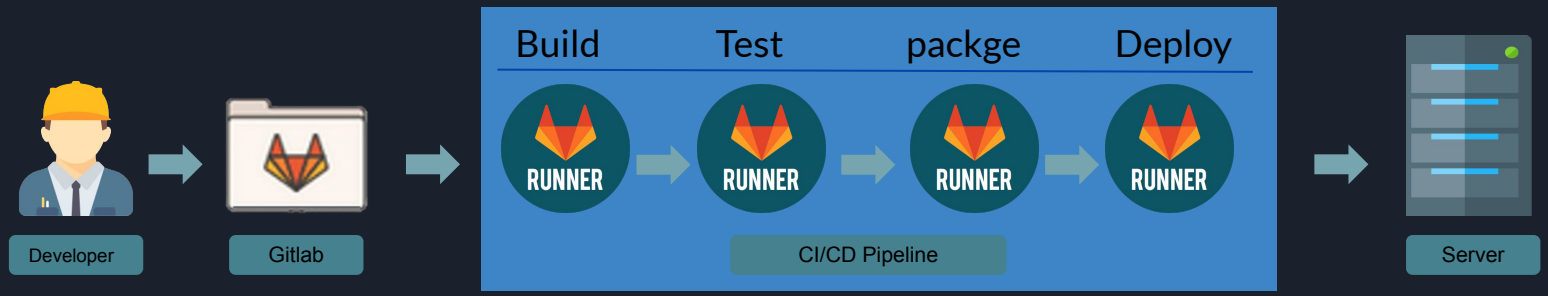
infra : 好 開給你試試

(當帳單 開始飆漲 會計/主管眼神凝視著
表示要開始小心，使用情況)

從 aws console -> aws cli -> cloudformation(手刻 真的很吃經驗)
AWS CDK 的降臨！！！

About why not use AWS CDK story ?!

CI/CD Pipeline On Gitlab



About Gitlab Runner

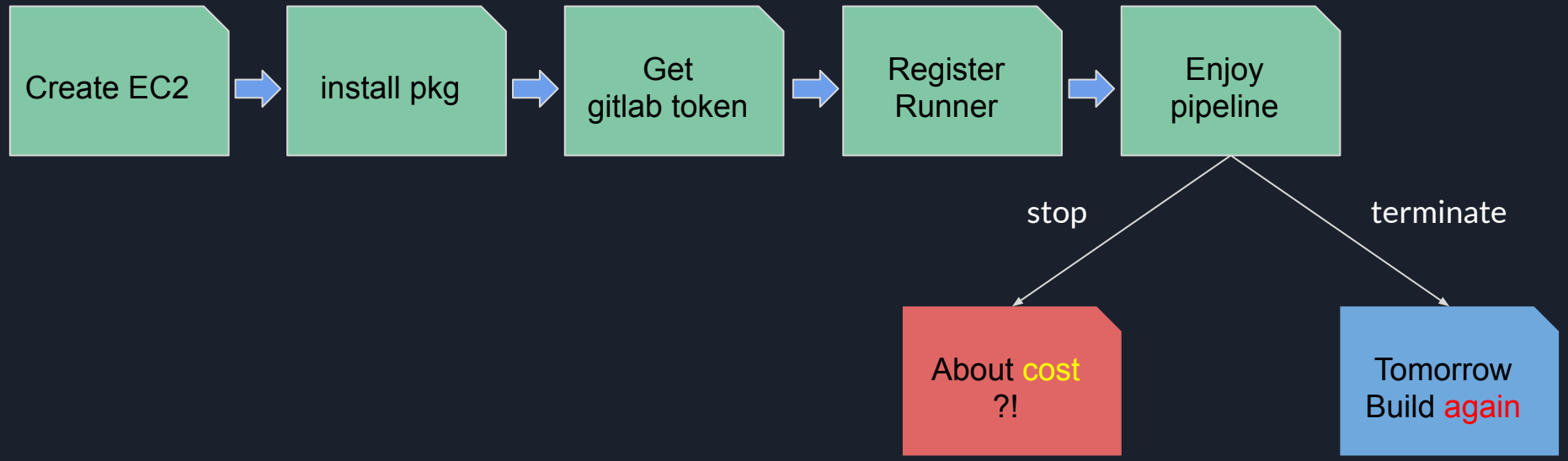
GitLab Runner是一個開源項目，用於運行您的作業並將結果發送回 GitLab。

它與GitLab CI / CD結合使用，GitLab CI / CD是GitLab隨附的用於協調作業的開源持續集成服務。

Share Runner (2000 mins/M)

- 幫助你執行您的 CI/CD Job
- CI 感覺可以用 Share Runner，那 CD 呢？！
- CD 畢竟會觸及到您的環境，感覺Share Runner 不是一個很好的選擇。-> build self runner.

Build your Gitlab Runner



Build your Gitlab Runner

Build your Gitlab Runner

CDK-GITLAB-RUNNER Construct lib

README.md

npm package 1.47.1 | pypi package 1.47.1 | Release passing

DOWNLOADS: npm 410 | pypi 526/month

iam role self enable | vpc self enable | 1.47.1 stable

Welcome to `cdk-gitlab-runner`

This repository template helps you create gitlab runner on your aws account via AWS CDK one line.

Note

Default will help you generate below services:

- VPC
  - Public Subnet (2)
- EC2 (1 T3.large)

Before start you need gitlab runner token in your `gitlab project` or `gitlab group` : https://github.com/guan840912/cdk-gitlab-runner

Let’s Build Runner via CDK

只需幾行代碼創建自己的gitlab runner 。

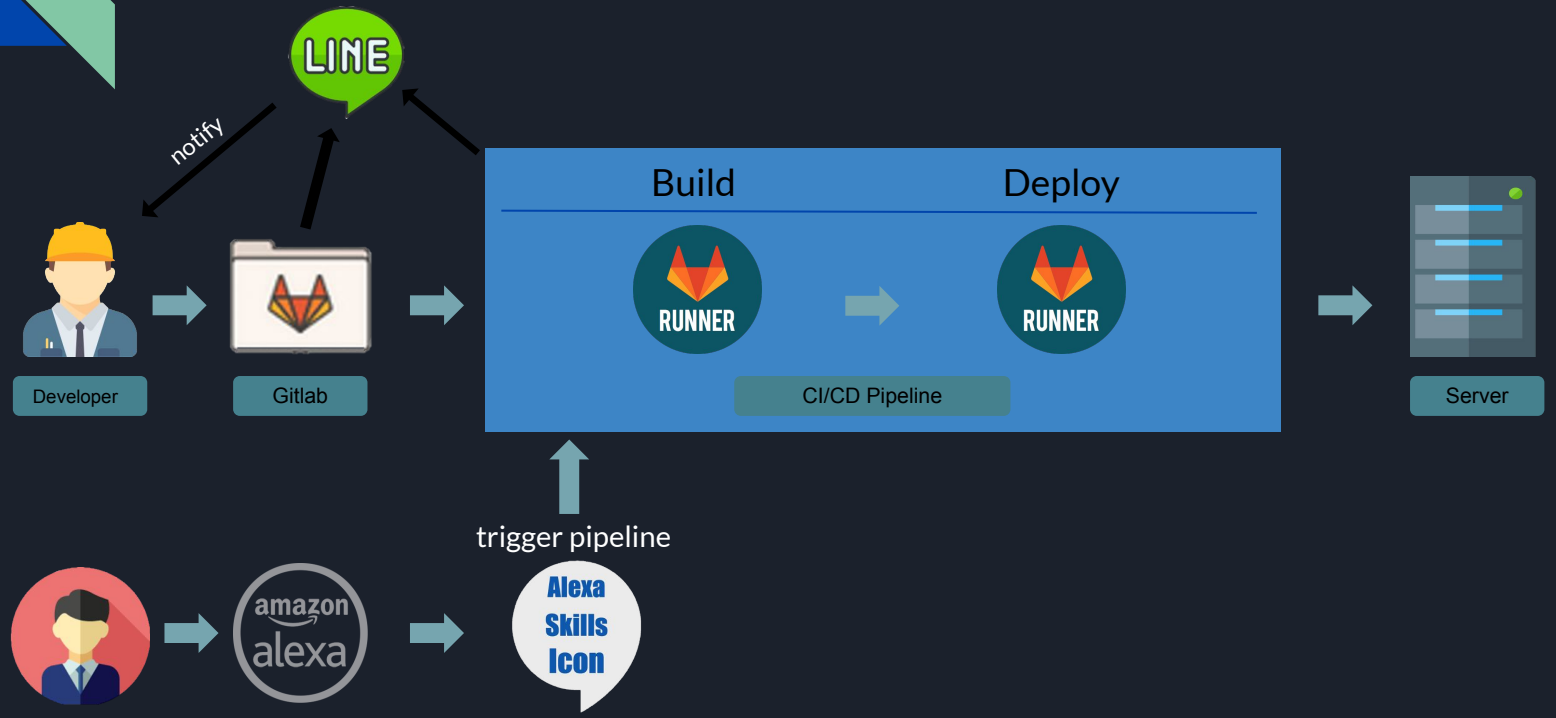
aws_cdk_meetup_stack.py × app.py

aws_cdk_meetup > aws_cdk_meetup_stack.py > ...

```python
class AwsCdkMeetupStack(core.Stack):

    def __init__(self, scope: core.Construct, id: str, **kwargs) -> None:
        super().__init__(scope, id, **kwargs)
        # find self account default VPC
        selfvpc = ec2.Vpc.from_lookup(self, 'MyVPC', is_default=True)
        # Create a runner via https://pypi.org/project/cdk-gitlab-runner/
        runner = GitlabContainerRunner(self, 'gitlab-runner', gitlabtoken=os.environ['GITLABTOKEN'],
                                       ec2type="t3.small", tag1='cdk', tag2='meetup', tag3='aws', selfvpc=selfvpc)
        # add another policy to runner runner.runner_role.add_managed_policy(iam.ManagedPolicy.from_aws_managed_po
```

CDK demo ...


LINE
notify
Developer
Gitlab
Build
Deploy
RUNNER
RUNNER
CI/CD Pipeline
Server
trigger pipeline
amazon
alexa
Alexa
Skills
Icon

demo source code

```python
from cdk_gitlab_runner import GitlabContainerRunner
from aws_cdk import (
    core, aws_ec2 as ec2,
    aws_iam as iam,
    aws_route53 as r53
)
import os
import requests
# my ip
myip = requests.get('https://checkip.amazonaws.com').text.rstrip()
# need to change your host_zone .
my_hosted_zone = os.environ['NEIL_HOST_ZONE']
# need to change your zone_name .
my_zone_name = os.environ['NEIL_HOST_ZONE_NAME']


class AwsCdkMeetupStack(core.Stack):

    def __init__(self, scope: core.Construct, id: str, **kwargs) -> None:
        super().__init__(scope, id, **kwargs)
        # find self account default VPC
        selfvpc = ec2.Vpc.from_lookup(self, 'MyVPC', is_default=True)
        # Create a runner via https://pypi.org/project/cdk-gitlab-runner/
        runner = GitlabContainerRunner(self, 'gitlab-runner', gitlabtoken=os.environ['GITLABTOKEN'],
                        ec2type="t3.small", tag1='cdk', tag2='meetup', tag3='aws',selfvpc=selfvpc)
        # add another policy to runner
runner.runner_role.add_managed_policy(iam.ManagedPolicy.from_aws_managed_policy_name("AmazonS3ReadOnlyA
ccess"))
        # EKS mapping aws-auth config role , ecs deploy policy
        # Let's access lab web .
        runner.runner_ec2.connections.allow_from(
            ec2.Peer.ipv4(myip+'/32'), ec2.Port.tcp(80))
        runner.runner_ec2.connections.allow_from(
            ec2.Peer.ipv4(myip+'/32'), ec2.Port.tcp(443))
        # find my route53 hostzone .
        zone = r53.HostedZone.from_hosted_zone_attributes(
            self, 'MYHOSTED_ZONE', hosted_zone_id=my_hosted_zone, zone_name=my_zone_name)

        # target runner instance public ip .
        runnerip = runner.runner_ec2.instance_public_ip
        target_ins = r53.RecordTarget.from_ip_addresses(runnerip)
        newdomain = r53.ARecord(self, "A", zone=zone,target=target_ins,record_name="cdkdemo",
                                ttl=core.Duration.minutes(1))
        # http://cdk-demo.jsregistry.ga
        core.CfnOutput(self, 'Runner-Public-DNS-NAME',
                       value=newdomain.domain_name)
```

iam source code

```python
from aws_cdk import (
    aws_iam as iam,
    core
)

class AboutMe(core.Stack):
    def __init__(self, scope: core.Construct, id: str, **kwargs) -> None:
        super().__init__(scope, id, **kwargs)

        iam.Role(
            self, 'About_ME',
            role_name='Neil-Guan' ,
            assumed_by=iam.ServicePrincipal('cathayholdings.com.tw'),
            description="CathayHoldings DDT  Enginner")

app = core.App()
neil_guan= AboutMe(app, "Neil-Guan-Profile")
core.Tag.add(neil_guan , "a.NAME","Neil Guan", priority=3)
core.Tag.add(neil_guan , "b.EVENT","CDK Meetup Taipei", priority=1)
core.Tag.add(neil_guan , "c.SKILLS","AWS K8S OPENSHIFT", priority=2)
core.Tag.add(neil_guan , "d.EMAIL","neilguan@cathayholdings.com.tw", priority=4)
app.synth()
```