AWS Thai Blog

กำหนดหมายเลขไอพีสาธารณะแบบคงที่ให้กับเครื่อง worker ของ Amazon EKS ใน Local Zones ด้วย KubeIP v2

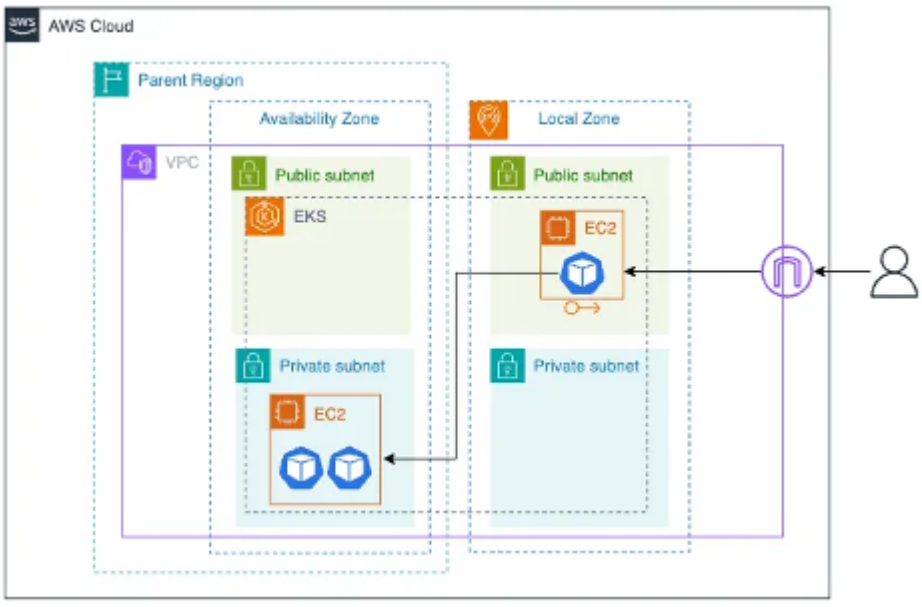
by Wiriyang Pipatsakulroj | on 03 OCT 2024 | in Amazon Elastic Kubernetes Service, Compute | Permalink | Share

ลูกค้า AWS ใช้งาน Amazon EKS ใน AWS Local Zones เพื่อให้ลูกค้าของตนเข้าถึงบริการด้วยความหน่วงที่ต่ำ และปฏิบัติตามนโยบายด้านการประมวลผลข้อมูลในประเทศ รวมถึงในบางกรณีมีความจำเป็นต้องใช้งานหมายเลขไอพีสาธารณะแบบคงที่เพื่อสื่อสารกับระบบของคู่ค้าที่มีข้อกำหนดดังกล่าว อย่างไรก็ตาม ทรัพยากร Kubernetes (k8s) เช่น เครื่อง worker อาจมีการปิดเปิดเครื่องตามสถานการณ์ต่างๆ ทำให้เกิดการเปลี่ยนแปลงที่อยู่ไอพีของเครื่องได้ ยกตัวอย่างเช่น การอัปเกรดเวอร์ชันของ EKS คลัสเตอร์ โดยเราสามารถนำ KubeIP v2 มาช่วยในการกำหนดหมายเลขไอพีสาธารณะแบบคงที่ได้ โดยใช้ประโยชน์จากความสามารถของคลาวด์ เพื่อให้แน่ใจว่าการกำหนดที่อยู่ไอพีจะเป็นหมายเลขเดิม แม้ว่าเครื่อง worker จะถูกปิดและสร้างขึ้นใหม่ก็ตาม

โซลูชันโดยรวม

บทความนี้แสดงวิธีการติดตั้ง Amazon EKS cluster ที่ประกอบด้วย managed node group ที่อยู่ใน AWS region และ self-managed node group ที่อยู่ใน Local Zone รวมทั้งสาธิตการกำหนด Elastic IP Address ของ AWS ให้กับ EKS worker ใน Local Zone โดยใช้ KubeIP v2

แผนภาพสถาปัตยกรรมด้านล่างแสดงถึงแอปพลิเคชัน edge ทำงานอยู่บนเครื่อง EC2 ที่มีหมายเลขไอพีสาธารณะแบบคงที่กำหนดอยู่ใน Local Zone และรับทราฟฟิคจากผู้ใช้งาน จากนั้นส่งต่อทราฟฟิคไปยังแอปพลิเคชัน backend ที่อยู่ใน AWS region



ข้อกำหนดเบื้องต้น

- เปิดใช้งาน Local Zones บน AWS region ที่รันแอปพลิเคชัน
- กำหนดสิทธิ์ในบัญชี AWS ที่จำเป็นสำหรับใช้งาน AWS CLI
- ติดตั้งเครื่องมือ CLI ที่ใช้ในการสร้างทรัพยากรตามตัวอย่างดังนี้ AWS CLI, Terraform และ kubectl

มาลงมือสร้างกัน

1) ในบทความนี้ เราจะทำการดีพอยตามตัวอย่างซอร์สโค๊ด ซึ่งประกอบไปด้วย 4 ไดเรกทอรี ได้แก่ 01-vpc 02-eks 03-kubeip และ 04-app โดยจะเริ่มดีพอยจาก 01-vpc ไป 04-app

เริ่มแรก ให้ทำการคัดลอกตัวอย่างจากบน github มายังไดเรกทอรีปัจจุบัน

```Bash
git clone https://github.com/duoh/kubeip-eks-localzone
cd kubeip-eks-localzone
```

2) เราจะทำการสร้าง VPC และ subnet ใน AWS region และ Local Zone ตามในไฟล์ main.tf โดยใช้โมดูล vpc เพื่อสร้าง subnet ใน AWS region และใช้ aws_subnet resources ในการสร้าง subnet ใน Local Zone

```Bash
  private_subnet_tags = {
    "kubernetes.io/cluster/${var.cluster_name}" = "shared"
    "kubernetes.io/role/internal-elb"           = "1"
  }
}

resource "aws_subnet" "public-subnet-lz" {
  vpc_id                  = module.vpc.vpc_id
  cidr_block              = cidrsubnet(var.vpc_cidr, 8, 5)
  availability_zone       = local.lzs[0]
  map_public_ip_on_launch = true
}

resource "aws_subnet" "private-subnet-lz" {
  ...
}

...
```

กำหนดค่าตัวแปรอินพุต จากตัวอย่าง name (VPC) เป็นค่า `kubeip-lz-eks-vpc` ส่วน region (AWS region) เป็น 'ap-southeast-1' และ lzs (Local Zone) คือ 'ap-southeast-1-bkk-1a'

```Bash
cd 01-vpc
vi example.auto.tfvars
```

```Bash
region          = "ap-southeast-1"
lzs             = ["ap-southeast-1-bkk-1a"]
name            = "kubeip-eks-lz-vpc"
vpc_cidr        = "10.0.0.0/16"
cluster_name    = "kubeip-eks-lz-cluster"
```

ทำการดีพอย VPC subnet และ route table ด้วย terraform

```Bash
terraform init
terraform apply -auto-approve
```

โน๊ตผลลัพธ์ไว้ใช้งานในขั้นตอนถัดไป

```Bash
Outputs:
private_subnets = [
  "subnet-09139a5ea9dcd335d",
  "subnet-0d7cc25e5066687be",
  "subnet-04bb1e787c9b6e28f",
]
public_subnets_local_zone = "subnet-0edd1b731c48fb7d7"
vpc_id = "vpc-061f434818a666d8b"
```

3) ถัดมา ให้เราทำการสร้าง EKS คลัสเตอร์ที่มี managed node group ใน AWS region และสร้าง self-managed node group ใน Local Zone สำหรับ self-managed node group นั้น จะมีแท็ก 'eks.amazonaws.com/nodegroup=public-lz-ng' และ 'kubeip=use' กำหนดอยู่ด้วย นอกจากนี้เรายังใช้โมดูล kubeip_role เพื่อสร้าง IRSA (IAM Role for Service Accounts) สำหรับ KubeIP daemonSet

```Bash
module "eks" {
  source  = "terraform-aws-modules/eks/aws"
  cluster_name    = var.cluster_name
  cluster_version = "1.30"

  vpc_id                         = var.vpc_id
  subnet_ids                     = var.private_subnets
  cluster_endpoint_public_access = true
  enable_irsa = true
  ...
```

```
  eks_managed_node_groups = {
    region-ng = {
      ...
      subnet_ids    = var.private_subnets
      labels = {
        region  = "true"
      }
```

กำหนดตัวแปรอินพุต สำหรับ vpc_id private_subnets และ public_subnets_local_zone โดยให้ใช้ค่าที่คัดลอกจากผลลัพธ์ก่อนหน้า

```
Bash
  cd ../02-eks
  vi example.auto.tfvars
```

```
Bash
  region                 = "ap-southeast-1"
  vpc_id                 = "vpc-061f434818a666d8b"
  private_subnets          = [
                          "subnet-09139a5ea9dcd335d",
                          "subnet-0d7cc25e5066687be",
                          "subnet-04bb1e787c9b6e28f",
                         ]
  public_subnets_local_zone  = "subnet-0edd1b731c48fb7d7"
  cluster_name             = "kubeip-eks-lz-cluster"
  kubeip_role_name           = "kubeip-agent-role"
  kubeip_sa_name             = "kubeip-agent-sa"
```

ติดตั้งทรัพยากรต่างๆ ด้วย terraform

```
Bash
  terraform init
  terraform apply -auto-approve
```

บันทึกผลลัพธ์ ไว้สำหรับขั้นตอนถัดไป

```
Bash
  Outputs:
  eks_cluster_name = "kubeip-eks-lz-cluster"
  kubeip_role_arn = "arn:aws:iam::xxxxxxxxxxxx:role/kubeip-agent-role"
```

4) จากนั้น สร้าง Elastic IP Address และทรัพยากร k8s เช่น DaemonSet และ service account โดย Elastic IP Address ถูกสร้างตามที่กำหนดไว้ใน aws_eip resource ซึ่งแยกตามโซนผ่าน network_border_group argument สำหรับ KubeIP DaemonSet นั้น จะรันบนเครื่อง worker ใน Local Zone ซึ่งถูกกำหนดโดย node_selector field ของ

k8s ส่วน KubeIP เองก็สามารถเลือก EIP โดยใช้แท็กของ AWS ได้ ที่สำคัญคือเราต้องทำการสร้าง IRSA และกำหนด RBAC ให้กับ service account สำหรับ KubeIP DaemonSet

Bash

```
resource "kubernetes_service_account" "kubeip_service_account" {
  metadata {
    name        = var.kubeip_sa_name
    namespace   = "kube-system"
    annotations = {
      "eks.amazonaws.com/role-arn" = var.kubeip_role_arn
    }
  }
}

resource "kubernetes_cluster_role" "kubeip_cluster_role" {
  ...
}

resource "kubernetes_cluster_role_binding" "kubeip_cluster_role_binding" {
  ...
}
```

กำหนดตัวแปรอินพุต สำหรับ kubeip_role_arn ให้ใช้ค่าจากผลลัพธ์ก่อนหน้า

Bash

```
cd ../03-kubeip
vi example.auto.tfvars
```

Bash

```
region               = "ap-southeast-1"
network_border_group    = "ap-southeast-1-bkk-1"
cluster_name         = "kubeip-eks-lz-cluster"
kubeip_role_arn       = "arn:aws:iam::xxxxxxxxxxxx:role/kubeip-agent-role"
kubeip_sa_name        = "kubeip-agent-sa"
```

ติดตั้ง Elastic IP Address (EIP) และทรัพยากร Kubernetes (k8s) โดยใช้ Terraform

Bash

```
terraform init
terraform apply -auto-approve
```

บันทึกค่า IP ไว้เพื่อใช้ในการทดสอบ หลังจากดีพอยแอปพลิเคชันเสร็จ

```
Bash
  elastic_ips = [
    "15.220.243.225",
  ]
```

5) ท้ายสุด เราจะทำการทดสอบระบบโดยดีพอยแอปพลิเคชันบน EKS คลัสเตอร์

ไฟล์ที่มีชื่อว่า app_a.yaml และ app_b.yaml นั้นไว้สำหรับสร้างแอปพลิเคชัน backend ที่ทำงานใน AWS region

```
Bash
  apiVersion: apps/v1
  kind: Deployment
  metadata:
    name: app-a-deployment
    ...
  spec:
    ...
      spec:
        containers:
        - name: app-a
          image: hashicorp/http-echo
          ports:
          - containerPort: 5678
          args: ["-text=<h1>I'm APP <em>A</em></h1>"]
        nodeSelector:
          region: "true"
  ---
  apiVersion: v1
  kind: Service
```

ไฟล์ที่มีชื่อว่า edge_svc.yaml นั้น ไว้สร้างแอปพลิเคชัน edge ที่รันใน Local Zone

```
Bash
```

```
  spec:
    type: NodePort
    ...
```

ย้ายไดเรกทอรีปัจจุบันไปที่ 04-app และรันคำสั่ง aws cli ตามด้วย AWS region ที่ใช้งาน เพื่ออัปเดตไฟล์ kubeconfig สำหรับใช้ยืนยันตัวตนกับ EKS คลัสเตอร์

```Bash
cd ../04-app
aws eks update-kubeconfig --name kubeip-lz-cluster --region ap-southeast-1
```

รันคำสั่ง kubectl เพื่อทดสอบว่าติดต่อกลับคลัสเตอร์ได้

```Bash
kubectl get no
```

```Bash
NAME                                              STATUS   ROLES    AGE     VERSION
ip-10-0-10-213.ap-southeast-1.compute.internal   Ready    <none>   4m      v1.30.0-eks-036c24b
ip-10-0-5-76.ap-southeast-1.compute.internal     Ready    <none>   3m11s   v1.30.0-eks-036c24b
```

ติดตั้งแอปพลิเคชันและตรวจสอบว่าทำงานได้หรือไม่

```Bash
kubectl apply -f edge_svc.yaml
kubectl apply -f app_a.yaml
kubectl apply -f app_b.yaml

kubectl get po
```

```Bash
NAME                              READY   STATUS    RESTARTS   AGE
app-a-deployment-587b484997-k6f8q   1/1     Running   0          12s
app-b-deployment-78bc6675db-2mk2s   1/1     Running   0          12s
edge-deployment-f6b9f4d5f-5j6f7     1/1     Running   0          13s
```

ตรวจสอบผลลัพธ์ว่าแอปพลิเคชันทำงานได้ โดยใช้ที่อยู่ไอพีสาธารณะจากผลลัพธ์ก่อนหน้า

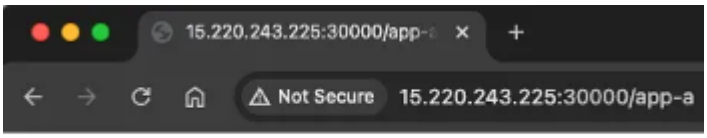
ทำการทดสอบผ่าน curl

```Bash
% curl 15.220.243.225:30000
<h1>I am EDGE SERVICE</h1>
```

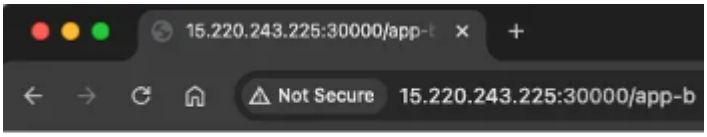
```
% curl 15.220.243.225:30000/app-a
<h1>I'm APP <em>A</em></h1>

% curl 15.220.243.225:30000/app-b
<h1>I'm APP <em>B</em></h1>
```

I am EDGE SERVICE



I'm APP *A*



I'm APP *B*

บทสรุป

AWS Local Zones มีข้อจำกัดจากการใช้งานได้แค่ Application Load Balancer (ALB) จากบริการของ Elastic Load Balancing (ELB) เท่านั้น นอกจากนี้หมายเลขไอพีสาธารณะที่กำหนดให้กับ ALB นั้น สามารถเปลี่ยนแปลงได้ตลอดเวลา สำหรับผู้ดูแล EKS คลัสเตอร์ที่ใช้งานใน Local Zone และจำเป็นต้องมั่นใจว่าค่าไอพีเดิมจะถูกกำหนดให้แก่ระบบอยู่ตลอดเวลา ไม่ว่าเครื่อง worker จะถูกปิดและเปิดขึ้นมาใหม่ก็ตาม ทำให้ต้องสร้างและดูแลโซลูชันเอง ซึ่งในกรณีนี้ KubeIP v2 เป็นตัวเลือกที่เหมาะสม ที่สามารถนำมาใช้ตอบโจทย์ดังกล่าวได้

บทความนี้เขียนโดยคุณ Wiriyang Pipatsakulroj, Senior Cloud Architect DoiT

Review โดยคุณ Chatcharoen Chivakanit, Senior Solutions Architect AWS

เกี่ยวกับ DoiT

DoiT จัดหาเทคโนโลยีอัจฉริยะและความเชี่ยวชาญด้านคลาวด์ เพื่อช่วยให้องค์กรเข้าใจและใช้ประโยชน์จาก Public Cloud อย่าง Amazon Web Services (AWS) ในการขับเคลื่อนการเติบโตทางธุรกิจ คุณสามารถดูข้อเสนอและสอบถามได้ที่ doit.com