AWS Thai Blog

Karpenter เปิดตัว version 1.0

by Patis Piriyahaphan | on 31 DEC 2024 | in Amazon Elastic Kubernetes Service, Containers | Permalink | Share

บทนำ

ในเดือนพฤศจิกายน 2564, AWS ได้ประกาศเปิดตัว Karpenter v0.5 ซึ่งเป็นโครงการโอเพนซอร์สใหม่เกี่ยวกับ Kubernetes cluster auto scaling แต่เดิมถูกคิดค้นขึ้นมาเพื่อเป็นทางเลือกที่ยืดหยุ่น และมีประสิทธิภาพสูงแทน Kubernetes Cluster Autoscaler ในช่วงเกือบสามปีที่ผ่านมา Karpenter ได้พัฒนาอย่างมากจนมี feature ที่ครบถ้วนสำหรับ Kubernetes ในการจัดการ lifecycle ของ Kubernetes node

บทความนี้แปลมาจาก Announcing Karpenter 1.0 โดย Alex Kestner, Robert Northard, และ Rajdeep Saha

โครงการนี้ได้ถูกนำไปใช้ใน use case ที่สำคัญในบริษัทที่เป็นผู้นำหลาย ๆ ในอุตสาหกรรม มีการเพิ่ม feature สำคัญ เช่น workload consolidation ซึ่งออกแบบมาเพื่อเพิ่ม utilization โดยอัตโนมัติ และ disruption controls เพื่อให้ผู้ใช้สามารถระบุวิธีการและเวลาที่ Karpenter จะดำเนินการจัดการ lifecycle ของ node ใน cluster ในเดือนตุลาคม 2566 โครงการนี้ได้เลื่อนขั้นเป็น beta และ AWS เป็น contributed หลักของโครงการให้กับ Cloud Native Computing Foundation (CNCF) ผ่าน Kubernetes SIG auto scaling (Autoscaling Special Interest Group) การมีส่วนร่วมใน Karpenter community ทำให้กลายเป็นหนึ่งในสิบโครงการโอเพนซอร์สของ AWS ที่ได้รับความนิยมสูงสุดโดยดูจากจำนวนดาวใน GitHub และการมีส่วนร่วมจากสมาชิกที่ไม่ใช่ AWS ได้เพิ่มขึ้นทั้งในด้านจำนวนสมาชิกและ scope เพื่อเป็นการสนับสนุนโครงการ เพื่อให้รองรับการพัฒนาและการมีส่วนร่วมของสมาชิก ทีม Karpenter ที่ AWS ยังคงทำงานอย่างต่อเนื่องเพื่อยกระดับความสมบูรณ์และความเสถียรในการดำเนินงานของโครงการ

วันนี้ ด้วยการเปิดตัว Karpenter v1.0.0 เรารู้สึกภูมิใจที่จะประกาศว่า Karpenter ได้เลื่อนขั้นออกจาก beta แล้ว ด้วยการเปิดตัวนี้ API ของ Karpenter ทั้ง `NodePool` และ `EC2NodeClass` จะยังคงมีให้ใช้งานสำหรับการเปิดตัว minor version ของ 1.0 ในอนาคต และจะไม่มีการแก้ไขในลักษณะที่ส่งผลให้เกิดการเปลี่ยนแปลงที่จะทำให้เกิด breaking change จากการเปิดตัว minor version ไปยังอีก minor version อื่น ในโพสต์นี้เราจะอธิบายถึงการเปลี่ยนแปลงระหว่าง Karpenter v0.37 ปัจจุบันและ v1.0.0

มีอะไรเปลี่ยนแปลงบ้าง ?

ในส่วนหนึ่งของการเปิดตัว v1 นั้น ส่วนของ custom resource definition (CRD) application programming interface (API) ยังคงไม่มีการเปลี่ยนแปลง เราได้สร้าง conversion webhook เพื่อทำให้การ migrate จาก beta เป็นไปอย่างราบรื่นมากขึ้น ใน minor เวอร์ชันถัดไปของ Karpenter หลังจาก v1 (`v1.1.0`) เรามีแผนที่จะยกเลิกการสนับสนุน API v1beta1 และต่อไปนี้คือสรุปของ feature ใหม่ ๆ และการเปลี่ยนแปลงต่าง ๆ

ปรับปรุง disruption controls by reason

ในการเปิดตัว Karpenter v0.34.0 Karpenter ได้มี disruption controls (การจัดการการถูกรบกวน) เพื่อให้ผู้ใช้มีการควบคุมได้มากขึ้นว่าในการกำหนดให้ Karpenter จะ terminates nodes อย่างไรและเมื่อไร เพื่อปรับปรุงความสมดุลระหว่างประสิทธิภาพด้านต้นทุน ความปลอดภัย และ availability ของแอปพลิเคชัน disruption budgets ใช้ cron syntax และสามารถกำหนดเวลาให้ใช้ในช่วงเวลาที่ต้องการ วันของสัปดาห์ ชั่วโมง นาที หรือตลอดเวลา เพื่อให้มีแอปพลิเคชันมี availability ที่สูง โดยค่าเริ่มต้น หากไม่ได้ตั้งค่าการ disruption Karpenter จะจำกัดการ disruption ไว้ที่ 10% ของ node ในทุกช่วงเวลา

Karpenter v1 เพิ่มการรองรับ disruption by reason เพื่อที่จะ support เหตุการณ์ต่าง ๆ ได้แก่ `Underutilized`, `Empty` และ `Drifted` ซึ่งช่วยให้ผู้ใช้สามารถควบคุมการหยุดชะงักได้อย่างละเอียดมากขึ้นในแต่ล่ะเหตุการณ์ ตัวอย่างเช่น disruption budget ต่อไปนี้ทำให้ผู้ใช้ควบคุมดังนี้:

- 0% ของ node สามารถถูก disrupt ได้ตั้งแต่วันจันทร์ถึงวันศุกร์ เริ่มตั้งแต่เวลา 9:00 เป็นเวลาแปดชั่วโมง หาก drifted หรือ underutilized
- 100% ของ node สามารถถูก disrupt ได้หากว่างเปล่าตลอดเวลา
- ในช่วงเวลาอื่นๆ ของวัน อนุญาตให้ 10% ของ node ถูกรบกวนได้ เมื่อ drifted หรือ underutilized และ budget ไม่ได้ถูกใช้งาน (active) อยู่

ผู้ใช้อาจใช้ budget เหล่านี้เพื่อให้แน่ใจว่า node ที่ไม่มี workload สามารถถูก terminate ได้ในช่วงเวลาที่การใช้งานแอปพลิเคชันสูงสุด เพื่อเพิ่มประสิทธิภาพนในการใช้งาน หาก reason ไม่ได้ถูกกำหนดไว้ budget ถูกใช้กับทุกเหตุการณ์

YAML

```yaml
...
disruption:
  budgets:
  - nodes: “0”
    schedule: "0 9 * * mon-fri"
    duration: 8h
    reasons:
    - Drifted
    - Underutilized
  - nodes: "100%"
    reasons:
    - Empty
  - nodes: "10%"
    reasons:
    - Drifted
    - Underutilized
...
```

เปลี่ยนชื่อ consolidation policy จาก WhenUnderutilized เป็น WhenEmptyOrUnderutilized

`WhenUnderutilized` consolidation policy ได้ถูกเปลี่ยนชื่อเป็น `WhenEmptyOrUnderutilized` ฟังก์ชันการทำงานยังคงเหมือนเดิมเช่น ใน v1beta1 ที่ Karpenter จะ consolidate node ที่ถูกใช้งานบางส่วนหรือว่างเปล่าเมื่อ `consolidationPolicy=WhenUnderutilized` ในชื่อใหม่ `WhenEmptyOrUnderutilized` จะทำให้แสดงถึงเงื่อนไขอย่างชัดเจนและถูกต้อง

YAML

```yaml
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: default
spec:
```

```
  disruption:
    consolidationPolicy: WhenEmptyOrUnderutilized
```

`consolidationPolicy=WhenEmpty` เท่านั้น ซึ่งเป็นเวลาที่ pod สุดท้ายถูกลบออก ตอนนี้ `consolidateAfter` สามารถใช้ได้เมื่อ `consolidationPolicy=WhenEmptyOrUnderutilized` ดังนั้นจึงอนุญาตให้ผู้ใช้ระบุเป็นชั่วโมง นาที หรือวินาทีว่า Karpenter จะรอนานเท่าไหร่เมื่อมีการเพิ่มหรือลบ pod ก่อนที่จะทำการ consolidate หากคุณต้องการพฤติกรรมเดียวกับ v1beta1 ให้ตั้งค่า `consolidateAfter` เป็น 0 เมื่อ `consolidationPolicy=WhenEmptyOrUnderutilized`

การจัดการ disruption แบบใหม่ terminationGracePeriod

Cluster administrator ต้องการวิธีการบังคับใช้อายุการใช้งานสูงสุดของ node ภายใน Karpenter โดยตรง และให้สอดคล้องกับข้อกำหนดของ security requirements Karpenter จะ disrupts node โดยยึดตาม Pod Disruption Budgets (PDB), terminationGracePeriodSeconds ของ pod และ `karpenter.sh/do-not-disrupt` annotation หากการตั้งค่าเหล่านี้ถูกกำหนดค่าผิดพลาด Karpenter จะบล็อกไม่ให้ node ถูก disrupt ซึ่งป้องกันไม่ให้ cluster administrator นำ Amazon Machine Images (AMIs) ใหม่มาใช้

ดังนั้น จึงได้มีการแนะนำ `terminationGracePeriod` ขึ้นมา `terminationGracePeriod` คือเวลาสูงสุดที่ Karpenter จะ drain node ก่อนที่จะบังคับ terminate และจะไม่รอ node ที่จะมาทดแทนหลังจากที่ node หมดอายุแล้ว อายุการใช้งานสูงสุดของ node คือ `terminationGracePeriod + expireAfter` ในส่วนของการเปลี่ยนแปลงนี้ การกำหนดค่า `expireAfter` ได้ถูกย้ายจาก disruption block ไปยัง template spec

ในตัวอย่างต่อไปนี้ cluster administrator อาจกำหนดค่า `NodePool` เพื่อให้โหนดเริ่ม drain หลังจาก 30 วัน แต่มีระยะเวลา grace period 24 ชั่วโมงก่อนที่จะถูกบังคับ terminate เพื่อให้ workload ที่กำลังทำงานอยู่ (เช่น batch jobs ที่ทำงานเป็นเวลานาน) มีเวลาเพียงพอในการทำงานให้เสร็จสิ้นก่อนที่จะถูกยุติอย่างบังคับ

YAML

```yaml
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: default
spec:
  template:
    spec:
      terminationGracePeriod: 24h
      expireAfter: 720h
...
```

ยกเลิกการใช้งาน Drift feature gate

การ drift ของ Karpenter จะแทนที่ node ที่ drift ไปจากสถานะที่ต้องการ (เช่น การใช้ AMI ที่ outdated) ใน v1 การ drift ได้รับการ promote เป็น stable และ feature gate ได้ถูกยกเลิก ซึ่งหมายความว่าตอนนี้ node จะ drift โดย default หากผู้ใช้ปิดการใช้งาน feature gate ของ drift ใน v1beta1 ตอนนี้พวกเขาสามารถควบคุมการ drift ได้โดยใช้ disruption budgets by reason.

จำเป็นต้องกำหนด amiSelectorTerms

ใน Karpenter v1beta1 APIs เมื่อระบุ `amiFamily` โดยไม่มี `amiSelectorTerms` Karpenter จะอัปเดต node โดยอัตโนมัติผ่านการ drift เมื่อมีการเปิดตัวเวอร์ชันใหม่ของ Amazon EKS optimized AMI วิธีนี้ทำงานได้ดีในสภาพแวดล้อม pre-production ที่ทำการอัพเกรดไปยังเวอร์ชันล่าสุดอัตโนมัติเพื่อทดสอบ แต่อาจไม่เป็นที่ต้องการใน production ตอนนี้ Karpenter แนะนำให้ผู้ใช้ทำการ pin AMIs ใน production environment สามารถดูข้อมูลเพิ่มเติมเกี่ยวกับวิธีการจัดการ AMIs ได้ในเอกสารประกอบของ Karpenter

ตอนนี้ `amiSelectorTerms` ได้ถูกกำหนดให้เป็น require field และสามารถเริ่มใช้ `alias` ซึ่งประกอบด้วย AMI family และเวอร์ชัน (`family@version`) หากมี `alias` อยู่ใน `EC2NodeClass` Karpenter จะเลือก Amazon EKS optimized AMI สำหรับ family นั้น ด้วย feature ใหม่นี้ ผู้ใช้สามารถ pin ไปยังเวอร์ชันเฉพาะของ Amazon EKS optimized AMI ได้ Amazon EKS optimized AMI family ต่อไปนี้สามารถกำหนดค่าได้: `al2` , `al2023` , `bottlerocket` , `windows2019` , และ `windows2022` ตัวอย่างจะแสดงในเนื้อหาต่อ ๆ ไป

การใช้ Amazon EKS optimized AMIs

ในตัวอย่างนี้ Karpenter จัดเตรียม node ด้วย Amazon EKS optimized AMI สำหรับ Bottlerocket เวอร์ชัน 1.20.3 แม้ว่า AWS จะเปิดตัวเวอร์ชันใหม่กว่าของ Bottlerocket Amazon EKS optimized AMI โหนด worker ก็จะไม่เกิดการ drift

```yaml
apiVersion: karpenter.k8s.aws/v1
kind: EC2NodeClass
metadata:
  name: default
spec:
...
  amiSelectorTerms:
    - alias: bottlerocket@v1.20.3
...
```

การใช้ custom AMIs

ถ้า `EC2NodeClass` ไม่ได้ระบุคำเรียกแทน (`alias` term) ไว้ ทำให้ `amiFamily` จำเป็นต้องกำหนดว่าจะใช้ user data อะไร เราสามารถตั้งค่า `amiFamily` ได้ เช่น `AL2` , `AL2023` , `Bottlerocket` , `Windows2019` และ `Windows2022` เพื่อเลือก user data ที่สร้างไว้ล่วงหน้า หรือตั้งเป็น `Custom` หากผู้ใช้ต้องการกำหนด user data เอง คุณสามารถใช้ฟิลด์ที่มีอยู่ เช่น tags, name หรือ ID ใน `amiSelectorTerms` เพื่อเลือก AMI ตัวอย่างของ user data ที่ถูกกำหนดเข้าไปสามารถดูได้จากเอกสารของ Karpenter สำหรับ AMI families ที่ได้รับการปรับแต่งสำหรับ Amazon EKS

ในตัวอย่างต่อไปนี้ เราทำการให้ `EC2NodeClass` เลือก AMI ที่ผู้ใช้ระบุ (user-specified) ด้วย ID "ami-123" และใช้งาน Bottlerocket

```yaml
apiVersion: karpenter.k8s.aws/v1
kind: EC2NodeClass
metadata:
  name: default
spec:
...
  amiFamily: Bottlerocket
  amiSelectorTerms:
    - id: ami-123
...
```

ยกเลิกการเลือก Ubuntu AMI family

เริ่มตั้งแต่เวอร์ชัน 1 เป็นต้นไป AMI family ของ Ubuntu ได้ถูกยกลิกออกไปแล้ว หากต้องการใช้ AMI ของ Ubuntu ต่อไป คุณสามารถกำหนดค่า AMI ใน `amiSelectorTerms` โดยระบุ ID ล่าสุดของ Ubuntu AMI นอกจากนี้ คุณยังสามารถอ้างอิง `amiFamily: AL2` ใน `EC2NodeClass` ของคุณเพื่อรับการกำหนดค่า user data แบบเดียวกับที่คุณเคยทำก่อนหน้านี้ ต่อไปนี้เป็นตัวอย่าง:

```yaml
apiVersion: karpenter.k8s.aws/v1
kind: EC2NodeClass
metadata:
  name: default
spec:
...
  amiFamily: AL2
  amiSelectorTerms:
    - id: ami-123
...
```

จำกัดการเข้าถึง Instance Metadata Service จาก containers โดยค่าเริ่มต้น

การจำกัดไม่ให้ Pod เข้าถึง Instance profile ของ AWS Identity Access and Management (IAM) ที่ผูกติดกับ node เป็น Best practices ของ Amazon EKS เพื่อให้แน่ใจว่าแอปพลิเคชันของคุณมีเฉพาะสิทธิ์ที่จำเป็นเท่านั้น ไม่ใช่สิทธิ์ทั้งหมดของ node ดังนั้น โดยค่าเริ่มต้นสำหรับ `EC2NodeClass` ใหม่ การเข้าถึง Instance Metadata Service (IMDS) จะถูกบล็อกโดยการตั้งค่า hop count เป็น 1 (`httpPutResponseHopLimit:1`) และกำหนดให้ใช้ IMDSv2 (`httpTokens: required`) Pod ที่ใช้ Network mode แบบ Host ยังคงสามารถเข้าถึง IMDS ได้ ผู้ใช้ควรใช้ Amazon EKS Pod Identity หรือ IAM roles for service accounts เพื่อให้สิทธิ์กับ Pod ในการเข้าถึง AWS services

ย้าย kubelet configuration ไปยัง EC2NodeClass

Karpenter สามารถในการระบุ subset ของ kubelet arguments สำหรับการปรับแต่งเพิ่มเติม ใน Karpenter v1 การกำหนดค่า kubelet ได้ถูกย้ายไปยัง API ของ `EC2NodeClass` หากคุณได้กำหนดค่า kubelet แบบกำหนดเองและมีหลาย `NodePool` ที่มีการกำหนดค่า kubelet ที่แตกต่างกันแต่อ้างอิงไปยัง `EC2NodeClass` เดียวกัน ตอนนี้คุณจำเป็นต้องใช้หลาย `EC2NodeClass` ใน Karpenter v1 ตัว webhook ยังคงรักษาความเข้ากันได้นี้ไว้ได้ แต่อย่างไรก็ตาม ก่อนที่จะย้ายไปใช้ v1.1.x ผู้ใช้ต้องอัปเดต `NodePool` ให้อ้างอิงถึง `EC2NodeClass` ที่ถูกต้อง ซึ่งจะส่งผลใน node drifting

NodeClaims ไม่สามารถเปลี่ยนแปลงได้ (immutable)

Karpenter v1beta1 ไม่ได้บังคับใช้การ immutability กับ `NodeClaims` แต่จะ assume ว่าผู้ใช้จะไม่ดำเนินการอะไรที่ขัดแย้งหลังจากสร้างขึ้นแล้ว ดังนั้น ตอนนี้ `NodeClaims` จึง immutable เนื่องจากตัวควบคุม lifecycle ของ NodeClaim จะไม่ตอบสนองต่อการเปลี่ยนแปลงหลังจากการเริ่มต้น instance ครั้งแรก

กำหนดให้ต้องระบุทุกฟิลด์ nodeClassRef ของ NodePool และเปลี่ยนชื่อฟิลด์ apiVersion เป็น group

Karpenter v1beta1 ไม่ได้จำเป็นให้ผู้ใช้ต้องตั้งค่า `apiVersion` และชนิดของ `NodeClass` ที่พวกเขาอ้างอิงถึง ใน Karpenter v1 ผู้ใช้จำเป็นต้องตั้งค่าทุกฟิลด์ของ `nodeClassRef` นอกจากนี้ ฟิลด์ `apiVersion` ใน `nodeClassRef` ได้ถูกเปลี่ยนชื่อเป็น `group`

```yaml
...
  nodeClassRef:
    group: karpenter.k8s.aws
    kind: EC2NodeClass
    name: default
...
```

การเปลี่ยนแปลงของ Karpenter Prometheus metric

Karpenter มี metric หลายตัวที่พร้อมใช้งานใน Prometheus เพื่อให้สามารถตรวจสอบสถานะของKarpenter controller และสถานะของ cluster provisioning ได้ ในส่วนหนึ่งของการเปิดตัว Karpenter v1 metric จำนวนมากของ v1beta1 ได้มีการเปลี่ยนแปลง ดังนั้นสำหรับผู้ใช้ที่มี dashboards พร้อม queries ที่ใช้ metric เหล่านี้จะต้องได้รับการอัปเดต สำหรับรายการโดยละเอียดของการเปลี่ยนแปลง metric กรุณาตรวจสอบเอกสารการอัปเกรด Karpenter v1

แผนของการ deprecations

ในส่วนหนึ่งของ beta deprecations (การเลิกใช้งาน) ต่อไปนี้ได้ถูกลบออกใน v1:

- `karpenter.sh/do-not-evict` ที่เป็นการควบคุมแบบ pod-levelในเวอร์ชัน alpha การควบคุมนี้ถูกแทนที่ด้วย `karpenter.sh/do-not-disrupt` ซึ่งปิดการใช้งานการ disruption node ที่ pod กำลังทำงานอยู่ `karpenter.sh/do-not-evict` ถูกประกาศว่า deprecated ตลอดช่วง beta และถูกลบออกใน v1
- `karpenter.sh/do-not-consolidate`ที่เป็นการควบคุมแบบ pod-level ในเวอร์ชัน alpha การควบคุมนี้ถูกแทนที่ด้วย `karpenter.sh/do-not-disrupt` ซึ่งปิดการใช้งานการ disruption node แทนที่จะเป็นเพียงการรวม (consolidation) `karpenter.sh/do-not-consolidate` ถูกประกาศว่าเลิกใช้งานตลอดช่วง beta และถูกลบออกใน v1

- การกำหนดค่าแบบ ConfigMap จะ deprecated ใน v1beta1 และถูกลบออกทั้งหมดใน v1 การกำหนดค่านี้ถูกแทนที่ด้วยการกำหนดค่าแบบ CLI/environment variable based configuration
- `karpenter.sh/managed-by` ซึ่งเก็บชื่อ cluster ถูกแทนที่ด้วย `eks:eks-cluster-name`

สำหรับรายการเต็มของฟีเจอร์ใหม่ การเปลี่ยนแปลง และการเลิกใช้งาน กรุณาอ่านบันทึกการเปลี่ยนแปลง (changelog)

การ Migration

เนื่องจาก v1 APIs สำหรับ Karpenter ที่ไม่ได้ส่งผลให้เกิดการเปลี่ยนแปลงกับ API group หรือ resource ทำให้สามารถใช้ Kubernetes webhook conversion process เพื่ออัปเกรด API แบบ in-place โดยไม่ต้อง roll node ใหม่ ก่อนการอัปเกรด คุณต้องอยู่บน Karpenter เวอร์ชัน (0.33.0+) ที่รองรับ API v1beta1 เช่น `NodePool` , `NodeClaim` , และ `EC2NodeClass`

สรุปกระบวนการอัปเกรดจาก beta เป็น v1 มีดังนี้

1. ใช้ CRD ของ NodePool, NodeClaim และ EC2NodeClass เวอร์ชัน v1 ที่อัปเดตแล้ว
2. อัปเกรดตัวควบคุม Karpenter เป็นเวอร์ชัน v1.0.0 เวอร์ชันนี้ของ Karpenter เริ่มทำงานกับ API v1 schema Resource จะถูกแปลงจากเวอร์ชัน v1beta1 เป็น v1 โดยอัตโนมัติ โดยใช้ conversion webhooks ของ Karpenter และผู้ให้บริการ (สำหรับการเปลี่ยนแปลง `EC2NodeClass`)
3. ต่อไปและก่อนการอัปเกรดเป็น Karpenter v1.1.0 ผู้ใช้ต้องอัปเดต v1beta1 manifests ของตนเพื่อใช้เวอร์ชัน v1 ใหม่ ทำงานกับการเปลี่ยนแปลง API นี้ได้ ดูรายละเอียดเพิ่มเติมได้ที่ส่วน “before upgrading to Karpenter v1.1.0” ใน เอกสารการ v1 migration

สำหรับขั้นตอนการอัปเกรดโดยละเอียด โปรดดูเอกสาร Karpenter v1 migration

บทสรุป

ในบทความนี้ คุณได้เรียนรู้เกี่ยวกับการเปิดตัว Karpenter 1.0.0 และสรุปฟีเจอร์ใหม่และการเปลี่ยนแปลงต่างๆ ก่อนที่คุณจะอัปเกรด Karpenter เป็น v1.0.0 เราขอแนะนำให้คุณอ่านเอกสารการย้าย Karpenter v1 ฉบับเต็ม และทดสอบกระบวนการอัปเกรดของคุณใน non-production หากคุณมีคำถามหรือข้อเสนอแนะ สามารถติดต่อได้ที่ช่อง #karpenter channel บน Kubernetes Slack หรือบน GitHub ซึ่งคุณสามารถแบ่งปันความคิดเห็นได้

TAGS: Amazon EKS