Running Arm nodes with AWS Graviton on Amazon EKS

Michael Hausenblas, Product Developer Advocate
2020-08-17
Table of contents

• Why bother?
• Multi-arch Apps
• Arm-support in Amazon EKS
• Demo
Why bother?
Benefits

- Best price performance
- Extensive ecosystem support
- Enhanced security for cloud applications

20% lower cost and up to 40% higher performance for M6g, C6g, and R6g instances over M5, C5, and R5 instances, based on internal testing of workloads.

<table>
<thead>
<tr>
<th>General Purpose</th>
<th>Compute Optimized</th>
<th>Memory Optimized</th>
</tr>
</thead>
<tbody>
<tr>
<td>M6g</td>
<td>C6g</td>
<td>R6g</td>
</tr>
<tr>
<td>Best price performance for general purpose workloads with balanced compute, memory, and networking.</td>
<td>Best price performance for compute-intensive workloads.</td>
<td>Best price performance for workloads that process large data sets in memory.</td>
</tr>
<tr>
<td><strong>Built for</strong>: General-purpose workloads such as application servers, mid-size data stores, microservices, and cluster computing.</td>
<td><strong>Built for</strong>: Compute-intensive applications such as high performance computing, video encoding, gaming, and CPU-based machine learning inference acceleration.</td>
<td><strong>Built for</strong>: Memory-intensive workloads such as open-source databases (MySQL, MariaDB, and PostgreSQL), or in-memory caches (Redis, KeyDB, Memcached).</td>
</tr>
</tbody>
</table>
Multi-arch Apps
Arm-support in Amazon EKS
AWS Graviton2 for Amazon EKS is GA!

- Supporting Arm V8.2 (and others) architectures
- Along with multi-arch support in Amazon ECR, end-to-end coverage
- Amazon EKS & tooling taking care of architecture-specific config
- Mixed managed node groups now supported
- Getting started: aws/aws-graviton-getting-started
Arm in Action
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: plone
spec:
  replicas: 1
  selector:
    matchLabels:
      app: plone
  template:
    metadata:
      labels:
        app: plone
    spec:
      containers:
        - name: main
          image: arm64v8/plone:5.2.1
          ports:
            - containerPort: 8080

---

apiVersion: v1
kind: Service
metadata:
  name: plone
spec:
  ports:
    - port: 80
      targetPort: 8080
  selector:
    app: plone

$ kubectl port-forward svc/plone 8888:80
```
Questions? Suggestions?

Ask on the chat or visit our virtual booth during KubeCon EU 2020!

Twitter: @mhausenblas
Mail: hausenbl@amazon.com