Amazon EC2 :: Optimize EKS cost with Spot & A1 Instances

Arthur Petitpierre – arthurpt@amazon.com
Specialist Solutions Architect – EC2 A1 Instances

November 2019
Who am I?

Arthur Petitpierre – arthurpt@amazon.com / @ArthurPtP
Specialist Solutions Architect – A1 Instances @AWS

Previously:
• HPC Specialist SA @AWS
• HPC Services CTO @ATOS
• And a few other stuffs…

Occasionally deliver Snowballs around Paris Seattle on a cargo-bike
Amazon EC2 purchase options

On-Demand
Pay for compute capacity the second with no long-term commitments

Reserved Instances
Make a 1 or 3-year commitment and receive a significant discount off On-Demand prices

Spiky workloads, to define needs

Committed & steady-state usage

Spot Instances
Spare EC2 capacity at savings of up to 90% off On-Demand prices

Fault-tolerant, flexible, stateless workloads
Spot is easy

Price changes infrequently based on *long term* supply and demand of spare capacity in each pool independently.

Just request capacity and pay the current rate. No Bidding.

Interruptions only happen when OD needs capacity. No outbidding.
### Large customer base

<table>
<thead>
<tr>
<th>Research</th>
<th>Consumer apps</th>
<th>AdTech &amp; MarTech</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caltech</td>
<td>yahoo</td>
<td>ironSource</td>
</tr>
<tr>
<td>CLEMSON UNIVERSITY</td>
<td>mercado libre</td>
<td>LKQD</td>
</tr>
<tr>
<td>illumina</td>
<td>Aol.</td>
<td>codewise</td>
</tr>
<tr>
<td>OpenAI</td>
<td>reddit</td>
<td>AdRoll</td>
</tr>
<tr>
<td>SevenBridges</td>
<td>twitter</td>
<td>NetSeer</td>
</tr>
<tr>
<td>NASA</td>
<td>Pinterest</td>
<td>Quantcast</td>
</tr>
<tr>
<td>HUMAN LONGEVITY, INC</td>
<td>lyra</td>
<td>bloomreach</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMPLEX</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sports, media, &amp; entertainment</th>
<th>B2B enterprise tech</th>
<th>Financial services</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMCAST</td>
<td>Autodesk</td>
<td>Finra</td>
</tr>
<tr>
<td>FUSEFX</td>
<td>HGST</td>
<td>CoreLogic</td>
</tr>
<tr>
<td>ESPN</td>
<td>red violet</td>
<td></td>
</tr>
<tr>
<td>Theory Studios</td>
<td>LOTAME</td>
<td></td>
</tr>
<tr>
<td>SONY</td>
<td>Mapbox</td>
<td></td>
</tr>
<tr>
<td>Nexon</td>
<td>zuora</td>
<td></td>
</tr>
<tr>
<td>scrippsnetworks Initiative</td>
<td></td>
<td>AIG</td>
</tr>
</tbody>
</table>

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
Is my workload Spot Ready?

Stateless

Fault-Tolerant

Flexible: Multi-AZ and Instance Flexibility

Loosely Coupled

Looks familiar?
Spot pricing model change – no more bidding

New smooth pricing
November 2017
## EC2 Spot pools – instance flexibility

<table>
<thead>
<tr>
<th>Instance Size</th>
<th>1a</th>
<th>1b</th>
<th>1c</th>
<th>On Demand</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>8XL</td>
<td>$0.50</td>
<td>$0.27</td>
<td>$0.29</td>
<td>$1.76</td>
<td></td>
</tr>
<tr>
<td>4XL</td>
<td>$0.21</td>
<td>$0.30</td>
<td>$0.16</td>
<td>$0.88</td>
<td></td>
</tr>
<tr>
<td>2XL</td>
<td>$0.08</td>
<td>$0.07</td>
<td>$0.08</td>
<td>$0.44</td>
<td></td>
</tr>
<tr>
<td>XL</td>
<td>$0.04</td>
<td>$0.05</td>
<td>$0.04</td>
<td>$0.22</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>$0.01</td>
<td>$0.01</td>
<td>$0.04</td>
<td>$0.11</td>
<td></td>
</tr>
</tbody>
</table>

- Each instance family
- Each instance size
- Each Availability Zone (69)
- In every region (22)
- Is a separate **Spot pool**

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
## Monitoring Spot usage – Savings Summary

A high-level summary of your savings across all of your **running** Spot Instances. For detailed reporting on your account-level Spot usage, visit Cost Explorer.

### Spot usage and savings

<table>
<thead>
<tr>
<th>Spot Instances</th>
<th>vCPU-hours</th>
<th>mem(GiB)-hours</th>
<th>On-Demand total</th>
<th>Spot total</th>
<th>Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td>178</td>
<td>487.5</td>
<td>$9.54</td>
<td>$2.66</td>
<td>72%</td>
</tr>
</tbody>
</table>

*Average cost per vCPU-hour: $0.0150
*Average cost per mem(GiB)-hour: $0.0055

### Details

<table>
<thead>
<tr>
<th>Instance Type</th>
<th>vCPU-hours</th>
<th>mem(GiB)-hours</th>
<th>Total Cost</th>
<th>Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>c3.large (2)</td>
<td>88</td>
<td>165</td>
<td>$1.37 total</td>
<td>74% savings</td>
</tr>
<tr>
<td>c4.large (6)</td>
<td>12</td>
<td>22.5</td>
<td>$0.20 total</td>
<td>71% savings</td>
</tr>
<tr>
<td>c5.large (3)</td>
<td>6</td>
<td>12</td>
<td>$0.11 total</td>
<td>63% savings</td>
</tr>
<tr>
<td>m5.large (6)</td>
<td>12</td>
<td>48</td>
<td>$0.22 total</td>
<td>66% savings</td>
</tr>
<tr>
<td>r5.large (6)</td>
<td>12</td>
<td>96</td>
<td>$0.23 total</td>
<td>73% savings</td>
</tr>
<tr>
<td>t2.medium (12)</td>
<td>24</td>
<td>48</td>
<td>$0.18 total</td>
<td>70% savings</td>
</tr>
<tr>
<td>t2.large (12)</td>
<td>24</td>
<td>96</td>
<td>$0.36 total</td>
<td>70% savings</td>
</tr>
</tbody>
</table>

* Spot savings are estimated savings and may differ from actual savings. This is because the savings shown on this page do not include the billing adjustments for your usage.
What about interruptions?

**Minimal interruptions**
Over 95% of the instances were not interrupted in the last 3 months

The work you are doing to make your applications fault-tolerant also benefits Spot

Spot is optimized for stateless, fault-tolerant, or flexible workloads.

Any application that can can have part or all of the work, paused and resumed or restarted, can use Spot.

Check for 2-minute instance termination notice via instance metadata or CloudWatch Events and *automate by*:

- Checkpointing
- Draining from ELB
- Using stop-start and hibernate to restart faster
EC2 Spot with Amazon EKS

- Run a DaemonSet on every worker to catch the Spot interruption and coordon & drain the node
- Use labels to identify Spot nodes (for the DaemonSet, and other purposes – affinity & tolerations?)
The new ASG combines purchase options, instance types and AZs in a single ASG.

m4.large Spot Instances
m5.large Spot Instances
c4.xlarge On-Demand instances

Single ASG
Main takeaways for success with Spot

• Be instance type agnostic and let ASG/Fleet provide the required capacity at the lowest price
• Adopt Launch Templates to benefit from new ASG and Fleet features
• New instance families generally have higher interruption rates
• Architect for fault-tolerance to be Spot compatible and increase your availability
EC2 Spot instance Workshops:
https://ec2spotworkshops.com
Choice of processors and architectures

Intel
Intel Xeon Scalable (Skylake) processor

AMD
AMD EPYC processor

AWS
AWS Graviton processor
64-bit Arm

Choice of GPUs and FPGAs for compute acceleration

Right compute for each application and workload
First instance powered by AWS Graviton Processor

Amazon EC2 A1
Run scale-out and Arm-based applications in the cloud

Up to 45% cost savings
AWS Graviton Processor
64-bit Arm Neoverse cores and custom AWS silicon

Flexibility and choice for your workloads
Lower cost
Maximize resource efficiency with AWS Nitro System
Target applications for Amazon EC2 A1

**Web tier**
- Containerized microservices

**Caching fleets**
- IoT, Gaming, Arm workloads
Arm software ecosystem for A1 instances

**OSVs and ISVs**
- Amazon Linux 2
- Ubuntu 16.04, 18.04 and newer
- Red Hat Enterprise Linux 7.6 and 8.0
- SUSE Linux Enterprise Server for Arm 15
- Docker Desktop Community and Docker Enterprise Engine (Beta)
  - Added since launch: Fedora Rawhide, Fedora Atomic, Debian 10, and Ubuntu 18.10 (Bionic)
  - More coming soon.

**Containers**
- Most Docker official images support arm64

**Amazon ECS**
- Available today!

**Amazon EKS**
- Public Preview!

**Tools**
- AWS Marketplace
- AWS Systems Manager
- Amazon CloudWatch
- AWS CodeCommit
- AWS Cloud9
- AWS CodePipeline
- Amazon Inspector
- AWS Batch
- + Amazon Corretto (OpenJDK)
Amazon EKS A1 Instances Preview Program

Start here to participate in the Amazon EC2 A1 instance preview program for Amazon Elastic Container Service for Kubernetes (EKS). Using the instructions and code in this repository you can run containers using EC2 A1 instances on a Kubernetes cluster that is managed by Amazon EKS.

Amazon EC2 A1 instances deliver significant cost savings for scale-out and Arm-based applications such as web servers, containerized microservices, caching fleets, and distributed data stores.

Note: The assets and instructions in this repository folder are offered as part of a public preview program administered by AWS.

Using the instructions and assets in this repository folder is governed as a preview program under the AWS Service Terms.
Blog shout out - Cost optimization for Kubernetes on AWS

This post was contributed by AWS Container Hero, Casey Lee, Director of Engineering for Liatrio

https://aws.amazon.com/blogs/containers/cost-optimization-for-kubernetes-on-aws/

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
Demo: Docker Desktop
x86_64/arm64 build
Demo goals

Show how seamless it is to build bi-arch containers and use them in a simple web application

What will we leverage?
- Docker buildx
- AWS CloudFormation
- AWS Elastic Load Balancer
- Amazon EC2 A1 and T3 Instances
Demo Architecture
Show me the code!
Wrap-up

What have we learnt?
- There's an easy transition path from x86_64 to arm64
- Bi-arch containers are easy to build with buildx
- AWS EC2 A1 instances are yet another cost reduction lever
Thank you

Arthur Petitpierre – arthurpt@amazon.com
@ArthurPtP