Vending Machine
for Data Science Experiments

Christina Andonov
Senior Specialist Solutions Architect
AWS

Apoorva Kulkarni
Senior Specialist Solutions Architect
AWS
AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
AWS_SECRET_ACCESS_KEY=wJalrXUtFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Priority #1

Priority #2
Ray on Kubernetes

Source: https://docs.ray.io/en/latest/cluster/kubernetes/index.html
Deploy Ray on Kubernetes

1. Deploy a **KubeRay Operator**

```bash
helm repo add kuberay https://ray-project.github.io/kuberay-helm/
helm repo update

# Install both CRDs and KubeRay operator v1.0.0.
helm install kuberay-operator kuberay/kuberay-operator --version 1.0.0

# Confirm that the operator is running in the namespace 'default'.
kubectl get pods

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>kuberay-operator-7fbdbf8c89-pt8bk</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>27s</td>
</tr>
</tbody>
</table>
```

2. Deploy a **RayCluster** custom resource

```bash
# Deploy a sample RayCluster CR from the KubeRay Helm chart repo:
helm install raycluster kuberay/ray-cluster --version 1.0.0

# Once the RayCluster CR has been created, you can view it by running:
kubectl get rayclusters

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESIRED WORKERS</th>
<th>AVAILABLE WORKERS</th>
<th>STATUS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>raycluster-kuberay</td>
<td>1</td>
<td>1</td>
<td>ready</td>
<td>72s</td>
</tr>
</tbody>
</table>
```

*Source: https://docs.ray.io/en/latest/cluster/kubernetes/getting-started/raycluster-quick-start.html*
Data Scientists

Jupiter Notebook

Ray Cluster

CPUs
Data Scientists

Jupyter Notebook

Ray Cluster

CPUs → GPUs
## Instance type

**Accelerated Computing**

### Viewing 43 of 698 available instances

<table>
<thead>
<tr>
<th>Instance name</th>
<th>On-Demand hourly rate</th>
<th>vCPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>inf1.xlarge</td>
<td>$0.228</td>
<td>4</td>
</tr>
<tr>
<td>inf1.2xlarge</td>
<td>$0.362</td>
<td>8</td>
</tr>
<tr>
<td>g4ad.xlarge</td>
<td>$0.37853</td>
<td>4</td>
</tr>
<tr>
<td>g4dn.xlarge</td>
<td>$0.526</td>
<td>4</td>
</tr>
<tr>
<td>g4ad.2xlarge</td>
<td>$0.54117</td>
<td>8</td>
</tr>
</tbody>
</table>
How much is p5.48xlarge?

$98.32/hour

$70K/month
<table>
<thead>
<tr>
<th>T-Shirt Size</th>
<th>Instance Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>g5.2xlarge</td>
</tr>
<tr>
<td>L</td>
<td>p4d.24xlarge</td>
</tr>
<tr>
<td>XL</td>
<td>p5.48xlarge</td>
</tr>
<tr>
<td>T-Shirt Size</td>
<td>Instance Size</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>M</td>
<td>g5.2xlarge</td>
</tr>
<tr>
<td></td>
<td>g5.8xlarge</td>
</tr>
<tr>
<td></td>
<td>g5.12xlarge</td>
</tr>
<tr>
<td>L</td>
<td>p4d.24xlarge</td>
</tr>
<tr>
<td>XL</td>
<td>p5.48xlarge</td>
</tr>
</tbody>
</table>
https://s12d.com/ca-to-karpenter
<table>
<thead>
<tr>
<th>T-Shirt Size</th>
<th>Instance Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>g5.2xlarge</td>
</tr>
<tr>
<td></td>
<td>g5.8xlarge</td>
</tr>
<tr>
<td></td>
<td>g5.12xlarge</td>
</tr>
<tr>
<td>L</td>
<td>p4d.24xlarge</td>
</tr>
<tr>
<td>XL</td>
<td>p5.48xlarge</td>
</tr>
<tr>
<td>T-Shirt Size</td>
<td>Instance Size</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
</tr>
<tr>
<td>M</td>
<td>g5.2xlarge</td>
</tr>
<tr>
<td></td>
<td>g5.8xlarge</td>
</tr>
<tr>
<td></td>
<td>g5.12xlarge</td>
</tr>
<tr>
<td>L</td>
<td>p4d.24xlarge</td>
</tr>
<tr>
<td>XL</td>
<td>p5.48xlarge</td>
</tr>
</tbody>
</table>
Karpenter

T-Shirt Size | Instance Size
---|---
M | g5.2xlarge
| g5.8xlarge
| g5.12xlarge
L | p4d.24xlarge
XL | p5.48xlarge

```yaml
{{- if eq .Values.experiment_size "medium" }}
apiversion: karpenter.sh/v1beta1
kind: NodePool
spec:
  requirements:
    - key: node.kubernetes.io/instance-type
      operator: In
      values: ["g5.4xlarge", "g5.8xlarge", "g5.12xlarge"]
    - key: karpenter.sh/capacity-type
      operator: In
      values: ["spot", "on-demand"]
{{- end }}

{{- if eq .Values.experiment_size "large" }}
apiversion: karpenter.sh/v1beta1
kind: NodePool
spec:
  requirements:
    - key: node.kubernetes.io/instance-type
      operator: In
      values: ["p4d.24xlarge"]
    - key: karpenter.sh/capacity-type
      operator: In
      values: ["spot", "on-demand"]
{{- end }}

{{- if eq .Values.experiment_size "xlarge" }}
apiversion: karpenter.sh/v1beta1
kind: NodePool
spec:
  requirements:
    - key: node.kubernetes.io/instance-type
      operator: In
      values: ["p5.48xlarge"]
    - key: karpenter.sh/capacity-type
      operator: In
      values: ["spot", "on-demand"]
{{- end }}
```
Team A

*experiment_name*: "team-a-exp1"
*experiment_size*: "medium"
*principal_arn*: "team-a-role"

```
apiVersion: karpenter.sh/v1beta1
group: karpenter.sh
customResourceDefinition:
  name: nodepool
kind: NodePool
spec:
  requirements:
  - key: node.kubernetes.io/instance-type
    operator: In
    values: ["g5.4xlarge", "g5.8xlarge", "g5.12xlarge"]
  - key: karpenter.sh/capacity-type
    operator: In
    values: ["spot", "on-demand"]
```

Team B

*experiment_name*: "team-b-exp1"
*experiment_size*: "medium"
*principal_arn*: "team-b-role"

```
apiVersion: karpenter.sh/v1beta1
group: karpenter.sh
customResourceDefinition:
  name: nodepool
kind: NodePool
spec:
  requirements:
  - key: node.kubernetes.io/instance-type
    operator: In
    values: ["g5.4xlarge", "g5.8xlarge", "g5.12xlarge"]
  - key: karpenter.sh/capacity-type
    operator: In
    values: ["spot", "on-demand"]
```
```python
# Specify required resources for an actor.
@ray.remote(num_cpus=28, num_gpus=4)
class Actor:
    pass
```

```
workerGroupSpecs:
    ...
    resources:
        limits:
            cpu: "7"
            memory: "30G"
            nvidia.com/gpu: 1
```

<table>
<thead>
<tr>
<th>Instance</th>
<th>GPU</th>
<th>CPU</th>
<th>Mem</th>
</tr>
</thead>
<tbody>
<tr>
<td>g5.2xlarge</td>
<td>1</td>
<td>8</td>
<td>32</td>
</tr>
<tr>
<td>g5.8xlarge</td>
<td>1</td>
<td>32</td>
<td>128</td>
</tr>
<tr>
<td>g5.12xlarge</td>
<td>4</td>
<td>64</td>
<td>192</td>
</tr>
</tbody>
</table>

```
g5.2xlarges
```

Team A ➔ Jupiter Notebook ➔ Ray Cluster ➔ Karpenter NodePool

AWS

© 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.
# Specify required resources for an actor.

```python
@ray.remote(num_cpus=28, num_gpus=4)
class Actor:
    pass
```

<table>
<thead>
<tr>
<th>Instance</th>
<th>GPU</th>
<th>CPU</th>
<th>Mem</th>
</tr>
</thead>
<tbody>
<tr>
<td>g5.2xlarge</td>
<td>1</td>
<td>8</td>
<td>32</td>
</tr>
<tr>
<td>g5.8xlarge</td>
<td>1</td>
<td>32</td>
<td>128</td>
</tr>
<tr>
<td>g5.12xlarge</td>
<td>4</td>
<td>64</td>
<td>192</td>
</tr>
</tbody>
</table>

workerGroupSpecs:
- replicas: 0
- minReplicas: 0
- maxReplicas: 10
resource "aws_s3_bucket" "my_bucket" {
  bucket_prefix = "my-bucket-"
  tags = local.tags
}

resource "aws_s3_bucket_public_access_block" "my_bucket" {
  bucket = aws_s3_bucket.my_bucket.id
  block_public_acls = true
  block_public_policy = true
  ignore_public_acls = true
  restrict_public_buckets = true
}

resource "aws_s3_bucket_lifecycle_configuration" "my_bucket" {
  bucket = aws_s3_bucket.my_bucket.id
  rule {
    id = "some-data"
    expiration {
      days = 7
    }
    status = "Enabled"
  }
}

resource "aws_s3_bucket_ownership_controls" "my_bucket" {
  bucket = aws_s3_bucket.my_bucket.id
  rule {
    object_ownership = "BucketOwnerPreferred"
  }
}

resource "aws_s3_bucket_acl" "my_bucket" {
  depends_on = [aws_s3_bucket_ownership_controls.my_bucket]
  bucket = aws_s3_bucket.my_bucket.id
  acl = "private"
}

data "aws_iam_policy_document" "assume_policy" {
  statement {
    effect = "Allow"
    actions = ["sts:AssumeRoleWithWebIdentity"]
    principals {
      type = "Federated"
      identifiers = ["arn:aws:iam::1111222333:oidc-provider/oidc.eks.us-east-1.amazonaws.com/id/1234567890ABC"]
    }
    condition = {
      test = "StringLike"
      values = [""]
    }
  }
}

data "aws_iam_policy_document" "my_policy" {
  statement {
    sid = "S3ReadAccessToBucket"
    effect = "Allow"
    actions = ["s3:ListBucket", "s3:GetObject"
    resources = [aws_s3_bucket.my_bucket.arn, "$(aws_s3_bucket.my_bucket.arn)/**"]
  }
}

resource "aws_iam" "my_policy" {
  name = "my-policy"
  policy = data.aws_iam_policy_document.my_policy.json
}

resource "aws_iam_role" "my_role" {
  name = "my-role"
  assume_role_policy = data.aws_iam_policy_document.assume_role.json
}

resource "aws_iam_role_policy_attachment" "policy-attach" {
  role = aws_iam_role.my_role.name
  policy_arn = aws_iam_policy.my_policy.arn
}
module "s3" {
  source = "my-org-tf/modules/s3"
  version = "1.15.1"

  bucket_prefix       = "my-bucket-"
  lifecycle_expiration = 7
  object_ownership    = "bucketOwnerPreferred"
  acl                 = "private"
...}

S3
Policy
Role
experiment_name: "team-a-expl"
experiment_size: "medium"
principal_arn: "team-a-role"
Demo
Supercharge your Data and AI/ML Journey with Amazon EKS

Let's Spin Up

AI/ML
Unlocking Best Practices for AI/ML Deployment on EKS with KubeFlow, JupyterHub, and More

Data Analytics
Best Practice Data Analytics Deployment Templates and Examples for EKS with Apache Spark, Spark Operator, Dask, Beam, and More

Amazon EMR on EKS
Optimized Multi-Tenant Deployment of Amazon EMR on EKS Cluster with Best Practices using Karpenter Autoscaler and Apache Yunikorn Templates

https://s12d.com/DoEKS
Thank you!

Christina Andonov
candonov@amazon.com
LinkedIn: christina-andonov

Apoorva Kulkarni
kuapoovr@amazon.com
LinkedIn: askulkarni-aws