aws

Guia de desenvolvimento do Amazon EMR no EKS

Amazon EMR

Amazon EMR: Guia de desenvolvimento do Amazon EMR no EKS

Table of Contents

O que é o Amazon EMR no EKS? 1
Arquitetura 2
Conceitos 3
Namespace do Kubernetes 3
Cluster virtual 3
Execução de trabalho 4
Contêineres do Amazon EMR 4
Como os componentes funcionam em conjunto 4
Conceitos básicos 6
Execução de uma aplicação do Spark 7
Práticas recomendadas 13
Segurança 13
Envio de trabalho para o Pyspark 13
Armazenamento 13
Integração com o metastore 14
Depuração 14
Solução de problemas do Amazon EMR no EKS 14
Posicionamento de nó 14
Desempenho 14
Otimização de custo 15
Usar o AWS Outposts 15
Personalização de imagens do Docker 16
Como personalizar imagens do Docker 16
Pré-requisitos 17
Etapa 1: recuperar uma imagem base do Amazon Elastic Container Registry (Amazon ECR) 17
Etapa 2: personalizar uma imagem base 18
Etapa 3: (opcional, mas recomendada) validar uma imagem personalizada 19
Etapa 4: publicar uma imagem personalizada 20
Etapa 5: enviar uma workload do Spark no Amazon EMR usando uma imagem personalizada 21
Personalização de imagens do Docker para endpoints interativos 24
Trabalho com imagens de múltiplas arquiteturas 25
Como selecionar um URI de imagem base 27

Contas de registro do Amazon ECR ... 28
Considerações ... 29
Execução de trabalhos do Flink ... 31
Operador do Kubernetes para Flink ... 32
Configuração ... 32
Conceitos básicos ... 34
Execução de uma aplicação do Flink ... 35
Segurança ... 37
Desinstalação do operador ... 39
Kubernetes nativo ... 40
Configuração ... 41
Conceitos básicos ... 41
Requisitos de segurança ... 44
Monitorar ... 45
Uso do Amazon Managed Service for Prometheus ... 45
Uso da interface do usuário do Flink ... 47
Uso da configuração de monitoramento ... 48
Uso da alta disponibilidade ... 53
Uso do Autoscaler ... 59
Solução de problemas ... 61
Solução de problemas do Apache Flink para Amazon EMR no EKS ... 61
Versões compatíveis ... 62
Execução de trabalhos do Spark ... 64
StartJobRun ... 64
Configuração ... 65
Conceitos básicos ... 88
Operador do Spark ... 91
Configuração ... 91
Conceitos básicos ... 92
Autoescalabilidade vertical ... 95
Desinstalar ... 99
Segurança ... 100
spark-submit ... 110
Configuração ... 110
Conceitos básicos ... 111
Segurança ... 112

Gerenciamento de execuções de trabalhos ... 113
Gerenciamento com a CLI ... 113
Execução de scripts do Spark SQL ... 119
Estados de execução de trabalho ... 122
Visualização de trabalhos no console ... 122
Erros comuns na execução de trabalhos ... 123
Uso da classificação de envio de trabalho ... 129
Visão geral ... 129
Exemplos ... 129
Uso de modelos de trabalho ... 133
Criação e uso de um modelo de trabalho para iniciar uma execução de trabalho ... 133
Definição de parâmetros de modelos de trabalhos ... 135
Controle do acesso aos modelos de trabalhos ... 137
Uso de modelos de pod ... 139
Cenários comuns ... 139
Habilitação de modelos de pod com o Amazon EMR no EKS ... 141
Campos do modelo de pod ... 144
Considerações sobre contêineres sidecar ... 147
Uso de políticas de repetição ... 149
Definição de uma política de repetição ... 149
Recuperação de um status da política ... 151
Monitoramento do trabalho ... 152
Descoberta de logs de drivers ... 152
Uso da alternância de log de eventos do Spark ... 153
Uso da alternância de log do contêiner do Spark ... 154
Uso da escalabilidade automática vertical ... 156
Configuração ... 156
Conceitos básicos ... 159
Configuração ... 161
Monitoramento das recomendações ... 167
Como desinstalar ... 168
Execução de workloads interativas ... 169
Visão geral dos endpoints interativos ... 169
Pré-requisitos para os endpoints interativos ... 171
AWS CLI ... 172
eksctl ... 172

Cluster do Amazon EKS 172
Concessão de acesso ao cluster 173
Ativação de perfis do IAM para contas de serviço 173
Criação de um perfil de execução de trabalho do IAM 173
Concessão de acesso para os usuários 173
Registro do cluster do Amazon EKS com o Amazon EMR 174
Load Balancer Controller 174
Criação de um endpoint interativo 174
Criação de um endpoint interativo 174
Especificação de parâmetros personalizados 175
..... 176
Parâmetros para um endpoint interativo 177
Definição de configurações para endpoints interativos 178
Monitorar trabalhos Spark 178
Modelos de pod personalizados 180
Implantação de um pod do JEG em um grupo de nós 180
Opções de configuração do JEG 184
Modificação de parâmetros do PySpark 185
Imagem de kernel personalizada 186
Monitoramento de endpoints interativos 187
Exemplos 190
Uso de cadernos Jupyter de hospedagem própria 190
Crie um grupo de segurança 191
Criação de um endpoint interativo 191
Obtenção do URL do servidor de gateway 192
Obtenção do token de autenticação 192
Implantação de um caderno 193
Limpeza 198
Outras operações 199
..... 199
Listagem de endpoints interativos 201
Exclusão de um endpoint interativo 202
Monitorar trabalhos 203
Monitoramento de trabalhos com o Amazon CloudWatch Events 203
Automatização do Amazon EMR no EKS com o CloudWatch Events 204
Exemplo: configuração de uma regra que invoque o Lambda 205

Monitoramento do pod de drivers do trabalho com uma política de repetição usando o Amazon CloudWatch Events ... 206
Gerenciamento de clusters virtuais ... 207
Criação de um cluster virtual ... 207
Listagem de clusters virtuais ... 209
Descrição de um cluster virtual ... 209
Exclusão de um cluster virtual ... 209
Estados de um cluster virtual ... 209
Tutoriais ... 210
Uso do Delta Lake ... 210
Uso do Iceberg ... 211
Uso do RAPIDS para Spark ... 212
Uso do Spark no Redshift ... 216
Iniciar uma aplicação do Spark ... 217
Autenticação no Amazon Redshift ... 218
Leitura e gravação para o Amazon Redshift ... 220
Considerações ... 222
Uso do Volcano ... 223
Visão geral ... 223
Instalação ... 223
Envio: operador do Spark ... 225
Envio: `spark-submit` ... 227
Uso do YuniKorn ... 228
Visão geral ... 228
Crie seu cluster do ... 228
Instalação do YuniKorn ... 230
Envio: operador do Spark ... 231
Envio: `spark-submit` ... 234
Segurança ... 13
Práticas recomendadas ... 237
Aplicação do princípio de privilégio mínimo ... 237
Listagem de controle de acesso para endpoints ... 237
Obtenção das atualizações de segurança mais recentes para as imagens personalizadas ... 238
Limitação do acesso à credencial do pod ... 238
Isolamento de código de uma aplicação não confiável ... 238
Permissões de controle de acesso por perfil (RBAC) ... 238

Restrição do acesso às credenciais do perfil do IAM ou do perfil de instância do grupo de nós ... 239
Proteção de dados ... 240
Criptografia em repouso ... 241
Criptografia em trânsito ... 243
Gerenciamento de identidade e acesso ... 244
Público ... 245
Como autenticar com identidades ... 245
Gerenciamento do acesso usando políticas ... 249
Como o Amazon EMR no EKS funciona com o IAM ... 252
Uso de funções vinculadas a serviço ... 258
Políticas gerenciadas para o Amazon EMR no EKS ... 262
Uso de perfis de execução de trabalho com o Amazon EMR no EKS ... 263
Exemplos de políticas baseadas em identidade ... 266
Políticas para controle de acesso baseado em etiquetas ... 269
Solução de problemas ... 272
Registro e monitoramento ... 274
Logs do CloudTrail ... 274
Validação de conformidade ... 277
Resiliência ... 277
Segurança da infraestrutura ... 278
Análise de configuração e vulnerabilidade ... 278
Endpoints da VPC de interface ... 278
Criação de uma política de endpoint da VPC para o Amazon EMR no EKS ... 279
Acesso entre contas ... 282
Pré-requisitos ... 283
Como acessar um bucket do Amazon S3 ou uma tabela do DynamoDB entre contas ... 283
Marcar recursos ... 288
Conceitos básicos de tags ... 288
Marcar com tag os recursos do ... 289
Restrições de tags ... 290
Trabalho com etiquetas usando a AWS CLI e a API do Amazon EMR no EKS ... 291
Solução de problemas ... 14
Falhas em trabalhos que usam PVC ... 292
Verificação ... 292
Patch ... 293

Aplicação manual de patches ... 296
Falhas de escalabilidade automática vertical ... 298
Erro 403 Forbidden ... 299
Namespace não encontrado ... 299
Erro de credenciais do Docker ... 299
Falhas em operadores do Spark ... 300
Falha na instalação do chart do Helm ... 300
Exceção FileSystem sem suporte ... 300
Cotas e endpoints de serviço ... 302
Service endpoints (Endpoints de serviço) ... 302
Service Quotas ... 303
Versões de liberação ... 305
Versões 6.14.0 ... 306
Versões ... 306
Notas de lançamento ... 307
Recursos ... 309
emr-6.14.0-latest ... 309
emr-6.14.0-20231005 ... 309
Versões 6.13.0 ... 309
Versões ... 310
Notas de lançamento ... 311
Recursos ... 312
emr-6.13.0-latest ... 313
emr-6.13.0-20230814 ... 313
Versões 6.12.0 ... 313
Versões ... 313
Notas de lançamento ... 314
Recursos ... 316
emr-6.12.0-latest ... 316
emr-6.12.0-20230701 ... 316
Versões 6.11.0 ... 316
Versões ... 317
Notas de lançamento ... 317
Recursos ... 319
emr-6.11.0-latest ... 319
emr-6.11.0-20230509 ... 319

Versões 6.10.0 320
emr-6.10.0-latest 322
emr-6.10.0-20230624 323
emr-6.10.0-20230421 323
emr-6.10.0-20230403 323
emr-6.10.0-20230220 323
Versões 6.9.0 324
emr-6.9.0-latest 327
emr-6.9.0-20230624 327
emr-6.9.0-20221108 327
Versões 6.8.0 327
emr-6.8.0-latest 332
emr-6.8.0-20230624 332
emr-6.8.0-20221219 332
emr-6.8.0-20220802 333
Versões 6.7.0 333
emr-6.7.0-latest 335
emr-6.7.0-20230624 335
emr-6.7.0-20221219 335
emr-6.7.0-20220630 335
Versões 6.6.0 336
emr-6.6.0-latest 337
emr-6.6.0-20230624 337
emr-6.6.0-20221219 338
emr-6.6.0-20220411 338
Versões 6.5.0 338
emr-6.5.0-latest 340
emr-6.5.0-20221219 340
emr-6.5.0-20220802 340
emr-6.5.0-20211119 340
Versões 6.4.0 341
emr-6.4.0-latest 342
emr-6.4.0-20221219 342
emr-6.4.0-20210830 342
Versões 6.3.0 343
emr-6.3.0-latest 344

emr-6.3.0-20220802 ... 344
emr-6.3.0-20211008 ... 345
emr-6.3.0-20210802 ... 345
emr-6.3.0-20210429 ... 345
Versões 6.2.0 ... 345
emr-6.2.0-latest ... 347
emr-6.2.0-20220802 ... 347
emr-6.2.0-20211008 ... 347
emr-6.2.0-20210802 ... 348
emr-6.2.0-20210615 ... 348
emr-6.2.0-20210129 ... 348
emr-6.2.0-20201218 ... 348
emr-6.2.0-20201201 ... 349
Versões 5.36.0 ... 349
emr-5.36.0-latest ... 350
emr-5.36.0-20221219 ... 351
emr-5.36.0-20220620 ... 351
emr-5.36.0-20220525 ... 351
Versões 5.35.0 ... 351
emr-5.35.0-latest ... 353
emr-5.35.0-20221219 ... 353
emr-5.35.0-20220802 ... 353
emr-5.35.0-20220307 ... 353
Versões 5.34 ... 354
emr-5.34.0-latest ... 355
emr-5.34.0-20220802 ... 355
emr-5.34.0-20211208 ... 355
Versões 5.33.0 ... 356
emr-5.33.0-latest ... 357
emr-5.33.0-20221219 ... 357
emr-5.33.0-20220802 ... 358
emr-5.33.0-20211008 ... 358
emr-5.33.0-20210802 ... 358
emr-5.33.0-20210615 ... 358
emr-5.33.0-20210323 ... 359
Versões 5.32.0 ... 359

emr-5.32.0-latest 360
emr-5.32.0-20220802 361
emr-5.32.0-20211008 361
emr-5.32.0-20210802 361
emr-5.32.0-20210615 361
emr-5.32.0-20210129 362
emr-5.32.0-20201218 362
emr-5.32.0-20201201 362
Histórico do documento 363

O que é o Amazon EMR no EKS?

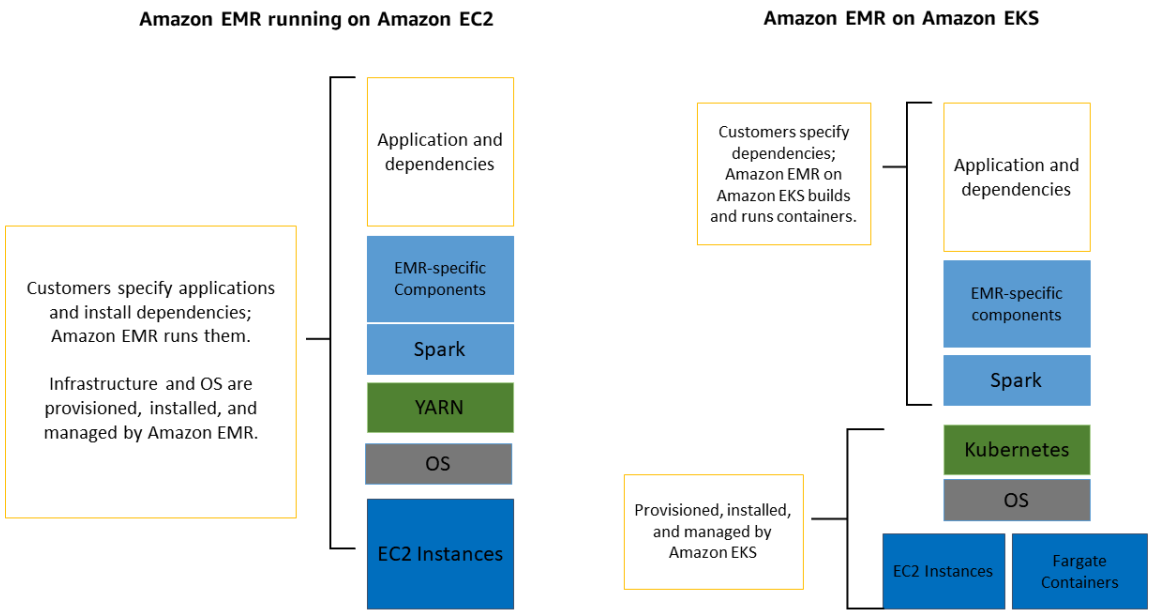
O Amazon EMR no EKS oferece uma opção de implantação para o Amazon EMR que permite executar estruturas de big data de código aberto no Amazon Elastic Kubernetes Service (Amazon EKS). Com essa opção de implantação, você pode se concentrar na execução de workloads de análise enquanto o Amazon EMR no EKS cria, configura e gerencia contêineres para aplicações de código aberto.

Se você já usa o Amazon EMR, agora poderá executar aplicações baseadas no Amazon EMR com outros tipos de aplicações no mesmo cluster do Amazon EKS. Esta opção de implantação também melhora a utilização de recursos e simplifica o gerenciamento da infraestrutura em diversas zonas de disponibilidade. Se você já executa estruturas de big data no Amazon EKS, poderá usar o Amazon EMR para automatizar o provisionamento e o gerenciamento, e executar o Apache Spark mais rapidamente.

O Amazon EMR no EKS possibilita que sua equipe colabore com mais eficiência e processe grandes quantidades de dados com mais facilidade e economia:

- Você pode executar aplicações em um grupo comum de recursos sem a necessidade de provisionar infraestrutura. Você pode usar o Amazon EMR Studio e o AWS SDK ou AWS CLI para desenvolver, enviar e diagnosticar aplicações de análise em execução em clusters do EKS. É possível executar trabalhos programados no Amazon EMR no EKS usando o Apache Airflow autogerenciado ou o Amazon Managed Workflows for Apache Airflow (MWAA).
- As equipes de infraestrutura podem gerenciar centralmente uma plataforma de computação comum para consolidar as workloads do Amazon EMR com outras aplicações baseadas em contêiner. Você pode simplificar o gerenciamento da infraestrutura com ferramentas comuns do Amazon EKS e aproveitar um cluster compartilhado para workloads que precisam de versões diferentes de estruturas de código aberto. Também é possível reduzir a sobrecarga operacional com o gerenciamento automatizado de cluster do Kubernetes e com a aplicação de patches para o sistema operacional. Com o Amazon EC2 e o AWS Fargate, você pode habilitar diversos recursos de computação para atender aos requisitos de performance, de operações ou de finanças.

O diagrama a seguir mostra os dois diferentes modelos de implantação do Amazon EMR.



Tópicos

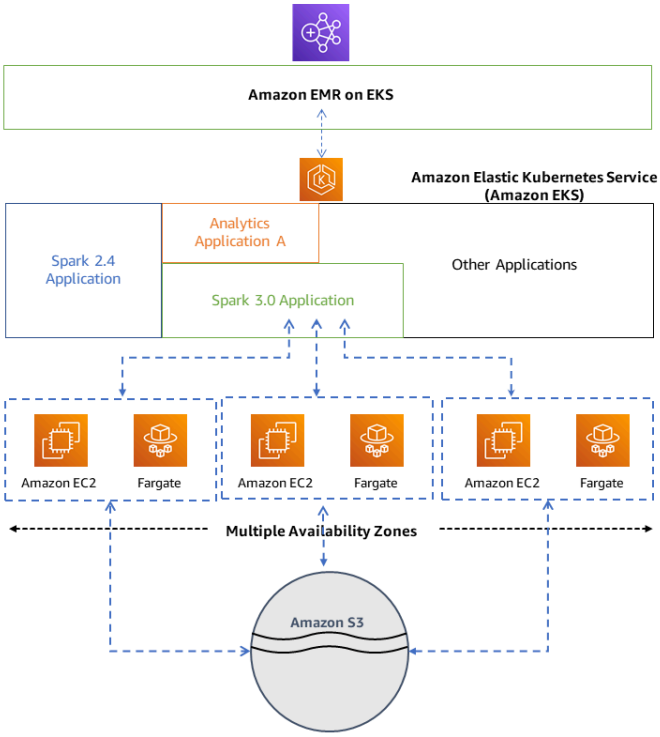
- Arquitetura
- Conceitos
- Como os componentes funcionam em conjunto

Arquitetura

O Amazon EMR no EKS realiza um acoplamento fraco das aplicações à infraestrutura na qual elas são executadas. Cada camada de infraestrutura fornece orquestração para a camada subsequente. Ao enviar um trabalho para o Amazon EMR, a definição do trabalho contém todos os parâmetros específicos da aplicação. O Amazon EMR usa esses parâmetros para instruir o Amazon EKS sobre quais pods e contêineres implantar. Em seguida, o Amazon EKS disponibiliza on-line os recursos de computação do Amazon EC2 e do AWS Fargate necessários para a execução do trabalho.

Com esse acoplamento fraco de serviços, você pode executar simultaneamente múltiplos trabalhos isolados com segurança. Também é possível realizar uma avalição comparativa do mesmo trabalho com diferentes back-ends de computação ou distribuir o trabalho em diversas zonas de disponibilidade para melhorar a disponibilidade.

O diagrama a seguir ilustra como o Amazon EMR no EKS funciona com outros serviços da AWS.



Conceitos

Namespace do Kubernetes

O Amazon EKS usa namespaces do Kubernetes para dividir os recursos de cluster entre diversos usuários e aplicações. Esses namespaces são a base para ambientes multilocatários. Um namespace do Kubernetes pode ter o Amazon EC2 ou o AWS Fargate como o provedor de computação. Essa flexibilidade oferece diferentes opções de performance e de custos para a execução de seus trabalhos.

Cluster virtual

Um cluster virtual corresponde a um namespace do Kubernetes no qual o Amazon EMR está registrado. O Amazon EMR usa clusters virtuais para executar trabalhos e hospedar endpoints. Diversos clusters virtuais podem ser apoiados pelo mesmo cluster físico. No entanto, cada cluster virtual é mapeado para um namespace em um cluster do EKS. Os clusters virtuais não criam quaisquer recursos ativos que contribuam para o seu faturamento ou que requeiram gerenciamento do ciclo de vida de forma externa ao serviço.

Execução de trabalho

Uma execução de trabalho é uma unidade de trabalho, como um JAR do Spark, um script do PySpark ou uma consulta do Spark SQL, que você envia ao Amazon EMR no EKS. Um trabalho pode ter várias execuções de trabalhos. Ao enviar uma execução de trabalho, você inclui as seguintes informações:

- Um cluster virtual no qual o trabalho deve ser executado.
- Um nome do trabalho para a identificação do trabalho.
- O perfil de execução, que é um perfil do IAM com escopo definido que executa o trabalho e permite especificar quais recursos podem ser acessados pelo trabalho.
- O rótulo de versão do Amazon EMR que especifica a versão das aplicações de código aberto a serem usadas.
- Os artefatos a serem usados ao enviar seu trabalho, como os parâmetros spark-submit.

Por padrão, os logs são carregados no servidor de histórico do Spark e podem ser acessados do AWS Management Console. Você também pode enviar logs de eventos, logs de execução e métricas para o Amazon S3 e para o Amazon CloudWatch.

Contêineres do Amazon EMR

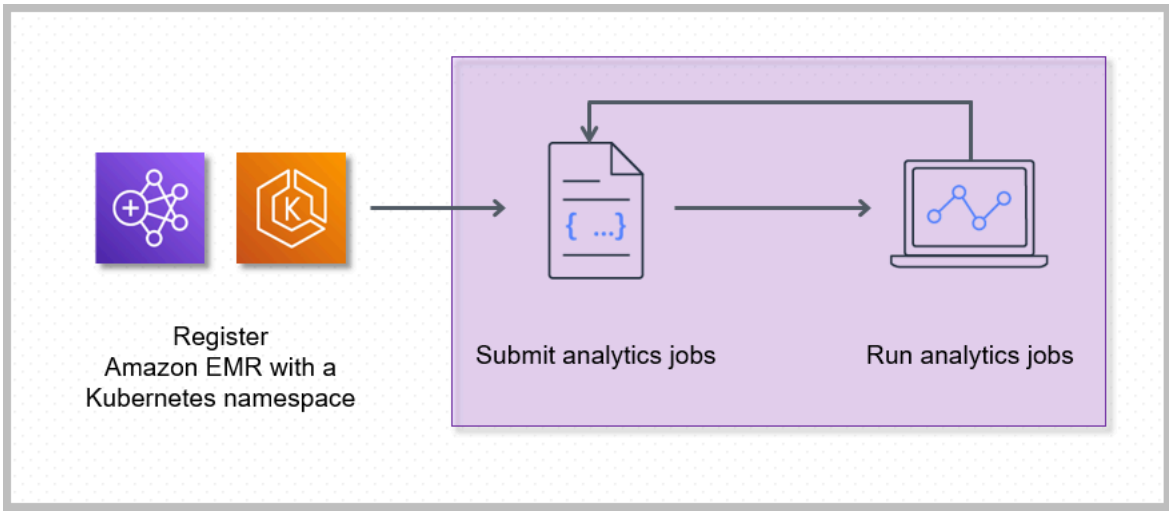
Os contêineres do Amazon EMR correspondem ao nome da API do Amazon EMR no EKS. O prefixo `emr-containers` é usado nos seguintes cenários:

- É o prefixo nos comandos da CLI para o Amazon EMR no EKS. Por exemplo, `aws emr-containers start-job-run`.
- É o prefixo antes das ações de política do IAM para o Amazon EMR no EKS. Por exemplo, `"Action": [ "emr-containers:StartJobRun"]`. Para obter mais informações, consulte Ações de políticas para o Amazon EMR no EKS.
- É o prefixo usado nos endpoints de serviço do Amazon EMR no EKS. Por exemplo, `emr-containers.us-east-1.amazonaws.com`. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Como os componentes funcionam em conjunto

As seguintes etapas e o diagrama abaixo ilustram o fluxo de trabalho do Amazon EMR no EKS:

- Uso de um cluster existente do Amazon EKS ou criação um cluster ao usar o utilitário de linha de comando eksctl ou o console do Amazon EKS.
- Criação de um cluster virtual ao registrar o Amazon EMR com um namespace em um cluster do EKS.
- Envio do seu trabalho para o cluster virtual usando a AWS CLI ou o SDK.



O registro do Amazon EMR com um namespace do Kubernetes no Amazon EKS cria um cluster virtual. O Amazon EMR passará a poder executar workloads de análise nesse namespace. Quando você usa o Amazon EMR no EKS para enviar trabalhos do Spark ao cluster virtual, o Amazon EMR no EKS solicita que o programador do Kubernetes no Amazon EKS programe pods.

Para cada trabalho executado, o Amazon EMR no EKS cria um contêiner com uma imagem base do Amazon Linux 2, o Apache Spark e as dependências associadas. Cada execução de trabalho ocorre em um pod que faz download do contêiner e começa a executá-lo. O pod é encerrado após o término do trabalho. Se a imagem de contêiner tiver sido implantada anteriormente no nó, uma imagem em cache será usada e o download será ignorado. Os contêineres sidecar, como encaminhadores de log ou de métricas, podem ser implantados no pod. Após o término do trabalho, você ainda poderá depurá-lo usando a interface do usuário da aplicação do Spark no console do Amazon EMR.

Conceitos básicos

Este tópico ajuda você a começar a usar o Amazon EMR no EKS ao implantar uma aplicação do Spark em um cluster virtual. Antes de começar, siga as etapas em Configuração do Amazon EMR no EKS. Para obter outros modelos que podem ajudar você a começar a usar, consulte nosso EMR Containers Best Practices Guide no GitHub.

Você precisará das seguintes informações para as etapas de configuração:

- O ID do cluster virtual para o cluster do Amazon EKS e o namespace do Kubernetes registrado no Amazon EMR.

  > **Important**
  >
  > Ao criar um cluster do EKS, certifique-se de usar m5.xlarge como tipo de instância ou qualquer outro tipo de instância com CPU e memória superiores. Usar um tipo de instância com CPU ou memória inferior ao m5.xlarge, pode levar à falha do trabalho devido à insuficiência de recursos disponíveis no cluster.

- O nome do perfil do IAM usado para a execução do trabalho.
- O rótulo de versão da versão do Amazon EMR (por exemplo, `emr-6.4.0-latest`).
- As metas de destino para o registro em log e o monitoramento:
  - O nome do grupo de logs do Amazon CloudWatch e o prefixo do fluxo de logs.
  - A localização do Amazon S3 para armazenar logs de eventos e de contêineres.

> **Important**
>
> Os trabalhos do Amazon EMR no EKS usam o Amazon CloudWatch e o Amazon S3 como as metas de destino para o monitoramento e o registro em log. Você pode monitorar o progresso do trabalho e solucionar falhas ao visualizar os logs de trabalho enviados para esses destinos. Para ativar o registro em log, a política do IAM associada ao perfil do IAM para a execução do trabalho deve ter as permissões necessárias para acessar os recursos das metas. Se a política do IAM não tiver as permissões obrigatórias, você deverá seguir as etapas descritas em Atualização da política de confiança do perfil de execução de trabalho, Configuração de uma execução de trabalho para usar logs do Amazon S3 e Configuração de

uma execução de trabalho para usar o CloudWatch Logs antes de executar este exemplo de trabalho.

Execução de uma aplicação do Spark

Siga as etapas a seguir para executar uma aplicação simples do Spark no Amazon EMR no EKS. O arquivo da aplicação `entryPoint` para a aplicação em Python do Spark está localizado em `s3://`*`REGION`*`.elasticmapreduce/emr-containers/samples/wordcount/scripts/wordcount.py`. A *`REGION`* corresponde à região na qual reside o cluster virtual do Amazon EMR no EKS, como *`us-east-1`*.

1. Atualize a política do IAM para o perfil de execução de trabalho com as permissões necessárias, conforme demonstram as instruções de política a seguir.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadFromLoggingAndInputScriptBuckets",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::*.elasticmapreduce",
                "arn:aws:s3:::*.elasticmapreduce/*",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-OUTPUT",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-OUTPUT/*",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-LOGGING",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-LOGGING/*"
            ]
        },
        {
            "Sid": "WriteToLoggingAndOutputDataBuckets",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:DeleteObject"
            ],
```

```
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-OUTPUT/*",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-LOGGING/*"
            ]
        },
        {
            "Sid": "DescribeAndCreateCloudwatchLogStream",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        },
        {
            "Sid": "WriteToCloudwatchLogs",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:my_log_group_name:log-
stream:my_log_stream_prefix/*"
            ]
        }
    ]
}
```

- A primeira instrução `ReadFromLoggingAndInputScriptBuckets` nesta política concede acesso para `ListBucket` e `GetObjects` aos seguintes buckets do Amazon S3:
  - *REGION*`.elasticmapreduce`: o bucket em que o arquivo da aplicação `entryPoint` está localizado.
  - *DOC-EXAMPLE-BUCKET-OUTPUT*: um bucket que você define para os dados de saída.
  - *DOC-EXAMPLE-BUCKET-LOGGING*: um bucket que você define para os dados de registro em log.

- A segunda instrução `WriteToLoggingAndOutputDataBuckets` nesta política concede ao trabalho as permissões para gravar dados em seus buckets de saída e de registro em log, respectivamente.
- A terceira instrução `DescribeAndCreateCloudwatchLogStream` concede ao trabalho as permissões para descrever e criar com o Amazon CloudWatch Logs.
- A quarta instrução `WriteToCloudwatchLogs` concede as permissões para gravar logs em um grupo de logs do Amazon CloudWatch denominado *`my_log_group_name`* em um fluxo de logs chamado *`my_log_stream_prefix`*.

2. Para executar uma aplicação em Python do Spark, use o comando apresentado a seguir. Substitua todos os valores substituíveis destacados *`em itálico e em vermelho`* por valores apropriados. A *`REGION`* corresponde à região na qual reside o cluster virtual do Amazon EMR no EKS, como *`us-east-1`*.

```
aws emr-containers start-job-run \
--virtual-cluster-id cluster_id \
--name sample-job-name \
--execution-role-arn execution-role-arn \
--release-label emr-6.4.0-latest \
--job-driver '{
  "sparkSubmitJobDriver": {
    "entryPoint": "s3://REGION.elasticmapreduce/emr-containers/samples/wordcount/
scripts/wordcount.py",
    "entryPointArguments": ["s3://DOC-EXAMPLE-BUCKET-OUTPUT/wordcount_output"],
    "sparkSubmitParameters": "--conf spark.executor.instances=2 --
conf spark.executor.memory=2G --conf spark.executor.cores=2 --conf
 spark.driver.cores=1"
  }
}' \
--configuration-overrides '{
  "monitoringConfiguration": {
    "cloudWatchMonitoringConfiguration": {
      "logGroupName": "my_log_group_name",
      "logStreamNamePrefix": "my_log_stream_prefix"
    },
    "s3MonitoringConfiguration": {
       "logUri": "s3://DOC-EXAMPLE-BUCKET-LOGGING"
    }
  }
}'
```

Os dados de saída deste trabalho estarão disponíveis em `s3://DOC-EXAMPLE-BUCKET-OUTPUT/wordcount_output`.

Você também pode criar um arquivo JSON com parâmetros especificados para sua execução de trabalho. Em seguida, execute o comando `start-job-run` com um caminho para o arquivo JSON. Para obter mais informações, consulte Envio de uma execução de trabalho com `StartJobRun`. Para obter mais detalhes sobre como configurar parâmetros de execução de trabalho, consulte Opções para a configuração de uma execução de trabalho.

3. Para executar uma aplicação em SQL do Spark, use o comando apresentado a seguir. Substitua todos os valores destacados *em itálico e em vermelho* por valores apropriados. A *REGION* corresponde à região na qual reside o cluster virtual do Amazon EMR no EKS, como *us-east-1*.

```
aws emr-containers start-job-run \
--virtual-cluster-id cluster_id \
--name sample-job-name \
--execution-role-arn execution-role-arn \
--release-label emr-6.7.0-latest \
--job-driver '{
  "sparkSqlJobDriver": {
    "entryPoint": "s3://query-file.sql",
    "sparkSqlParameters": "--conf spark.executor.instances=2 --
conf spark.executor.memory=2G --conf spark.executor.cores=2 --conf
 spark.driver.cores=1"
  }
}' \
--configuration-overrides '{
  "monitoringConfiguration": {
    "cloudWatchMonitoringConfiguration": {
      "logGroupName": "my_log_group_name",
      "logStreamNamePrefix": "my_log_stream_prefix"
    },
    "s3MonitoringConfiguration": {
       "logUri": "s3://DOC-EXAMPLE-BUCKET-LOGGING"
    }
  }
}'
```

Um arquivo de consulta SQL de exemplo é mostrado abaixo. Você deve ter um armazenamento de arquivos externo, como o S3, no qual os dados das tabelas são armazenados.

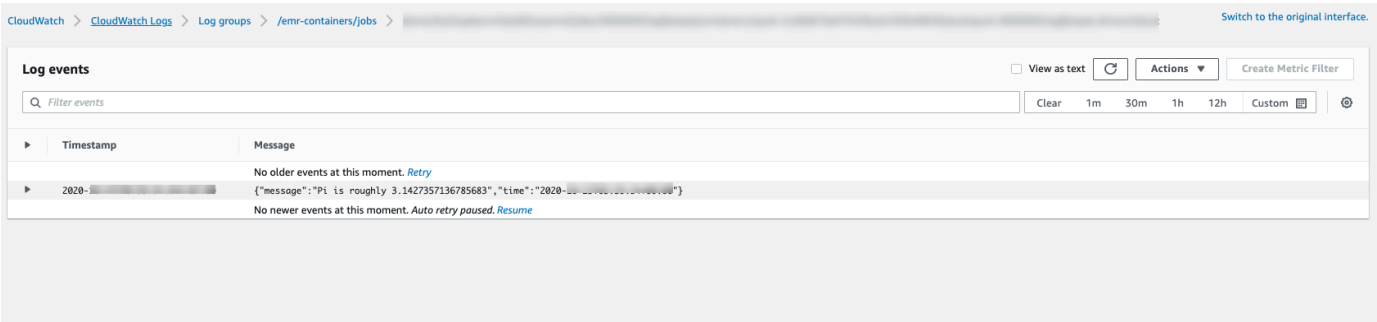
```
CREATE DATABASE demo;
CREATE EXTERNAL TABLE IF NOT EXISTS demo.amazonreview( marketplace string,
 customer_id string, review_id  string, product_id  string, product_parent  string,
 product_title  string, star_rating  integer, helpful_votes  integer, total_votes
  integer, vine  string, verified_purchase  string, review_headline  string,
 review_body  string, review_date  date, year  integer) STORED AS PARQUET LOCATION
 's3://URI to parquet files';
SELECT count(*) FROM demo.amazonreview;
SELECT count(*) FROM demo.amazonreview WHERE star_rating = 3;
```

A saída deste trabalho estará disponível nos logs stdout do driver no S3 ou no CloudWatch, dependendo do `monitoringConfiguration` que estiver configurado.

4. Você também pode criar um arquivo JSON com parâmetros especificados para sua execução de trabalho. Em seguida, execute o comando start-job-run com um caminho para o arquivo JSON. Para obter mais informações, consulte Envio de uma execução de trabalho. Para obter mais detalhes sobre como configurar parâmetros de execução de trabalho, consulte Opções para a configuração de uma execução de trabalho.

   Para monitorar o progresso do trabalho ou depurar falhas, é possível inspecionar os logs carregados no Amazon S3, no CloudWatch Logs ou em ambos. Consulte o caminho do log no Amazon S3 em Configuração de uma execução de trabalho para usar logs do S3 e dos logs do CloudWatch em Configuração de uma execução de trabalho para usar o CloudWatch Logs. Para visualizar os logs no CloudWatch Logs, siga as instruções abaixo.

   - Abra o console do CloudWatch em https://console.aws.amazon.com/cloudwatch/.
   - No painel de navegação, selecione Logs. Em seguida, escolha Grupos de logs.
   - Escolha o grupo de logs do Amazon EMR no EKS e, em seguida, visualize os eventos de log carregados.



> **Important**
>
> Os trabalhos têm uma política de repetição padrão configurada. Para obter informações sobre como modificar ou desabilitar a configuração, consulte Uso de políticas de repetição de trabalho.

Links para os guias de práticas recomendadas do Amazon EMR no EKS no GitHub

Desenvolvemos o Guia de práticas recomendadas do Amazon EMR no EKS com a colaboração da comunidade de código aberto para que fosse possível iterar rapidamente e fornecer recomendações para uma variedade de casos de uso. Recomendamos usar o Guia de práticas recomendadas do Amazon EMR no EKS para as seções. Selecione os links em cada seção para acessar o site do GitHub.

Segurança

> **Note**
>
> Para obter mais informações sobre segurança com o Amazon EMR no EKS, consulte Práticas recomendadas de segurança para o Amazon EMR no EKS.

Práticas recomendadas de criptografia: descreve como usar a criptografia para dados em repouso e em trânsito.

O gerenciamento da segurança da rede descreve como configurar grupos de segurança para pods do Amazon EMR no EKS ao se conectar a fontes de dados hospedadas em Serviços da AWS, como o Amazon RDS e o Amazon Redshift.

Uso do AWS Secrets Manager para armazenar segredos.

Envio de trabalho para o Pyspark

Envio de trabalho para o Pyspark: especifica diferentes tipos de empacotamentos para as aplicações do PySpark usando formatos de empacotamento como zip, egg, wheel e pex.

Armazenamento

Uso de volumes do EBS: descreve como usar o provisionamento estático e dinâmico para trabalhos que precisam de volumes do EBS.

Uso de volumes do Amazon FSx para Lustre: descreve como usar o provisionamento estático e dinâmico para trabalhos que precisam de volumes do Amazon FSx para Lustre.

Uso de volumes de armazenamento de instância: descreve como usar volumes de armazenamento de instância para o processamento de trabalhos.

Integração com o metastore

Uso do Hive Metastore: oferece diferentes maneiras de usar o Hive Metastore.

Uso do AWS Glue: oferece diferentes maneiras de configurar o catálogo do AWS Glue.

Depuração

Uso da depuração do Spark: descreve como alterar o nível de log.

Conexão com a interface do usuário do Spark no pod do driver.

Como usar o servidor de histórico do Spark de hospedagem própria com o Amazon EMR no EKS.

Solução de problemas do Amazon EMR no EKS

Solução de problemas.

Posicionamento de nó

Uso de seletores de nó do Kubernetes para `single-az` e outros casos de uso.

Uso do posicionamento de nó do Fargate.

Desempenho

Uso da alocação dinâmica de recursos (DRA).

Práticas recomendadas do EKS para o plug-in Container Network Interface (CNI) da Amazon VPC, o Cluster Autoscaler e o CoreDNS.

Otimização de custo

Uso de instâncias spot: práticas recomendadas para instâncias spot do Amazon EC2 e como usar o recurso de desativação de nó do Spark.

Usar o AWS Outposts

Execução do Amazon EMR no EKS usando o AWS Outposts.

Personalização de imagens do Docker para o Amazon EMR no EKS

É possível usar imagens do Docker personalizadas com o Amazon EMR no EKS. A personalização da imagem de runtime do Amazon EMR no EKS oferece os seguintes benefícios:

- Empacotamento das dependências e do ambiente de runtime da aplicação em um único contêiner imutável que promove a portabilidade e simplifica o gerenciamento de dependências para cada workload.
- Instalação e configuração de pacotes otimizados para as workloads. Esses pacotes podem não estar amplamente disponíveis na distribuição pública dos runtimes do Amazon EMR.
- Integração do Amazon EMR no EKS aos processos atuais de criação, teste e implantação estabelecidos em sua organização, incluindo desenvolvimento e testes locais.
- Aplicação dos processos de segurança estabelecidos, como a verificação de imagens, que atendem aos requisitos de conformidade e governança da sua organização.

Tópicos

- Como personalizar imagens do Docker
- Como selecionar um URI de imagem base
- Considerações

Como personalizar imagens do Docker

Siga as etapas a seguir para personalizar as imagens do Docker para o Amazon EMR no EKS.

- Pré-requisitos
- Etapa 1: recuperar uma imagem base do Amazon Elastic Container Registry (Amazon ECR)
- Etapa 2: personalizar uma imagem base
- Etapa 3: (opcional, mas recomendada) validar uma imagem personalizada
- Etapa 4: publicar uma imagem personalizada
- Etapa 5: enviar uma workload do Spark no Amazon EMR usando uma imagem personalizada

Confira outras opções que você pode considerar ao personalizar as imagens do Docker:

- Personalização de imagens do Docker para endpoints interativos
- Trabalho com imagens de múltiplas arquiteturas

Pré-requisitos

- Conclua as etapas em Configuração do Amazon EMR no EKS para o Amazon EMR no EKS.
- Instale o Docker em seu ambiente. Para obter mais informações, consulte Get Docker.

Etapa 1: recuperar uma imagem base do Amazon Elastic Container Registry (Amazon ECR)

A imagem base contém o runtime do Amazon EMR e os conectores usados para acessar outros serviços da AWS. Para a versão 6.9.0 e posteriores do Amazon EMR, você pode obter as imagens base na galeria pública do Amazon ECR. Navegue pela galeria para encontrar o link da imagem e extraia-a para seu Workspace local. Por exemplo, para a versão 6.14.0 do Amazon EMR, o comando `docker pull`, apresentado a seguir, fornece a imagem base padrão mais recente. Você pode substituir `emr-6.14.0:latest` por `emr-6.14.0-spark-rapids:latest` para recuperar a imagem que tem o acelerador RAPIDS da Nvidia. Você também pode substituir `emr-6.14.0:latest` por `emr-6.14.0-java11:latest` para recuperar a imagem com o runtime do Java 11.

```
docker pull public.ecr.aws/emr-on-eks/spark/emr-6.14.0:latest
```

Se você desejar recuperar a imagem base para uma versão 6.9.0 ou para versões anteriores do Amazon EMR, ou se preferir recuperá-la de contas de registro do Amazon ECR em cada região, use as seguintes etapas:

1. Escolha um URI de imagem base. O URI de imagem segue este formato, *ECR-registry-account*`.dkr.ecr.`*Region*`.amazonaws.com/spark/`*container-image-tag*, como demonstra o exemplo a seguir.

   ```
   895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.6.0:latest
   ```

   Para escolher uma imagem base em sua região, consulte Como selecionar um URI de imagem base.

2. Faça login no repositório do Amazon ECR no qual a imagem base está armazenada. Substitua *895885662937* e *us-west-2* pela conta de registro do Amazon ECR e pela região da AWS que você selecionou.

```
aws ecr get-login-password --region us-west-2 | docker login --username AWS --
password-stdin 895885662937.dkr.ecr.us-west-2.amazonaws.com
```

3. Extraia a imagem base para seu Workspace local. Substitua *emr-6.6.0:latest* pela etiqueta de imagem do contêiner que você selecionou.

```
docker pull 895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.6.0:latest
```

Etapa 2: personalizar uma imagem base

Siga as etapas apresentadas a seguir para personalizar a imagem base extraída do Amazon ECR.

1. Crie um novo `Dockerfile` em seu Workspace local.
2. Edite o `Dockerfile` que você acabou de criar e adicione o conteúdo a seguir. Este `Dockerfile` usa a imagem de contêiner que você extraiu de `895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.6.0:latest`.

```
FROM 895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.6.0:latest
USER root
### Add customization commands here ####
USER hadoop:hadoop
```

3. Adicione comandos no `Dockerfile` para personalizar a imagem base. Por exemplo, adicione um comando para instalar bibliotecas Python, como demonstra o `Dockerfile` a seguir.

```
FROM 895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.6.0:latest
USER root
RUN pip3 install --upgrade boto3 pandas numpy // For python 3
USER hadoop:hadoop
```

4. No mesmo diretório em que o `Dockerfile` foi criado, execute o comando apresentado a seguir para criar a imagem do Docker. Forneça um nome para a imagem do Docker, por exemplo, *emr6.6_custom*.

```
docker build -t emr6.6_custom .
```

Etapa 3: (opcional, mas recomendada) validar uma imagem personalizada

Recomendamos testar a compatibilidade da sua imagem personalizada antes de publicá-la. Você pode usar a CLI de imagem personalizada do Amazon EMR no EKS para verificar se sua imagem tem as estruturas de arquivos necessárias e as configurações corretas para a execução no Amazon EMR no EKS.

> **Note**
>
> A CLI de imagem personalizada do Amazon EMR no EKS não pode confirmar se sua imagem está isenta de erros. Tenha cuidado ao remover dependências das imagens base.

Siga as etapas apresentadas a seguir para validar a imagem personalizada.

1. Faça download e instale a CLI de imagem personalizada do Amazon EMR no EKS. Para obter mais informações, consulte Amazon EMR on EKS custom image CLI Installation Guide.
2. Execute o comando apresentado a seguir para testar a instalação.

   ```
   emr-on-eks-custom-image --version
   ```

   Confira a seguir um exemplo da saída.

   ```
   Amazon EMR on EKS Custom Image CLI
   Version: x.xx
   ```

3. Execute o comando apresentado a seguir para validar a imagem personalizada.

   ```
   emr-on-eks-custom-image validate-image -i image_name -r release_version [-
   t image_type]
   ```

   - `-i` especifica o URI da imagem local que precisa ser validado. Pode ser o URI da imagem, qualquer nome ou etiqueta que você definiu para a imagem.

- `-r` especifica a versão de liberação exata para a imagem base, por exemplo, `emr-6.6.0-latest`.
- `-t` especifica o tipo de imagem. Se for uma imagem do Spark, insira `spark`. O valor padrão é `spark`. A versão atual da CLI de imagem personalizada do Amazon EMR no EKS oferece suporte somente para imagens de runtime do Spark.

Se você executar o comando com êxito e a imagem personalizada atender a todas as configurações e estruturas de arquivos necessárias, a saída retornada exibirá os resultados de todos os testes, como demonstra o exemplo a seguir.

```
Amazon EMR on EKS Custom Image Test
Version: x.xx
... Checking if docker cli is installed
... Checking Image Manifest
[INFO] Image ID: xxx
[INFO] Created On: 2021-05-17T20:50:07.986662904Z
[INFO] Default User Set to hadoop:hadoop : PASS
[INFO] Working Directory Set to /home/hadoop : PASS
[INFO] Entrypoint Set to /usr/bin/entrypoint.sh : PASS
[INFO] SPARK_HOME is set with value: /usr/lib/spark : PASS
[INFO] JAVA_HOME is set with value: /etc/alternatives/jre : PASS
[INFO] File Structure Test for spark-jars in /usr/lib/spark/jars: PASS
[INFO] File Structure Test for hadoop-files in /usr/lib/hadoop: PASS
[INFO] File Structure Test for hadoop-jars in /usr/lib/hadoop/lib: PASS
[INFO] File Structure Test for bin-files in /usr/bin: PASS
... Start Running Sample Spark Job
[INFO] Sample Spark Job Test with local:///usr/lib/spark/examples/jars/spark-
examples.jar : PASS
----------------------------------------------------------------
Overall Custom Image Validation Succeeded.
----------------------------------------------------------------
```

Se a imagem personalizada não atender às configurações ou estruturas de arquivos necessárias, você obterá mensagens de erro. A saída retornada fornece informações sobre configurações ou estruturas de arquivos incorretas.

Etapa 4: publicar uma imagem personalizada

Publique a nova imagem do Docker no registro do Amazon ECR.

1. Execute o comando apresentado a seguir para criar um repositório do Amazon ECR para o armazenamento da imagem do Docker. Forneça um nome para o repositório, por exemplo, *emr6.6_custom_repo*. Substitua *us-west-2* pela sua região.

```
aws ecr create-repository \
    --repository-name emr6.6_custom_repo \
    --image-scanning-configuration scanOnPush=true \
    --region us-west-2
```

   Para obter mais informações, consulte Criar um repositório no Guia do usuário do Amazon ECR.

2. Execute o comando apresentado a seguir para realizar a autenticação em seu registro padrão.

```
aws ecr get-login-password --region us-west-2 | docker login --username AWS --
password-stdin aws_account_id.dkr.ecr.us-west-2.amazonaws.com
```

   Para obter mais informações, consulte Autenticar-se no registro padrão no Guia do usuário do Amazon ECR.

3. Faça a marcação e publique uma imagem no repositório do Amazon ECR que você criou.

   Marque a imagem.

```
docker tag emr6.6_custom aws_account_id.dkr.ecr.us-
west-2.amazonaws.com/emr6.6_custom_repo
```

   Envie a imagem.

```
docker push aws_account_id.dkr.ecr.us-west-2.amazonaws.com/emr6.6_custom_repo
```

   Para obter mais informações, consulte Enviar uma imagem ao Amazon ECR no Guia do usuário do Amazon ECR.

Etapa 5: enviar uma workload do Spark no Amazon EMR usando uma imagem personalizada

Depois que uma imagem personalizada for criada e publicada, será possível enviar um trabalho do Amazon EMR no EKS usando uma imagem personalizada.

Primeiro, crie um arquivo start-job-run-request.json e especifique o parâmetro `spark.kubernetes.container.image` para fazer referência à imagem personalizada, como demonstra o arquivo JSON de exemplo a seguir.

> **Note**
>
> Você pode usar o esquema `local://` para se referir aos arquivos disponíveis na imagem personalizada, conforme mostrado com o argumento `entryPoint` no trecho de código JSON abaixo. Você também pode usar o esquema `local://` para se referir às dependências da aplicação. Todos os arquivos e as dependências referenciados usando o esquema `local://` já devem estar presentes no caminho especificado na imagem personalizada.

```
{
    "name": "spark-custom-image",
    "virtualClusterId": "virtual-cluster-id",
    "executionRoleArn": "execution-role-arn",
    "releaseLabel": "emr-6.6.0-latest",
    "jobDriver": {
      "sparkSubmitJobDriver": {
        "entryPoint": "local:///usr/lib/spark/examples/jars/spark-examples.jar",
        "entryPointArguments": [
                  "10"
             ],
         "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi --conf
 spark.kubernetes.container.image=123456789012.dkr.ecr.us-west-2.amazonaws.com/
emr6.6_custom_repo"
       }
    }
}
```

Você também pode fazer referência à imagem personalizada com propriedades `applicationConfiguration`, como demonstra o exemplo a seguir.

```
{
    "name": "spark-custom-image",
```

```
    "virtualClusterId": "virtual-cluster-id",
    "executionRoleArn": "execution-role-arn",
    "releaseLabel": "emr-6.6.0-latest",
    "jobDriver": {
      "sparkSubmitJobDriver": {
        "entryPoint": "local:///usr/lib/spark/examples/jars/spark-examples.jar",
        "entryPointArguments": [
                  "10"
             ],
         "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi"
       }
    },
    "configurationOverrides": {
        "applicationConfiguration": [
            {
                "classification": "spark-defaults",
                "properties": {
                    "spark.kubernetes.container.image": "123456789012.dkr.ecr.us-
west-2.amazonaws.com/emr6.6_custom_repo"
                }
            }
        ]
    }
}
```

Em seguida, execute o comando `start-job-run` para enviar o trabalho.

```
aws emr-containers start-job-run --cli-input-json file://./start-job-run-request.json
```

Nos exemplos JSON acima, substitua *emr-6.6.0-latest* pela versão de liberação do Amazon EMR. Recomendamos fortemente usar a versão de liberação `-latest` para garantir que a versão selecionada contenha as atualizações de segurança mais recentes. Para obter mais informações sobre as versões de liberação do Amazon EMR e suas etiquetas de imagem, consulte Como selecionar um URI de imagem base.

> **Note**
>
> É possível usar `spark.kubernetes.driver.container.image` e `spark.kubernetes.executor.container.image` para especificar uma imagem diferente para pods de drivers e de executores.

Personalização de imagens do Docker para endpoints interativos

Você também pode personalizar as imagens do Docker para endpoints interativos com a finalidade de executar imagens base de kernel personalizadas. Isso ajuda a garantir que você tenha as dependências necessárias ao executar workloads interativas do EMR Studio.

1. Siga as etapas 1 a 4 descritas acima para personalizar uma imagem do Docker. Para versões 6.9.0 e posteriores do Amazon EMR, é possível obter o URI da imagem base na galeria pública do Amazon ECR. Para versões anteriores ao Amazon EMR 6.9.0, você pode obter a imagem nas contas de registro do Amazon ECR em cada Região da AWS, e a única diferença é o URI da imagem base em seu Dockerfile. O URI da imagem base segue o formato:

   ```
   ECR-registry-account.dkr.ecr.Region.amazonaws.com/notebook-spark/container-image-tag
   ```

   Você precisa usar `notebook-spark` no URI da imagem base, em vez de `spark`. A imagem base contém o runtime do Spark e os kernels do caderno que são executados com ele. Para obter mais informações sobre como selecionar as regiões e as etiquetas de imagem de contêiner, consulte Como selecionar um URI de imagem base.

   > **Note**
   >
   > No momento, somente substituições de imagens base têm suporte. A introdução de kernels completamente novos de outros tipos, que não as imagens base fornecidas pela AWS, não tem suporte.

2. Crie um endpoint interativo que possa ser usado com a imagem personalizada.

   Primeiro, crie um arquivo JSON chamado `custom-image-managed-endpoint.json` com o conteúdo apresentado a seguir.

   ```
   {
       "name": "endpoint-name",
       "virtualClusterId": "virtual-cluster-id",
       "type": "JUPYTER_ENTERPRISE_GATEWAY",
       "releaseLabel": "emr-6.6.0-latest",
       "executionRoleArn": "execution-role-arn",
       "certificateArn": "certificate-arn",
       "configurationOverrides": {
   ```

```
        "applicationConfiguration": [
            {
                "classification": "jupyter-kernel-overrides",
                "configurations": [
                    {
                        "classification": "python3",
                        "properties": {
                            "container-image": "123456789012.dkr.ecr.us-
west-2.amazonaws.com/custom-notebook-python:latest"
                        }
                    },
                    {
                        "classification": "spark-python-kubernetes",
                        "properties": {
                            "container-image": "123456789012.dkr.ecr.us-
west-2.amazonaws.com/custom-notebook-spark:latest"
                        }
                    }
                ]
            }
        ]
    }
}
```

Em seguida, crie um endpoint interativo usando as configurações especificadas no arquivo JSON, como demonstra o exemplo a seguir.

```
aws emr-containers create-managed-endpoint --cli-input-json custom-image-managed-
endpoint.json
```

Para obter mais informações, consulte Criação de um endpoint interativo para o cluster virtual.

3. Conecte-se ao endpoint interativo usando o EMR Studio. Para obter mais informações, consulte Connecting from Studio.

Trabalho com imagens de múltiplas arquiteturas

O Amazon EMR no EKS oferece suporte a imagens de contêiner de múltiplas arquiteturas para o Amazon Elastic Container Registry (Amazon ECR). Para obter mais informações, consulte Introducing multi-architecture container images for Amazon ECR.

As imagens personalizadas do Amazon EMR no EKS oferecem suporte a instâncias do EC2 baseadas no AWS Graviton e a instâncias do EC2 não baseadas no Graviton. As imagens baseadas no Graviton são armazenadas nos mesmos repositórios de imagens no Amazon ECR que as imagens não baseadas no Graviton.

Por exemplo, para inspecionar a lista de manifestos do Docker para imagens 6.6.0, execute o comando a seguir.

```
docker manifest inspect 895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/
emr-6.6.0:latest
```

Confira a saída a seguir. A arquitetura `arm64` é para instâncias baseadas no Graviton. A arquitetura `amd64` é para instâncias não baseadas no Graviton.

```
{
   "schemaVersion": 2,
   "mediaType": "application/vnd.docker.distribution.manifest.list.v2+json",
   "manifests": [
      {
         "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
         "size": 1805,
         "digest":
 "xxx123:6b971cb47d11011ab3d45fff925e9442914b4977ae0f9fbcdcf5cfa99a7593f0",
         "platform": {
            "architecture": "arm64",
            "os": "linux"
         }
      },
      {
         "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
         "size": 1805,
         "digest":
 "xxx123:6f2375582c9c57fa9838c1d3a626f1b4fc281e287d2963a72dfe0bd81117e52f",
         "platform": {
            "architecture": "amd64",
            "os": "linux"
         }
      }
   ]
}
```

Execute as seguintes etapas para criar imagens de múltiplas arquiteturas:

1. Crie um `Dockerfile` com o conteúdo apresentado a seguir para que você possa extrair a imagem `arm64`.

```
FROM --platform=arm64 895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/
emr-6.6.0:latest
USER root

RUN pip3 install boto3 // install customizations here
USER hadoop:hadoop
```

2. Siga as instruções em Introducing multi-architecture container images for Amazon ECR para desenvolver uma imagem com múltiplas arquiteturas.

> **Note**
>
> Você deve criar imagens `arm64` em instâncias `arm64`. De forma semelhante, você deve criar imagens `amd64` em instâncias `amd64`.

   Também é possível desenvolver imagens com múltiplas arquiteturas sem a necessidade de criar cada tipo de instância específico com o comando `buildx` do Docker. Para obter mais informações, consulte Leverage multi-CPU architecture support.

3. Após criar uma imagem com múltiplas arquiteturas, você poderá enviar um trabalho com o mesmo parâmetro `spark.kubernetes.container.image` e direcioná-lo para a imagem. Em um cluster heterogêneo com instâncias do EC2 baseadas no AWS Graviton e não baseadas no Graviton, a instância determina a imagem de arquitetura correta com base na arquitetura da instância que extrai a imagem.

Como selecionar um URI de imagem base

> **Note**
>
> Para versões 6.9.0 e posteriores do Amazon EMR, é possível recuperar a imagem base da galeria pública do Amazon ECR, portanto, não é necessário criar o URI da imagem base conforme as instruções nesta página. Para encontrar a etiqueta da imagem de contêiner para sua imagem base, consulte a página de notas de versão da versão correspondente do Amazon EMR no EKS.

As imagens base do Docker que você pode selecionar são armazenadas no Amazon Elastic Container Registry (Amazon ECR). O URI da imagem segue este formato: *ECR-registry-account*.dkr.ecr.*Region*.amazonaws.com/spark/*container-image-tag*, como demonstra o seguinte exemplo a seguir.

```
895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.14.0:latest
```

O URI da imagem para endpoints interativos segue este formato: *ECR-registry-account*.dkr.ecr.*Region*.amazonaws.com/notebook-spark/*container-image-tag*, como demonstra o exemplo a seguir. Você precisa usar `notebook-spark` no URI da imagem base, em vez de `spark`.

```
895885662937.dkr.ecr.us-west-2.amazonaws.com/notebook-spark/emr-6.14.0:latest
```

De forma semelhante, para imagens `python3` que não sejam do Spark para endpoints interativos, o URI da imagem é *ECR-registry-account*.dkr.ecr.*Region*.amazonaws.com/notebook-python/*container-image-tag*. O seguinte URI de exemplo está formatado corretamente:

```
895885662937.dkr.ecr.us-west-2.amazonaws.com/notebook-python/emr-6.14.0:latest
```

Para encontrar a etiqueta da imagem de contêiner para sua imagem base, consulte a página de notas de versão da versão correspondente do Amazon EMR no EKS.

Contas de registro do Amazon ECR por região

Para evitar a alta latência de rede, extraia uma imagem base da Região da AWS mais próxima. Selecione a conta de registro do Amazon ECR que corresponde à região da qual você extrairá a imagem com base na tabela a seguir.

Regiões	Contas de registro do Amazon ECR
ap-northeast-1	059004520145
ap-northeast-2	996579266876
ap-south-1	235914868574
ap-southeast-1	671219180197

Regiões	Contas de registro do Amazon ECR	
ap-southeast-2	038297999601	
ca-central-1	351826393999	
eu-central-1	107292555468	
eu-north-1	830386416364	
eu-west-1	483788554619	
eu-west-2	118780647275	
eu-west-3	307523725174	
sa-east-1	052806832358	
us-east-1	755674844232	
us-east-2	711395599931	
us-west-1	608033475327	
us-west-2	895885662937	

Considerações

Ao personalizar imagens do Docker, é possível escolher o runtime exato para seu trabalho em um nível granular. Siga estas práticas recomendadas ao usar esse recurso:

- A segurança é uma responsabilidade compartilhada entre a AWS e você. Você é responsável pela aplicação de patches de segurança nos binários adicionados à imagem. Siga as Práticas recomendadas de segurança para o Amazon EMR no EKS, especialmente o que é colocado em Obtenção das atualizações de segurança mais recentes para as imagens personalizadas e Aplicação do princípio de privilégio mínimo.
- Ao personalizar uma imagem base, você deve alterar o usuário do Docker para `hadoop:hadoop` com a finalidade de que os trabalhos não sejam executados com o usuário raiz.

- O Amazon EMR no EKS monta arquivos sobre as configurações para as imagens, como o `spark-defaults.conf`, no runtime. Para substituir esses arquivos de configuração, recomendamos usar o parâmetro `applicationOverrides` durante o envio do trabalho e não modificar diretamente os arquivos na imagem personalizada.
- O Amazon EMR no EKS monta determinadas pastas no runtime. Quaisquer modificações feitas nessas pastas não estarão disponíveis no contêiner. Se você desejar adicionar uma aplicação ou suas dependências para as imagens personalizadas, recomendamos escolher um diretório que não faça parte dos seguintes caminhos definidos previamente:
  - `/var/log/fluentd`
  - `/var/log/spark/user`
  - `/var/log/spark/apps`
  - `/mnt`
  - `/tmp`
  - `/home/hadoop`
- Você pode fazer upload de sua imagem personalizada para qualquer repositório compatível com Docker, como o Amazon ECR, o Docker Hub ou um repositório empresarial privado. Para obter mais informações sobre como configurar a autenticação de cluster do Amazon EKS com o repositório do Docker selecionado, consulte Pull an Image from a Private Registry.

Execução de trabalhos do Flink com o Amazon EMR no EKS

> **Note**
>
> A funcionalidade do Apache Flink está em versão de pré-visualização para o Amazon EMR no EKS e está sujeita a sofrer alterações. O recurso é fornecido como um serviço de Pré-visualização, conforme definido nos Termos de Serviço da AWS.

As versões 6.13.0 e superiores do Amazon EMR oferecem suporte ao Amazon EMR no EKS com o Apache Flink ou ao operador do Kubernetes para Flink, como um modelo de envio de trabalhos para o Amazon EMR no EKS. Com o Amazon EMR no EKS com o Apache Flink, você pode implantar e gerenciar aplicações do Flink com o runtime da versão do Amazon EMR em seus próprios clusters do Amazon EKS. Depois de implantar o operador do Kubernetes para Flink em seu cluster do Amazon EKS, você poderá enviar aplicações do Flink diretamente com o operador. O operador gerencia o ciclo de vida das aplicações do Flink.

Tópicos

- Operador do Kubernetes para Flink
- Kubernetes nativo
- Monitoramento do operador do Kubernetes para Flink e dos trabalhos do Flink
- Uso da alta disponibilidade (HA) para operadores e aplicações do Flink
- Uso do Autoscaler para aplicações do Flink
- Solução de problemas
- Versões compatíveis para Amazon EMR no EKS com Apache Flink

Operador do Kubernetes para Flink

> **Note**
>
> A funcionalidade do Apache Flink está em versão de pré-visualização para o Amazon EMR no EKS e está sujeita a sofrer alterações. O recurso é fornecido como um serviço de Pré-visualização, conforme definido nos Termos de Serviço da AWS.

As páginas apresentadas a seguir descrevem como configurar e usar o operador do Kubernetes para Flink com a finalidade de executar trabalhos do Flink com o Amazon EMR no EKS.

Tópicos

- Configuração do operador do Kubernetes para Flink para o Amazon EMR no EKS
- Conceitos básicos do operador do Kubernetes para Flink para o Amazon EMR no EKS
- Execução de uma aplicação do Flink
- Segurança
- Desinstalação do operador do Kubernetes para Flink para o Amazon EMR no EKS

Configuração do operador do Kubernetes para Flink para o Amazon EMR no EKS

Conclua as tarefas apresentadas a seguir para se preparar antes de instalar o operador do Kubernetes para Flink no Amazon EKS. Se você já se inscreveu na Amazon Web Services (AWS) e usou o Amazon EKS, está com quase tudo pronto para usar o Amazon EMR no EKS. Conclua as tarefas apresentadas a seguir para se preparar para usar o operador do Flink no Amazon EKS. Se você já completou algum dos pré-requisitos, pode ignorá-los e passar para os próximos.

- Instalar a AWS CLI: se você já instalou a AWS CLI, confirme se tem a versão mais recente.
- Instale o eksctl: o eksctl é uma ferramenta de linha de comando que você usa para se comunicar com o Amazon EKS.
- Instale o Helm: o gerenciador de pacotes Helm para o Kubernetes ajuda a instalar e gerenciar aplicações em seu cluster do Kubernetes.
- Configure um cluster do Amazon EKS: siga as etapas para criar um novo cluster do Kubernetes com nós no Amazon EKS.

- Selecione um rótulo de versão do Amazon EMR (versão 6.13.0 ou superiores): o operador do Kubernetes para Flink é compatível com as versões 6.13.0 e superiores do Amazon EMR.
- Habilite perfis do IAM para contas de serviço (IRSA) no cluster do Amazon EKS.
- Crie um perfil de execução de trabalho.
- Atualize a política de confiança do perfil de execução de trabalho.
- Crie um perfil de execução de operador. Esta etapa é opcional. É possível usar o mesmo perfil para trabalhos e operadores do Flink. Se desejar ter um perfil do IAM diferente para o operador, você poderá criar um perfil separado.
- Atualize a política de confiança do perfil de execução do operador. Você deve adicionar explicitamente uma entrada de política de confiança para os perfis que deseja usar para a conta de serviço do operador do Kubernetes para Flink no Amazon EMR. Você pode seguir este formato de exemplo:

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Principal": {
               "Federated": "arn:aws:iam::<Conta da AWS-id>:oidc-provider/
<OIDC_PROVIDER>"
           },
           "Action": "sts:AssumeRoleWithWebIdentity",
           "Condition": {
               "StringLike": {
                   "<OIDC_PROVIDER>:sub": "system:serviceaccount:<NAMESPACE>:emr-
containers-sa-flink-operator"
               }
           }
       }
   ]
}
```

Conceitos básicos do operador do Kubernetes para Flink para o Amazon EMR no EKS

> **Note**
>
> A funcionalidade do Apache Flink está em versão de pré-visualização para o Amazon EMR no EKS e está sujeita a sofrer alterações. O recurso é fornecido como um serviço de Pré-visualização, conforme definido nos Termos de Serviço da AWS.

Este tópico ajuda você a começar a usar o operador do Kubernetes para Flink no Amazon EKS ao implantar uma implantação do Flink.

Instalação do operador

Use as etapas a seguir para instalar o operador do Kubernetes para Apache Flink.

1. Caso ainda não tenha feito, conclua as etapas em the section called “Configuração”.
2. Instale o *cert-manager* (uma vez por cluster do Amazon EKS) para habilitar a adição do componente webhook.

```
kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/
v1.12.0/cert-manager.yaml
```

3. Instale o chart do Helm.

```
export VERSION=6.14.0 # The Amazon EMR release version
export NAMESPACE=The Kubernetes namespace to deploy the operator

helm install flink-kubernetes-operator-demo \
oci://public.ecr.aws/emr-on-eks/flink-kubernetes-operator \
--version $VERSION \
--namespace $NAMESPACE
```

Exemplos de resultado:

```
NAME: flink-kubernetes-operator-demo
LAST DEPLOYED: Tue May 31 17:38:56 2022
NAMESPACE: $NAMESPACE
STATUS: deployed
```

```
REVISION: 1
TEST SUITE: None
```

4. Aguarde a conclusão da implantação e verifique a instalação do chart.

```
kubectl wait deployment flink-kubernetes-operator-demo --namespace $NAMESPACE --for
 condition=Available=True --timeout=30s
```

5. Você deverá visualizar a mensagem a seguir quando a implantação for concluída.

```
deployment.apps/flink-kubernetes-operator-demo condition met
```

6. Use o comando apresentado a seguir para visualizar o operador implantado.

```
helm list --namespace $NAMESPACE
```

O exemplo a seguir mostra um exemplo de saída, em que a versão da aplicação `x.y.z-amzn-n` corresponderia à versão do operador do Flink para seu Amazon EMR na versão EKS. Para obter mais informações, consulte Versões compatíveis para Amazon EMR no EKS com Apache Flink.

```
NAME                            NAMESPACE    REVISION    UPDATED
               STATUS      CHART                                APP VERSION

flink-kubernetes-operator-demo    $NAMESPACE   1           2023-02-22
 16:43:45.24148 -0500 EST    deployed    flink-kubernetes-operator-emr-6.14.0
 x.y.z-amzn-n
```

Execução de uma aplicação do Flink

O operador do Kubernetes para Flink é compatível com a 6.13.0 ou superiores do Amazon EMR. Use as etapas apresentadas a seguir para executar uma aplicação do Flink com o operador do Kubernetes para Flink nas versões 6.13.0 ou superiores do Amazon EMR no EKS.

> **Note**
>
> Com a pré-visualização pública, o operador do Flink do Amazon EMR no EKS não oferece suporte a trabalhos de sessão do Flink. Você só pode enviar trabalhos

do Flink Application definidos na Definição de Recursos Personalizados (CRD) `flinkdeployments.flink.apache.org`.

> **Note**
>
> Você deve ter um bucket do Amazon S3 criado para armazenar os metadados de alta disponibilidade ao enviar o trabalho do Flink. Se não desejar esse atributo, você poderá desativá-lo. Por padrão, ele é habilitado.

1. Antes de executar uma aplicação do Flink com o operador do Kubernetes para Flink, conclua as etapas em the section called “Configuração” e the section called “Instalação do operador”.
2. Crie um arquivo de definição `FlinkDeployment` `basic-example.yaml` com o seguinte conteúdo de exemplo:

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: basic-example
spec:
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    state.checkpoints.dir: CHECKPOINT S3 STORAGE PATH
    state.savepoints.dir: SAVEPOINT S3 STORAGE PATH
  flinkVersion: v1_17
  executionRoleArn: JOB EXECUTION IAM ROLE ARN
  emrReleaseLabel: "emr-6.13.0-flink-latest"
  jobManager:
    storageDir: HA S3 STORAGE PATH
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  job:
    # if you have your job jar in S3 bucket you can use that path as well
    jarURI: local:///opt/flink/examples/streaming/StateMachineExample.jar
    parallelism: 2
```

```
    upgradeMode: savepoint
    savepointTriggerNonce: 0
  monitoringConfiguration:
    cloudWatchMonitoringConfiguration:
       logGroupName: LOG GROUP NAME
```

3. Envie a implantação do Flink com o comando apresentado a seguir. Isso também criará um objeto `FlinkDeployment` chamado `basic-example`.

```
kubectl create -f example.yaml -n <NAMESPACE>
```

4. Acesse a interface do usuário do Flink.

```
kubectl port-forward deployments/basic-example 8081 -n <NAMESPACE>
```

5. Abra `localhost:8081` para visualizar os trabalhos do Flink localmente.
6. Limpe o trabalho. Lembre-se de limpar os artefatos do S3 que foram criados para este trabalho, como pontos de verificação, HAs, metadados de pontos de salvamento e logs do CloudWatch.

Para obter mais informações sobre o envio de aplicações ao Flink usando o operador do Kubernetes para Flink, consulte Flink Kubernetes operator examples na pasta `apache/flink-kubernetes-operator` no GitHub.

Segurança

RBAC

Para implantar o operador e executar os trabalhos do Flink, é necessário criar dois perfis do Kubernetes: um perfil de operador e um perfil de trabalho. Por padrão, o Amazon EMR cria os dois perfis ao instalar o operador.

Perfil de operador

Usamos o perfil de operador para gerenciar `flinkdeployments` para a criação e o gerenciamento do JobManager para cada trabalho do Flink e para outros recursos, como serviços.

O nome padrão do perfil de operador é `emr-containers-sa-flink-operator` e ele requer as permissões apresentadas a seguir.

```
rules:
```

```
- apiGroups:
  - ""
  resources:
  - pods
  - services
  - events
  - configmaps
  - secrets
  - serviceaccounts
  verbs:
  - '*'
- apiGroups:
  - rbac.authorization.k8s.io
  resources:
  - roles
  - rolebindings
  verbs:
  - '*'
- apiGroups:
  - apps
  resources:
  - deployments
  - deployments/finalizers
  - replicasets
  verbs:
  - '*'
- apiGroups:
  - extensions
  resources:
  - deployments
  - ingresses
  verbs:
  - '*'
- apiGroups:
  - flink.apache.org
  resources:
  - flinkdeployments
  - flinkdeployments/status
  - flinksessionjobs
  - flinksessionjobs/status
  verbs:
  - '*'
- apiGroups:
  - networking.k8s.io
```

```
  resources:
  - ingresses
  verbs:
  - '*'
- apiGroups:
  - coordination.k8s.io
  resources:
  - leases
  verbs:
  - '*'
```

Perfil de trabalho

O JobManager usa o perfil de trabalho para a criação e o gerenciamento do TaskManagers e do ConfigMaps para cada trabalho.

```
rules:
- apiGroups:
  - ""
  resources:
  - pods
  - configmaps
  verbs:
  - '*'
- apiGroups:
  - apps
  resources:
  - deployments
  - deployments/finalizers
  verbs:
  - '*'
```

Desinstalação do operador do Kubernetes para Flink para o Amazon EMR no EKS

> **Note**
>
> A funcionalidade do Apache Flink está em versão de pré-visualização para o Amazon EMR no EKS e está sujeita a sofrer alterações. O recurso é fornecido como um serviço de Pré-visualização, conforme definido nos Termos de Serviço da AWS.

Siga estas etapas para desinstalar o operador do Kubernetes para Flink.

1. Exclua o operador.

```
helm uninstall flink-kubernetes-operator-demo -n <NAMESPACE>
```

2. Exclua os recursos do Kubernetes que o Helm não desinstala.

```
kubectl delete serviceaccounts, roles, rolebindings -l emr-
containers.amazonaws.com/component=flink.operator --namespace <namespace>
kubectl delete crd flinkdeployments.flink.apache.org
 flinksessionjobs.flink.apache.org
```

3. (Opcional) Exclua o cert-manager.

```
kubectl delete -f https://github.com/jetstack/cert-manager/releases/download/
v1.12.0/cert-manager.yaml
```

Kubernetes nativo

> **Note**
>
> A funcionalidade do Apache Flink está em versão de pré-visualização para o Amazon EMR no EKS e está sujeita a sofrer alterações. O recurso é fornecido como um serviço de Pré-visualização, conforme definido nos Termos de Serviço da AWS.

As versões 6.13.0 e superiores do Amazon EMR oferecem suporte ao Kubernetes nativo para Flink como uma ferramenta de linha de comando que você pode usar para enviar e executar aplicações do Flink para um cluster do Amazon EMR no EKS.

Tópicos

- Configuração do Kubernetes nativo para Flink para o Amazon EMR no EKS
- Conceitos básicos do Kubernetes nativo para Flink para o Amazon EMR no EKS
- Requisitos de segurança para a conta de serviço do JobManager do Flink para o Kubernetes nativo

Configuração do Kubernetes nativo para Flink para o Amazon EMR no EKS

> **Note**
>
> A funcionalidade do Apache Flink está em versão de pré-visualização para o Amazon EMR no EKS e está sujeita a sofrer alterações. O recurso é fornecido como um serviço de Pré-visualização, conforme definido nos Termos de Serviço da AWS.

Conclua as tarefas apresentadas a seguir para se preparar antes de executar uma aplicação com a CLI do Flink no Amazon EMR no EKS. Se você já se inscreveu na Amazon Web Services (AWS) e usou o Amazon EKS, está com quase tudo pronto para usar o Amazon EMR no EKS. Se você já completou algum dos pré-requisitos, pode ignorá-los e passar para os próximos.

- Instalar a AWS CLI: se você já instalou a AWS CLI, confirme se tem a versão mais recente.
- Configure um cluster do Amazon EKS: siga as etapas para criar um novo cluster do Kubernetes com nós no Amazon EKS.
- Selecione um URI de imagem base do Amazon EMR (versão 6.13.0 ou superiores): o comando do Kubernetes para Flink é compatível com as versões 6.13.0 e superiores do Amazon EMR.
- Confirme se a conta de serviço do JobManager tem permissões apropriadas para criar e monitorar os pods do TaskManager. Para obter mais informações, consulte Requisitos de segurança para a conta de serviço do JobManager do Flink para o Kubernetes nativo.
- Configure seu perfil de credenciais locais da AWS.
- Crie ou atualize um arquivo kubeconfig para um cluster do Amazon EKS no qual você deseja executar as aplicações do Flink.

Conceitos básicos do Kubernetes nativo para Flink para o Amazon EMR no EKS

Execução de uma aplicação do Flink

As versões 6.13.0 e posteriores do Amazon EMR oferecem suporte ao Kubernetes nativo para Flink para a execução de aplicações do Flink em um cluster do Amazon EKS. Para executar uma aplicação do Flink, siga estas etapas:

1. Antes de executar uma aplicação do Flink com o comando do Kubernetes nativo para Flink, conclua as etapas em the section called “Configuração”.
2. Defina os valores para as variáveis de ambiente a seguir.

```
export FLINK_HOME=
export NAMESPACE=flink
export CLUSTER_ID=flink-application-cluster
export IMAGE=<123456789012.dkr.ecr.sample-Região da AWS-.amazonaws.com/flink/
emr-6.13.0-flink:latest>
export FLINK_SERVICE_ACCOUNT=emr-containers-sa-flink
export FLINK_CLUSTER_ROLE_BINDING=emr-containers-crb-flink
```

3. Crie uma conta de serviço para gerenciar os recursos do Kubernetes.

```
kubectl create serviceaccount $FLINK_SERVICE_ACCOUNT -n $NAMESPACE
kubectl create clusterrolebinding $FLINK_CLUSTER_ROLE_BINDING --clusterrole=edit --
serviceaccount=$NAMESPACE:$FLINK_SERVICE_ACCOUNT
```

4. Execute o comando `run-application` da CLI.

```
$FLINK_HOME/bin/flink run-application \
    --target kubernetes-application \
    -Dkubernetes.namespace=$NAMESPACE \
    -Dkubernetes.cluster-id=$CLUSTER_ID \
    -Dkubernetes.container.image.ref=$IMAGE \
    -Dkubernetes.service-account=$FLINK_SERVICE_ACCOUNT \
    local:///opt/flink/examples/streaming/Iteration.jar
2022-12-29 21:13:06,947 INFO  org.apache.flink.kubernetes.utils.KubernetesUtils
            [] - Kubernetes deployment requires a fixed port. Configuration
 blob.server.port will be set to 6124
2022-12-29 21:13:06,948 INFO  org.apache.flink.kubernetes.utils.KubernetesUtils
            [] - Kubernetes deployment requires a fixed port. Configuration
 taskmanager.rpc.port will be set to 6122
2022-12-29 21:13:07,861 WARN
 org.apache.flink.kubernetes.KubernetesClusterDescriptor      [] - Please note that
 Flink client operations(e.g. cancel, list, stop, savepoint, etc.) won't work from
 outside the Kubernetes cluster since 'kubernetes.rest-service.exposed.type' has
 been set to ClusterIP.
2022-12-29 21:13:07,868 INFO
 org.apache.flink.kubernetes.KubernetesClusterDescriptor      [] - Create flink
 application cluster flink-application-cluster successfully, JobManager Web
 Interface: http://flink-application-cluster-rest.flink:8081
```

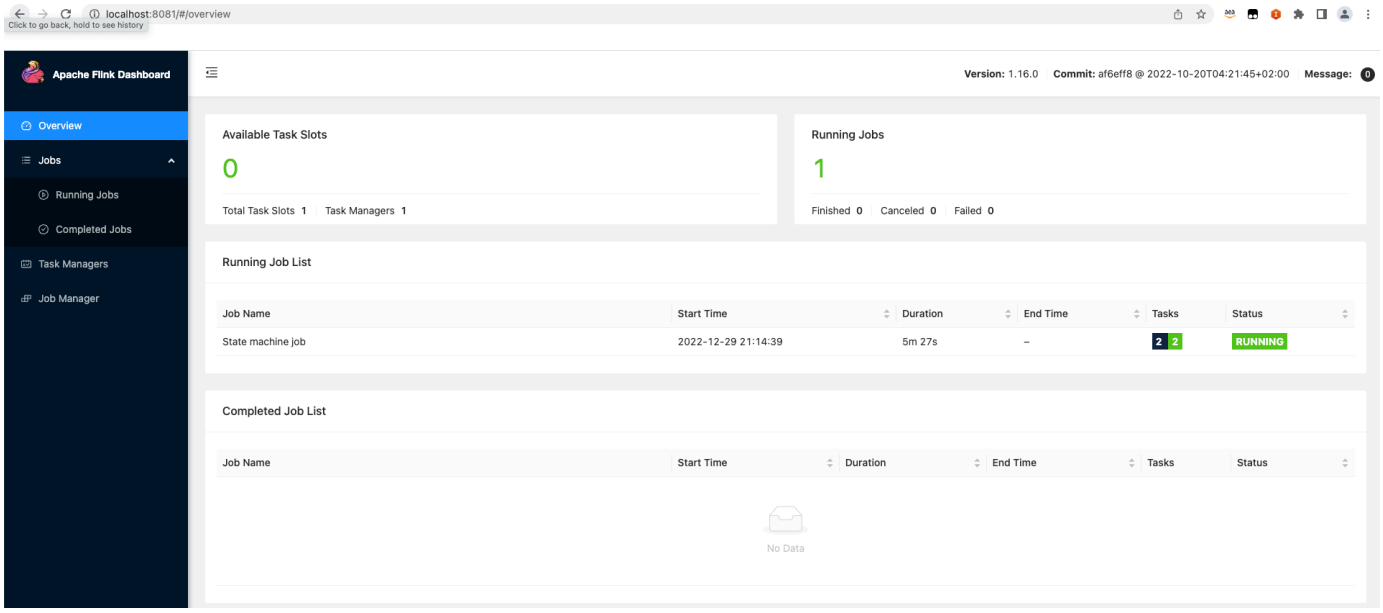
5. Examine os recursos do Kubernetes criados.

```
kubectl get all -n <namespace>
NAME READY STATUS RESTARTS AGE
pod/flink-application-cluster-546687cb47-w2p2z 1/1 Running 0 3m37s
pod/flink-application-cluster-taskmanager-1-1 1/1 Running 0 3m24s

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
service/flink-application-cluster ClusterIP None <none> 6123/TCP,6124/TCP 3m38s
service/flink-application-cluster-rest ClusterIP 10.100.132.158 <none> 8081/TCP
 3m38s

NAME READY UP-TO-DATE AVAILABLE AGE
deployment.apps/flink-application-cluster 1/1 1 1 3m38s

NAME DESIRED CURRENT READY AGE
replicaset.apps/flink-application-cluster-546687cb47 1 1 1 3m38s
```

6. Encaminhe a porta para 8081.

```
kubectl port-forward service/flink-application-cluster-rest 8081 -n <namespace>
Forwarding from 127.0.0.1:8081 -> 8081
```

7. Acesse localmente a interface do usuário do Flink.



8. Exclua a aplicação do Flink.

```
kubectl delete deployment.apps/flink-application-cluster -n <namespace>
deployment.apps "flink-application-cluster" deleted
```

Para obter mais informações sobre o envio de aplicações para o Flink, consulte Native Kubernetes na documentação do Apache Flink.

Requisitos de segurança para a conta de serviço do JobManager do Flink para o Kubernetes nativo

> **Note**
>
> A funcionalidade do Apache Flink está em versão de pré-visualização para o Amazon EMR no EKS e está sujeita a sofrer alterações. O recurso é fornecido como um serviço de Pré-visualização, conforme definido nos Termos de Serviço da AWS.

O pod do JobManager do Flink usa uma conta de serviço do Kubernetes para acessar o servidor da API do Kubernetes com a finalidade de criar e monitorar pods do TaskManager. A conta de serviço do JobManager deve ter as permissões apropriadas para criar e excluir pods do TaskManager, bem como para permitir que o TaskManager observe o ConfigMaps líder para recuperar o endereço do JobManager e do ResourceManager em seu cluster.

As regras apresentadas a seguir se aplicam a esta conta de serviço.

```
rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - "*"
```

```
- apiGroups:
  - ""
  resources:
  - configmaps
  verbs:
  - "*"
```

Monitoramento do operador do Kubernetes para Flink e dos trabalhos do Flink

> **Note**
>
> A funcionalidade do Apache Flink está em versão de pré-visualização para o Amazon EMR no EKS e está sujeita a sofrer alterações. O recurso é fornecido como um serviço de Pré-visualização, conforme definido nos Termos de Serviço da AWS.

Esta seção descreve diversas maneiras para monitorar trabalhos do Flink com o Amazon EMR no EKS.

Tópicos

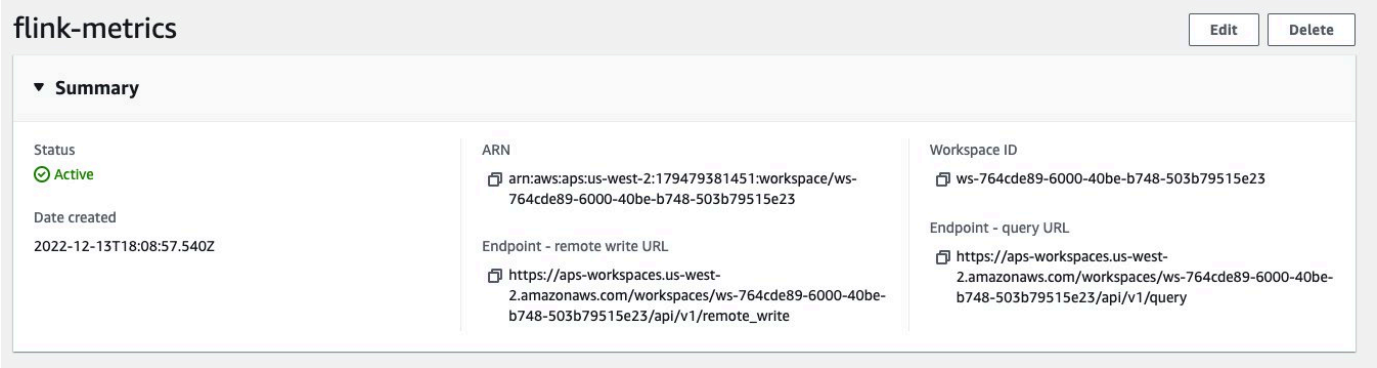
- Uso do Amazon Managed Service for Prometheus para o monitoramento de trabalhos do Flink
- Uso da interface do usuário do Flink para o monitoramento de trabalhos do Flink
- Uso da configuração de monitoramento para o monitoramento do operador do Kubernetes para Flink e dos trabalhos do Flink

Uso do Amazon Managed Service for Prometheus para o monitoramento de trabalhos do Flink

Você pode integrar o Apache Flink ao Amazon Managed Service for Prometheus (portal de gerenciamento). O Amazon Managed Service for Prometheus oferece suporte à ingestão de métricas de servidores do Amazon Managed Service for Prometheus em clusters em execução no Amazon EKS. O Amazon Managed Service for Prometheus funciona em conjunto com um servidor do Prometheus que já está em execução no cluster do Amazon EKS. A execução da integração do Amazon Managed Service for Prometheus com o operador do Flink do Amazon EMR implantará e

configurará automaticamente um servidor do Prometheus para a integração com o Amazon Managed Service for Prometheus.

1. Crie um Workspace do Amazon Managed Service for Prometheus. Este Workspace serve como um endpoint de ingestão. Você precisará do URL de gravação remota posteriormente.



2. Configure perfis do IAM para as contas de serviço.

   Para esse método de integração, use perfis do IAM para as contas de serviço no cluster do Amazon EKS em que o servidor do Prometheus está em execução. Esses perfis também são chamados de perfis de serviço.

   Se você ainda não tiver os perfis, configure perfis de serviço para a ingestão de métricas de clusters do Amazon EKS.

   Antes de continuar, crie um perfil do IAM chamado `amp-iamproxy-ingest-role`.
3. Instale o operador do Flink do Amazon EMR com o Amazon Managed Service for Prometheus.

Agora que você tem um Workspace do Amazon Managed Service for Prometheus, um perfil do IAM dedicado para o Amazon Managed Service for Prometheus e as permissões necessárias, é possível instalar o operador do Flink do Amazon EMR. Transfira o parâmetro `prometheus.install=true` e atualize-o para apontar para sua instância do Amazon Managed Service for Prometheus.

Isso instala automaticamente um relator do Prometheus no operador na porta 9999. Qualquer `FlinkDeployment` futura também expõe uma porta para `metrics` em 9249.

- As métricas do operador do Flink aparecem no Prometheus sob o rótulo `flink_k8soperator_`.
- As métricas Task Manager do Flink aparecem no Prometheus sob o rótulo `flink_taskmanager_`.
- As métricas Job Manager do Flink aparecem no Prometheus sob o rótulo `flink_jobmanager_`.

Use o comando `Helm Install --set` para transferir as substituições para o chart `flink-kubernetes-operator`.

```
helm install flink-kubernetes-operator-demo -n namespace \
 ~/workplace/helm/flink-kubernetes-operator \
 --set prometheus.install=true
```

Uso da interface do usuário do Flink para o monitoramento de trabalhos do Flink

Para monitorar a integridade e a performance de uma aplicação do Flink em execução, use o painel Web do Flink. Este painel fornece informações sobre o status do trabalho, o número de TaskManagers, e as métricas e os logs para o trabalho. Ele também permite visualizar e modificar a configuração do trabalho do Flink e interagir com o cluster do Flink para enviar ou cancelar trabalhos.

Para acessar o painel Web do Flink para uma aplicação do Flink em execução no Kubernetes:

1. Use o comando `kubectl port-forward` para encaminhar uma porta local para a porta na qual o painel Web do Flink está sendo executado nos pods do TaskManager da aplicação do Flink. Por padrão, esta porta é 8081. Substitua *`deployment-name`* pelo nome da implantação da aplicação do Flink acima.

   ```
   kubectl get deployments -n namespace
   ```

   Exemplos de resultado:

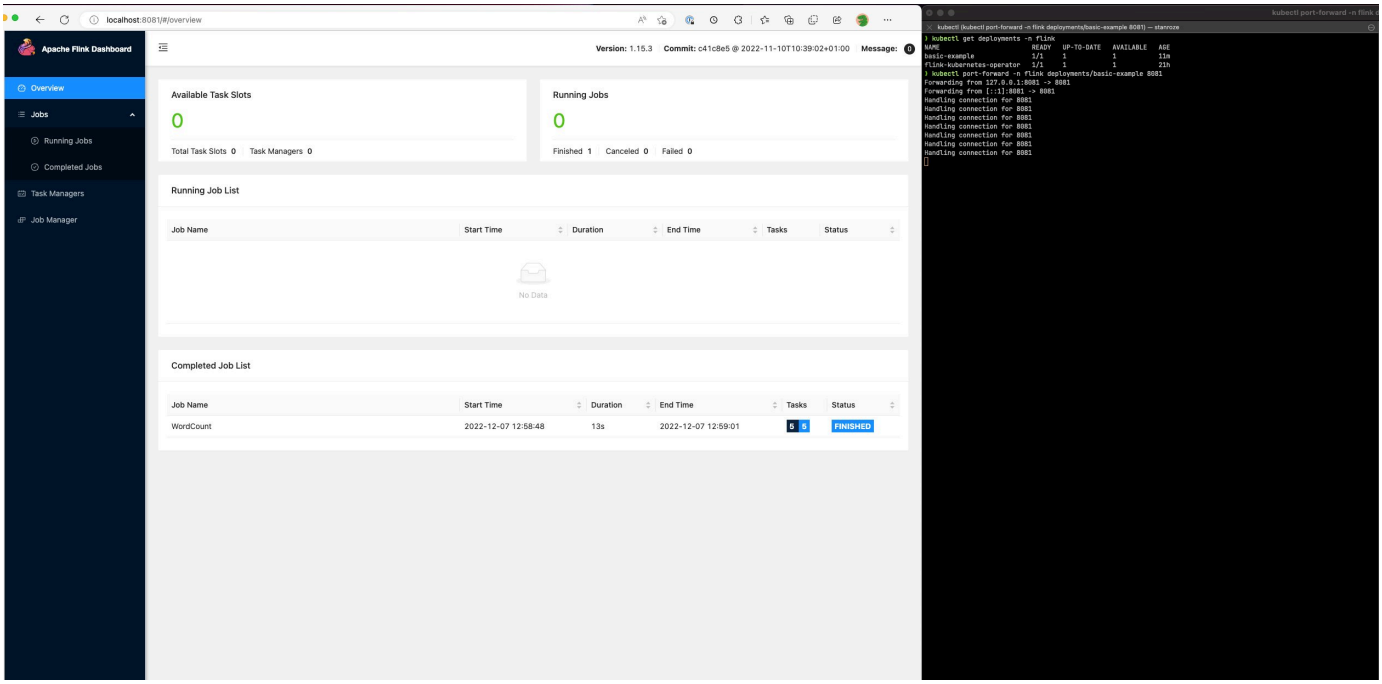
   ```
   kubectl get deployments -n flink-namespace
   NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
   basic-example               1/1       1            1           11m
   flink-kubernetes-operator   1/1       1            1           21h
   ```

   ```
   kubectl port-forward deployments/deployment-name 8081 -n namespace
   ```

2. Se desejar usar uma porta diferente localmente, use o parâmetro *`local-port`*:8081.

   ```
   kubectl port-forward -n flink deployments/basic-example 8080:8081
   ```

3. Em um navegador da Web, vá até `http://localhost:8081` (ou `http://localhost:`*`local-port`*, se você usou uma porta local personalizada) para acessar o painel Web do Flink. Este painel mostra informações sobre a aplicação do Flink em execução, como o status do trabalho, o número de TaskManagers, e as métricas e logs do trabalho.



Uso da configuração de monitoramento para o monitoramento do operador do Kubernetes para Flink e dos trabalhos do Flink

A configuração de monitoramento permite configurar facilmente o arquivamento de logs da aplicação do Flink e dos logs do operador no S3 e no CloudWatch (você pode escolher um deles ou ambos). Fazer isso adiciona um arquivo associado FluentD aos pods do JobManager e do TaskManager e, posteriormente, encaminha os logs desses componentes para os coletores configurados.

> **Note**
>
> Você deve configurar perfis do IAM para a conta de serviço do seu operador do Flink e seu trabalho do Flink (contas de serviço) para poder usar esse recurso, pois ele requer interação com outros Serviços da AWS. Você deve configurar isso usando IRSA em Configuração do operador do Kubernetes para Flink para o Amazon EMR no EKS.

Logs da aplicação do Flink

Você pode definir essa configuração da maneira apresentada a seguir.

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: basic-example
spec:
  image: FLINK IMAGE TAG
  imagePullPolicy: Always
  flinkVersion: v1_17
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
  executionRoleArn: JOB EXECUTION ROLE
  jobManager:
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  job:
    jarURI: local:///opt/flink/examples/streaming/StateMachineExample.jar
  monitoringConfiguration:
    s3MonitoringConfiguration:
      logUri: S3 BUCKET
    cloudWatchMonitoringConfiguration:
      logGroupName: LOG GROUP NAME
      logStreamNamePrefix: LOG GROUP STREAM PREFIX
    sideCarResources:
      limits:
        cpuLimit: 500m
        memoryLimit: 250Mi
    containerLogRotationConfiguration:
        rotationSize: 2Gb
        maxFilesToKeep: 10
```

A seguir, são apresentadas as opções de configuração.

- `s3MonitoringConfiguration`: a chave de configuração para configurar o encaminhamento para o S3.

- `logUri` (obrigatório): o caminho do bucket do S3 em que você deseja armazenar seus logs.
- O caminho no S3 depois que os logs forem carregados será semelhante ao apresentado a seguir.
  - Nenhuma alternância de log habilitada:

    ```
    s3://${logUri}/${POD NAME}/STDOUT or STDERR.gz
    ```

  - A alternância de log está habilitada. Você pode usar um arquivo rotacionado e um arquivo atual (um que não tenha carimbo de data).

    ```
    s3://${logUri}/${POD NAME}/STDOUT or STDERR.gz
    ```

    O formato a seguir é um número incremental.

    ```
    s3://${logUri}/${POD NAME}/stdout_YYYYMMDD_index.gz
    ```

- As permissões do IAM apresentadas a seguir são obrigatórias para usar este encaminhador.

```
{
    "Effect": "Allow",
    "Action": [
        "s3:PutObject"
    ],
    "Resource": [
       "S3_BUCKET_URI/*",
       "S3_BUCKET_URI"
    ]
}
```

- `cloudWatchMonitoringConfiguration`: a chave de configuração para configurar o encaminhamento para o CloudWatch.
  - `logGroupName` (obrigatório): o nome do grupo de logs do CloudWatch para o qual você deseja enviar os logs (há a criação automática do grupo, se ele não existir).
  - `logStreamNamePrefix` (opcional): o nome do fluxo de logs para o qual você deseja enviar os logs. O valor padrão é uma string vazia. O formato é o seguinte:

    ```
    ${logStreamNamePrefix}/${POD NAME}/STDOUT or STDERR
    ```

  - As permissões do IAM apresentadas a seguir são obrigatórias para usar este encaminhador.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogStream",
        "logs:CreateLogGroup",
        "logs:PutLogEvents"
    ],
    "Resource": [
        "arn:aws:logs:REGION:ACCOUNT-ID:log-group:{YOUR_LOG_GROUP_NAME}:*",
        "arn:aws:logs:REGION:ACCOUNT-ID:log-group:{YOUR_LOG_GROUP_NAME}"
    ]
}
```

- `sideCarResources` (opcional): a chave de configuração para definir limites de recursos no contêiner sidecar do Fluent Bit iniciado.
  - `memoryLimit` (opcional): o valor padrão é 512 Mi. Ajuste de acordo com suas necessidades.
  - `cpuLimit` (opcional): esta opção não tem um padrão. Ajuste de acordo com suas necessidades.
- `containerLogRotationConfiguration` (opcional): controla o comportamento de alternância de log do contêiner. Ele é habilitado por padrão.
  - `rotationSize` (obrigatório): especifica o tamanho do arquivo para a alternância de log. O intervalo de valores possíveis é de 2 KB a 2 GB. A parcela numérica da unidade do parâmetro rotationSize é transferida como um número inteiro. Como não há suporte para valores decimais, você pode especificar um tamanho de rotação de 1,5 GB, por exemplo, com o valor 1.500 MB. O padrão é 2 GB.
  - `maxFilesToKeep` (obrigatório): especifica o número máximo de arquivos a serem retidos no contêiner após a alternância ter ocorrido. O valor mínimo é de 1 e o valor máximo é de 50. O padrão é 10.

Logs do operador do Flink

Também podemos habilitar o arquivamento de logs para o operador ao usar as opções apresentadas a seguir no arquivo `values.yaml` da instalação do chart do Helm. Você pode ativar o S3, o CloudWatch ou ambos.

```
monitoringConfiguration:
  s3MonitoringConfiguration:
```

```
    logUri: "S3-BUCKET"
    totalFileSize: "1G"
    uploadTimeout: "1m"
  cloudWatchMonitoringConfiguration:
    logGroupName: "flink-log-group"
    logStreamNamePrefix: "example-job-prefix-test-2"
  sideCarResources:
    limits:
      cpuLimit: 1
      memoryLimit: 800Mi
  memoryBufferLimit: 700M
```

A seguir, estão apresentadas as opções de configuração disponíveis em `monitoringConfiguration`.

- `s3MonitoringConfiguration`: defina esta opção para realizar o arquivamento no S3.
- `logUri` (obrigatório): o caminho do bucket do S3 em que você deseja armazenar seus logs.
- A seguir, estão apresentados os formatos de como os caminhos do bucket do S3 podem parecer depois que os logs são carregados.
  - Nenhuma alternância de log habilitada.

    ```
    s3://${logUri}/${POD NAME}/OPERATOR or WEBHOOK/STDOUT or STDERR.gz
    ```

  - A alternância de log está habilitada. Você pode usar um arquivo rotacionado e um arquivo atual (um que não tenha carimbo de data).

    ```
    s3://${logUri}/${POD NAME}/OPERATOR or WEBHOOK/STDOUT or STDERR.gz
    ```

    O índice de formato a seguir é um número incremental.

    ```
    s3://${logUri}/${POD NAME}/OPERATOR or WEBHOOK/stdout_YYYYMMDD_index.gz
    ```

- `cloudWatchMonitoringConfiguration`: a chave de configuração para configurar o encaminhamento para o CloudWatch.
  - `logGroupName` (obrigatório): o nome do grupo de logs do CloudWatch para o qual você deseja enviar os logs. O grupo é criado automaticamente, se não existir.
  - `logStreamNamePrefix` (opcional): o nome do fluxo de logs para o qual você deseja enviar os logs. O valor padrão é uma string vazia. O formato no CloudWatch é o seguinte:

```
${logStreamNamePrefix}/${POD NAME}/STDOUT or STDERR
```

- `sideCarResources` (opcional): a chave de configuração para definir limites de recursos no contêiner sidecar do Fluent Bit iniciado.
  - `memoryLimit` (opcional): o limite de memória. Ajuste de acordo com suas necessidades. O padrão é de 512Mi.
  - `cpuLimit`: o limite de CPU. Ajuste de acordo com suas necessidades. Nenhum valor padrão.
- `containerLogRotationConfiguration` (opcional): controla o comportamento de alternância de log do contêiner. Ele é habilitado por padrão.
  - `rotationSize` (obrigatório): especifica o tamanho do arquivo para a alternância de log. O intervalo de valores possíveis é de 2 KB a 2 GB. A parcela numérica da unidade do parâmetro rotationSize é transferida como um número inteiro. Como não há suporte para valores decimais, você pode especificar um tamanho de rotação de 1,5 GB, por exemplo, com o valor 1.500 MB. O padrão é 2 GB.
  - `maxFilesToKeep` (obrigatório): especifica o número máximo de arquivos a serem retidos no contêiner após a alternância ter ocorrido. O valor mínimo é de 1 e o valor máximo é de 50. O padrão é 10.

Uso da alta disponibilidade (HA) para operadores e aplicações do Flink

> **Note**
>
> A funcionalidade gerenciada do Flink está em versão de pré-visualização para o Amazon EMR no EKS e está sujeita a sofrer alterações. O recurso é fornecido como um serviço de Pré-visualização, conforme definido nos Termos de Serviço da AWS.

JobManager do Flink

A alta disponibilidade (HA) para as implantações do Flink permite que os trabalhos continuem a fazer progressos mesmo se um erro transitório for encontrado e seu JobManager apresentar falhas. Com a HA habilitada, os trabalhos serão reiniciados, mas a partir do último ponto de verificação com êxito. Sem a HA habilitada, o Kubernetes reiniciará o JobManager, mas seu trabalho começará como um novo trabalho e perderá o progresso. Depois de configurar a HA, é possível dizer ao Kubernetes

para armazenar os metadados de HA em um armazenamento persistente para referência no caso de uma falha transitória no JobManager e, em seguida, retomar os trabalhos a partir do último ponto de verificação com êxito.

A HA é habilitada, por padrão, para os trabalhos do Flink (a contagem de réplicas é definida como dois, o que exigirá que você forneça um local de armazenamento do S3 para que os metadados de HA persistam).

Configurações de HA

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: basic-example
spec:
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
  executionRoleArn: "<JOB EXECUTION ROLE ARN>"
  emrReleaseLabel: "emr-6.13.0-flink-latest"
  jobManager:
    resource:
      memory: "2048m"
      cpu: 1
    replicas: 2
    highAvailabilityEnabled: true
    storageDir: "s3://<S3 PERSISTENT STORAGE DIR>"
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
```

A seguir estão as descrições das configurações de HA apresentadas acima no Job Manager (definidas em .spec.jobManager):

- `highAvailabilityEnabled` (opcional, o padrão é “true”): defina como `false` se você não desejar que a HA seja habilitada e não quiser usar as configurações de HA fornecidas. Você ainda pode manipular o campo “réplicas” para configurar manualmente a HA.
- `replicas` (opcional, o padrão é dois): definir esse número como maior que um cria outros JobManagers em espera e permite uma recuperação mais rápida do trabalho. Se você desabilitar a HA, deverá definir a contagem de réplicas como um ou continuará recebendo erros de validação (somente uma réplica tem suporte se a HA não estiver habilitada).

- `storageDir` (obrigatório): por usar a contagem de réplicas como dois, por padrão, é necessário fornecer um storageDir persistente. No momento, este campo aceita somente caminhos do S3 como local de armazenamento.

Localidade de pod

Se você habilitar a HA, também tentaremos colocar pods na mesma AZ, o que conduz a uma performance melhorada (latência de rede reduzida por ter pods nas mesmas AZs). Este é um processo de melhor esforço, ou seja, se você não tiver recursos suficientes na AZ em que a maioria dos seus pods está programada, os pods restantes ainda serão programados, mas poderão acabar em um nó externo a esta AZ.

Determinação da réplica líder

Se a HA estiver habilitada, as réplicas usarão uma concessão para determinar qual dos JMs é o líder e usarão um Configmap do K8s como datastore para armazenar esses metadados. Se desejar determinar o líder, você pode consultar o conteúdo do Configmap e a chave org.apache.flink.k8s.leader.restserver nos dados para encontrar o pod do K8s com o endereço IP. Você também pode usar os comandos bash apresentados a seguir.

```
ip=$(kubectl get configmap -n <NAMESPACE> <JOB-NAME>-cluster-config-map -o json | jq -
r ".data[\"org.apache.flink.k8s.leader.restserver\"]" | awk -F: '{print $2}' | awk -F
 '/' '{print $3}')
kubectl get pods -n <NAMESPACE>  -o json | jq -r ".items[] | select(.status.podIP ==
 \"$ip\") | .metadata.name"
```

Alta disponibilidade do operador do Flink

Habilitamos a alta disponibilidade para o operador do Flink com a finalidade de que possamos fazer failover para um operador do Flink em espera para minimizar o tempo de inatividade no loop de controle do operador, se ocorrerem falhas. A alta disponibilidade é habilitada, por padrão, e o número padrão de réplicas iniciais para o operador é dois. É possível configurar o campo de réplicas em seu arquivo `values.yaml` para o chart do Helm.

Os seguintes campos são personalizáveis:

- `replicas` (opcional, o padrão é dois): definir esse número como maior que um cria outros operadores em espera e permite uma recuperação mais rápida do trabalho.

- `highAvailabilityEnabled` (opcional, o padrão é “true”): controla se você deseja habilitar a HA. Especificar esse parâmetro como “true” habilita o suporte à implantação multi-AZ e define os parâmetros `flink-conf.yaml` corretos.

Você pode desativar a HA para seu operador ao definir a configuração apresentada a seguir em seu arquivo `values.yaml`.

```
...
imagePullSecrets: []

replicas: 1

# set this to false if you do not want HA
highAvailabilityEnabled: false
...
```

Implantação multi-AZ

Criamos os pods do operador em várias zonas de disponibilidade. Esta é uma restrição leve, e seus pods do operador serão programados na mesma AZ, se você não tiver recursos suficientes em uma AZ diferente.

Determinação da réplica líder

Se a HA estiver habilitada, as réplicas usarão uma concessão para determinar qual dos JMs é o líder e usarão uma concessão do K8s para a eleição do líder. Você pode descrever a concessão e consultar o campo .Spec.Holder Identity para determinar o líder atual.

```
kubectl describe lease <Helm Install Release Name>-<NAMESPACE>-lease -n <NAMESPACE> |
 grep "Holder Identity"
```

Interação entre o Flink e o S3

Configuração de credenciais de acesso

Certifique-se de ter configurado o IRSA com as permissões do IAM apropriadas para acessar o bucket do S3.

Busca por trabalhos em JARs do modo de aplicação do S3

O operador do Flink também oferece suporte à busca de aplicações do S3 em JARs. Você acabou de fornecer o local do S3 para o jarURI em sua especificação FlinkDeployment.

Você também pode usar esse recurso para fazer download de outros artefatos, como os scripts do PyFlink. O script do Python resultante é descartado no caminho `/opt/flink/usrlib/`.

O exemplo a seguir demonstra como usar esse recurso para um trabalho do PyFlink. Observe os campos jarURI e args.

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: python-example
spec:
  image: <YOUR CUSTOM PYFLINK IMAGE>
  emrReleaseLabel: "emr-6.12.0-flink-latest"
  flinkVersion: v1_16
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "1"
  serviceAccount: flink
  jobManager:
    highAvailabilityEnabled: false
    replicas: 1
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  job:
    jarURI: "s3://<S3-BUCKET>/scripts/pyflink.py" # Note, this will trigger the
 artifact download process
    entryClass: "org.apache.flink.client.python.PythonDriver"
    args: ["-pyclientexec", "/usr/local/bin/python3", "-py", "/opt/flink/usrlib/
pyflink.py"]
    parallelism: 1
    upgradeMode: stateless
```

Conectores do S3 para Flink

O Flink vem com dois conectores do S3 (listados abaixo). As seções a seguir debatem sobre o momento de usar cada conector.

Ponto de verificação: conector do S3 para Presto

- Defina o esquema do S3 como s3p://.
- O conector recomendado a ser usado como ponto de verificação para o s3.

Exemplo da especificação FlinkDeployment:

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: basic-example
spec:
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    state.checkpoints.dir: s3p://<UCKET-NAME>/flink-checkpoint/
```

- Defina o esquema do S3 como `s3://` ou (`s3a://`).
- O conector recomendado para a leitura e a gravação de arquivos do S3 (somente um conector do S3 que implementa a interface Flink Filesystem).
- Por padrão, definimos `fs.s3a.aws.credentials.provider` no arquivo `flink-conf.yaml`, que é `com.amazonaws.auth.WebIdentityTokenCredentialsProvider`. Se você substituir completamente o `flink-conf` padrão e estiver interagindo com o S3, certifique-se de usar este provedor.

Exemplo da especificação FlinkDeployment:

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: basic-example
spec:
  job:
    jarURI: local:///opt/flink/examples/streaming/WordCount.jar
    args: [ "--input", "s3a://<INPUT BUCKET>/PATH", "--output", "s3a://<OUTPUT BUCKET>/
PATH" ]
    parallelism: 2
    upgradeMode: stateless
```

Uso do Autoscaler para aplicações do Flink

> **Note**
>
> A funcionalidade do Apache Flink está em versão de pré-visualização para o Amazon EMR no EKS e está sujeita a sofrer alterações. O recurso é fornecido como um serviço de Pré-visualização, conforme definido nos Termos de Serviço da AWS.

O Autoscaler do operador pode ajudar a aliviar a contrapressão ao coletar métricas de trabalhos do Flink e ajustar automaticamente o paralelismo em nível de vértice do trabalho. Confira o seguinte exemplo de como a configuração pode se parecer:

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  ...
spec:
  ...
  flinkVersion: v1_17
  flinkConfiguration:
    kubernetes.operator.job.autoscaler.enabled: "true"
    kubernetes.operator.job.autoscaler.stabilization.interval: 1m
    kubernetes.operator.job.autoscaler.metrics.window: 5m
    kubernetes.operator.job.autoscaler.target.utilization: "0.6"
    kubernetes.operator.job.autoscaler.target.utilization.boundary: "0.2"
    kubernetes.operator.job.autoscaler.restart.time: 2m
    kubernetes.operator.job.autoscaler.catch-up.duration: 5m
    pipeline.max-parallelism: "720"
  ...
```

A seguir, são apresentadas as opções de configuração para o Autoscaler.

- `kubernetes.operator.job.autoscaler.scaling.enabled`: especifica se a ação do Autoscaler deve ser habilitada. O padrão é “false” para oferecer suporte a um modo passivo e de somente métricas, em que o Autoscaler coleta e avalia somente as métricas de performance relacionadas à escalabilidade, mas não aciona nenhuma atualização para os trabalhos. Essa opção pode ser usada para obter confiança no módulo sem qualquer impacto para as aplicações em execução.

- `kubernetes.operator.job.autoscaler.stabilization.interval`: o período de estabilização no qual nenhuma nova escalabilidade será executada. O padrão é de cinco minutos.
- `kubernetes.operator.job.autoscaler.metrics.window`: o tamanho da janela de agregação de métricas de escalabilidade. Quanto mais ampla for a janela, mais harmoniosa e estável ela será, mas o Autoscaler pode demorar mais para reagir a alterações repentinas de carga. O padrão é de dez minutos. Recomendamos que você experimente usando um valor entre 3 e 60 minutos.
- `kubernetes.operator.job.autoscaler.target.utilization`: a utilização desejada para o vértice para fornecer uma performance de trabalho estável e algum buffer para flutuações de carga. O padrão é `0.7`, visando 70% de utilização por carga para os vértices de trabalhos.
- `kubernetes.operator.job.autoscaler.target.utilization.boundary`: o limite da utilização desejada para o vértice, que serve como buffer extra para evitar uma escalabilidade imediata em flutuações de carga. O padrão é `0.4`, o que significa que é permitido um desvio de 40% da utilização desejada antes do acionamento de uma ação de escalabilidade.
- `kubernetes.operator.job.autoscaler.restart.time`: o tempo esperado para reiniciar a aplicação. O padrão é de três minutos.
- `kubernetes.operator.job.autoscaler.catch-up.duration`: o tempo esperado para a recuperação, ou seja, o processamento total de qualquer backlog após a conclusão de uma operação de escalabilidade. O padrão é de cinco minutos. Ao reduzir a duração da recuperação, é necessário que o Autoscaler reserve uma capacidade extra para as ações de escalabilidade.
- `pipeline.max-parallelism`: o paralelismo máximo que o Autoscaler pode usar. O Autoscaler ignora esse limite se ele for maior que o paralelismo máximo configurado na configuração do Flink ou diretamente em cada operador. O padrão é de 200. Observe que o Autoscaler calcula o paralelismo como um divisor do número de paralelismo máximo, portanto, é recomendado escolher configurações de paralelismo máximo com muitos divisores em vez de confiar nos padrões fornecidos pelo Flink. Recomendamos usar múltiplos de 60 para esta configuração, por exemplo, 120, 180, 240, 360, 720 etc.

Para obter uma página de referência de configurações mais detalhada, consulte Autoscaler Configuration.

Solução de problemas

Esta seção descreve como solucionar problemas com o Amazon EMR no EKS. Para obter informações sobre como solucionar problemas gerais com o Amazon EMR, consulte Troubleshoot a cluster no Guia de gerenciamento do Amazon EMR.

- Solução de problemas de trabalhos que usam PersistentVolumeClaims (PVC)
- Solução de problemas de escalabilidade automática vertical do Amazon EMR no EKS
- Solução de problemas do operador do Spark do Amazon EMR no EKS

Solução de problemas do Apache Flink para Amazon EMR no EKS

> **Note**
>
> A funcionalidade do Apache Flink está em versão de pré-visualização para o Amazon EMR no EKS e está sujeita a sofrer alterações. O recurso é fornecido como um serviço de Pré-visualização, conforme definido nos Termos de Serviço da AWS.

Mapeamento de recursos não encontrado ao instalar o chart do Helm

Você pode encontrar a mensagem de erro apresentada a seguir ao instalar o chart do Helm.

```
Error: INSTALLATION FAILED: pulling from host 1234567890.dkr.ecr.us-
west-2.amazonaws.com failed with status code [manifests 6.13.0]: 403 Forbidden Error:
 INSTALLATION FAILED: unable to build kubernetes objects from release manifest:
 [resource mapping not found for name: "flink-operator-serving-cert" namespace: "<the
 namespace to install your operator>" from "": no matches for kind "Certificate" in
 version "cert-manager.io/v1"

ensure CRDs are installed first, resource mapping not found for name: "flink-operator-
selfsigned-issuer" namespace: "<the namespace to install your operator>" " from "": no
 matches for kind "Issuer" in version "cert-manager.io/v1"

ensure CRDs are installed first].
```

Para resolver esse erro, instale o cert-manager para habilitar a adição do componente webhook. Você deve instalar o cert-manager em cada cluster do Amazon EKS que usa.

```
kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/v1.12.0
```

Erro de acesso negado ao AWS service (Serviço da AWS)

Se você ver um erroaccess denied , confirme se o perfil do IAM para `operatorExecutionRoleArn` no arquivo `values.yaml` do chart do Helm tem as permissões corretas. Além disso, certifique-se de que o perfil do IAM em `executionRoleArn` na sua especificação de `FlinkDeployment` tenha as permissões corretas.

`FlinkDeployment` está preso

Se seu `FlinkDeployment` ficar em um estado preso, use as etapas a seguir para forçar a exclusão da implantação:

1. Edite a execução da implantação.

   ```
   kubectl edit -n Flink Namespace flinkdeployments/App Name
   ```

2. Remova este finalizador.

   ```
   finalizers:
     - flinkdeployments.flink.apache.org/finalizer
   ```

3. Exclua a implantação.

   ```
   kubectl delete -n Flink Namespace flinkdeployments/App Name
   ```

Versões compatíveis para Amazon EMR no EKS com Apache Flink

O Apache Flink está disponível com as versões do Amazon EMR no EKS apresentadas a seguir. Para obter informações sobre todas as versões disponíveis, consulte Versões do Amazon EMR no EKS.

Rótulo da versão	Java	Flink	Operador do Flink
emr-6.14.0-flink-latest	11	1.17.1	-

Rótulo da versão	Java	Flink	Operador do Flink
emr-6.14.0-flink-k8s-operator-latest	11	1.17.1	1.6.0
emr-6.13.0-flink-latest	11	1.17.0	-
emr-6.13.0-flink-k8s-operator-latest	11	1.17.0	1.5.0

Execução de trabalhos com o Amazon EMR no EKS

Uma execução de trabalho é uma unidade de trabalho, como um JAR do Spark, um script do PySpark ou uma consulta do Spark SQL, que você envia ao Amazon EMR no EKS. Este tópico fornece uma visão geral do gerenciamento de execuções de trabalhos usando a AWS CLI, da visualização de execuções de trabalhos usando o console do Amazon EMR e da solução de erros comuns de execuções de trabalhos.

> **Note**
>
> Antes de enviar uma execução de trabalho com o Amazon EMR no EKS, você deve concluir as etapas em Configuração do Amazon EMR no EKS.

Tópicos

- Execução de trabalhos do Spark com StartJobRun
- Execução de trabalhos do Spark com o operador do Spark
- Execução de trabalhos do Spark com spark-submit
- Gerenciamento de execuções de trabalhos do Amazon EMR no EKS
- Uso da classificação de envio de trabalho
- Uso de modelos de trabalho
- Uso de modelos de pod
- Uso de políticas de repetição de trabalho
- Uso da alternância de log de eventos do Spark
- Uso da alternância de log do contêiner do Spark
- Uso da escalabilidade automática vertical com trabalhos do Spark no Amazon EMR

Execução de trabalhos do Spark com `StartJobRun`

Tópicos

- Configuração do Amazon EMR no EKS
- Envio de uma execução de trabalho com StartJobRun

Configuração do Amazon EMR no EKS

Conclua as tarefas apresentadas a seguir para se preparar para usar o Amazon EMR no EKS. Se você já se inscreveu na Amazon Web Services (AWS) e usa o Amazon EKS, está com quase tudo pronto para usar o Amazon EMR no EKS. Ignore qualquer tarefa que já tenha concluído.

> **Note**
>
> Você também pode acompanhar o Amazon EMR on EKS Workshop para configurar todos os recursos obrigatórios para executar trabalhos do Spark no Amazon EMR no EKS. O workshop também fornece automação usando modelos do CloudFormation para a criação dos recursos necessários para você começar a usar. Para obter outros modelos e práticas recomendadas, consulte nosso EMR Containers Best Practices Guide no GitHub.

1. Instalar a AWS CLI
2. Instalar o `eksctl`
3. Configuração de um cluster do Amazon EKS
4. Habilitação do acesso ao cluster para o Amazon EMR no EKS
5. Habilitação de perfis do IAM para contas de serviço (IRSA) no cluster do EKS
6. Criação de um perfil de execução de trabalho
7. Atualização da política de confiança do perfil de execução de trabalho
8. Concessão de acesso ao Amazon EMR no EKS para os usuários
9. Registro do cluster do Amazon EKS com o Amazon EMR

Instalar a AWS CLI

Você pode instalar a versão mais recente da AWS CLI para macOS, Linux ou Windows.

> **Important**
>
> Para configurar o Amazon EMR no EKS, você deve ter a versão mais recente da AWS CLI instalada.

Instalar ou atualizar a AWS CLI para macOS

1. Se a AWS CLI estiver instalada, determine qual versão você instalou.

```
aws --version
```

2. Se você tiver uma versão anterior da AWS CLI, use o comando apresentado a seguir para instalar a versão 2 mais recente da AWS CLI. Para obter outras opções de instalação ou para atualizar a versão 2 instalada no momento, consulte Upgrading the AWS CLI version 2 on macOS.

```
curl "https://awscli.amazonaws.com/AWSCLIV2.pkg" -o "AWSCLIV2.pkg"
sudo installer -pkg AWSCLIV2.pkg -target /
```

Se você não conseguir usar a versão 2 da AWS CLI, certifique-se de ter a versão 1 mais recente da AWS CLI instalada usando o comando apresentado a seguir.

```
pip3 install awscli --upgrade --user
```

Instalar ou atualizar a AWS CLI para Linux

1. Se a AWS CLI estiver instalada, determine qual versão você instalou.

```
aws --version
```

2. Se você tiver uma versão anterior da AWS CLI, use o comando apresentado a seguir para instalar a versão 2 mais recente da AWS CLI. Para obter outras opções de instalação ou para atualizar a versão 2 instalada no momento, consulte Upgrading the AWS CLI version 2 on Linux.

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

Se você não conseguir usar a versão 2 da AWS CLI, certifique-se de ter a versão 1 mais recente da AWS CLI instalada usando o comando apresentado a seguir.

```
pip3 install --upgrade --user awscli
```

Instalar ou atualizar a AWS CLI para Windows

1. Se a AWS CLI estiver instalada, determine qual versão você instalou.

   ```
   aws --version
   ```

2. Se você tiver uma versão anterior da AWS CLI, use o comando apresentado a seguir para instalar a versão 2 mais recente da AWS CLI. Para obter outras opções de instalação ou para atualizar a versão 2 instalada no momento, consulte Upgrading the AWS CLI version 2 on Windows.

   1. Faça o download do instalador MSI da AWS CLI para Windows (64 bits) em https://awscli.amazonaws.com/AWSCLIV2.msi.
   2. Execute o instalador MSI obtido por download e siga as instruções na tela. Por padrão, a AWS CLI é instalada em C:\Program Files\Amazon\AWSCLIV2.

   Se você não conseguir usar a versão 2 da AWS CLI, certifique-se de ter a versão 1 mais recente da AWS CLI instalada usando o comando apresentado a seguir.

   ```
   pip3 install --user --upgrade awscli
   ```

Configurar as credenciais da AWS CLI

Tanto o eksctl quanto a AWS CLI requerem que você tenha credenciais da AWS configuradas em seu ambiente. O comando `aws configure` é a maneira mais rápida de configurar sua instalação da AWS CLI para uso geral.

```
$ aws configure
AWS Access Key ID [None]: <AKIAIOSFODNN7EXAMPLE>
AWS Secret Access Key [None]: <wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY>
Default region name [None]: <region-code>
Default output format [None]: <json>
```

Ao digitar esse comando, a AWS CLI solicita quatro informações: chave de acesso, chave de acesso secreta, região da AWS e formato de saída. Essas informações são armazenadas em um perfil (um conjunto de configurações) chamado `default`. Este perfil é usado quando você executa comandos, a menos que algum outro seja especificado. Para obter mais informações, consulte Configurar a AWS CLI no Guia do usuário da AWS Command Line Interface.

Instalar o `eksctl`

Instale a versão mais recente do utilitário de linha de comando `eksctl` no macOS, Linux ou Windows. Para obter mais informações, consulte https://eksctl.io/.

> **Important**
>
> Para configurar o Amazon EMR no EKS, você deve ter a versão 0.34.0 ou posterior do `eksctl`. No entanto, recomendamos fazer o download do eksctl mais recente, pois algumas funcionalidades do Amazon EMR no EKS requerem versões posteriores. Para obter mais informações, consulte Instalar o `eksctl`.

Instalar ou atualizar o eksctl no macOS usando o Homebrew

A maneira mais fácil de começar a usar o Amazon EKS e o macOS é ao instalar o eksctl com o Homebrew. A fórmula do Homebrew para o eksctl instala o eksctl e quaisquer outras dependências obrigatórias para o Amazon EKS, como o kubectl. A fórmula também instala o aws-iam-authenticator, que será necessário se você não tiver a versão 1.16.156 ou posterior da AWS CLI instalada.

1. Se você ainda não tiver instalado o Homebrew no MacOS, instale-o com o seguinte comando.

   ```
   /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/
   master/install.sh)"
   ```

2. Instale a aba Weaveworks do Homebrew.

   ```
   brew tap weaveworks/tap
   ```

3. 1. Instale ou atualize o eksctl.
      - Instale o `eksctl` com o seguinte comando.

        ```
        brew install weaveworks/tap/eksctl
        ```

      - Se o `eksctl` já estiver instalado, execute o comando a seguir para atualizá-lo.

        ```
        brew upgrade eksctl & brew link --overwrite eksctl
        ```

   2. Verifique se a instalação foi bem-sucedida com o comando a seguir. Você deve ter a versão 0.34.0 ou posterior do `eksctl`.

```
eksctl version
```

Para instalar ou atualizar o **`eksctl`** no Linux usando o **`curl`**

1. Faça o download e extraia a versão mais recente do eksctl usando o comando a seguir.

```
curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/
download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
```

2. Mova o binário extraído para /usr/local/bin.

```
sudo mv /tmp/eksctl /usr/local/bin
```

3. Verifique se a instalação foi bem-sucedida com o comando a seguir. Você deve ter a versão 0.34.0 ou posterior do `eksctl`.

```
eksctl version
```

Para instalar ou atualizar o **`eksctl`** no Windows usando o Chocolatey

1. Se você ainda não tiver o Chocolatey instalado no seu sistema Windows, consulte Instalar o Chocolatey.
2. Instale ou atualize o `eksctl`.
   - Instale os binários usando o comando a seguir.

```
chocolatey install -y eksctl
```

   - Se eles já estiverem instalados, execute o seguinte comando para atualizá-los:

```
chocolatey upgrade -y eksctl
```

3. Verifique se a instalação foi bem-sucedida com o comando a seguir. Você deve ter a versão 0.34.0 ou posterior do `eksctl`.

```
eksctl version
```

Configuração de um cluster do Amazon EKS

O Amazon EKS é um serviço gerenciado que facilita a execução do Kubernetes na AWS, eliminando a necessidade de instalar, operar e manter seu próprio ambiente de gerenciamento ou nós do Kubernetes. Siga as etapas descritas abaixo para criar um novo cluster do Kubernetes com nós no Amazon EKS.

Pré-requisitos

> **Important**
>
> Antes de criar um cluster do Amazon EKS, conclua os Requisitos e considerações sobre a VPC e a sub-rede do Amazon EKS no Guia do usuário do Amazon EKS para garantir que seus clusters do Amazon EKS funcionem e sejam escalados conforme o esperado.

Você deve instalar e configurar as seguintes ferramentas e recursos necessários para criar e gerenciar um cluster do Amazon EKS:

- A versão mais recente da AWS CLI.
- A versão 1.20 ou posterior do `kubectl`.
- A versão mais recente de `eksctl`.

Para obter mais informações, consulte Instalar a AWS CLI, Instalar o `kubectl` e Instalar o `eksctl`.

Criação de um cluster do Amazon EKS usando **`eksctl`**

Siga as etapas apresentadas a seguir para criar um cluster do Amazon EKS usando `eksctl`.

> **Important**
>
> Para começar rapidamente, você pode criar um cluster do EKS e os nós com configurações padrão. Entretanto, para o uso em produção, recomendamos personalizar as configurações do cluster e dos nós para atender aos seus requisitos específicos. Para obter uma lista de todas as configurações e opções, execute o comando `eksctl create cluster -h`. Para obter mais informações, consulte Creating and Managing Clusters na documentação do `eksctl`.

1. Crie um par de chaves do Amazon EC2.

   Se você não tiver um par de chaves existente, poderá executar o comando apresentado a seguir para criar um novo par de chaves. Substitua `us-west-2` pela região na qual você deseja criar seu cluster.

   ```
   aws ec2 create-key-pair --region us-west-2 --key-name myKeyPair
   ```

   Salve a saída retornada em um arquivo no seu computador local. Para obter informações, consulte Creating or importing a key pair (Criar ou importar pares de chaves) no Manual do usuário do Amazon EC2 para instâncias do Linux.

   > **Note**
   >
   > Não é necessário um par de chaves para criar um cluster do EKS. Porém, especificar o par de chaves permite que você use o SSH para nós assim que eles forem criados. Você pode especificar um par de chaves somente ao criar o grupo de nós.

2. Crie um cluster do EKS.

   Execute o comando apresentado a seguir para criar um cluster do EKS e nós. Substitua *my-cluster* e *myKeyPair* pelo seu próprio nome de cluster e pelo nome do par de chaves. Substitua *us-west-2* pela região na qual você deseja criar seu cluster. Para obter mais informações sobre as regiões com suporte para o Amazon EKS, consulte Amazon Elastic Kubernetes Service endpoints and quotas.

   ```
   eksctl create cluster \
   --name my-cluster \
   --region us-west-2 \
   --with-oidc \
   --ssh-access \
   --ssh-public-key myKeyPair \
   --instance-types=m5.xlarge \
   --managed
   ```

   > **Important**
   >
   > Ao criar um cluster do EKS, use m5.xlarge como tipo de instância ou qualquer outro tipo de instância com CPU e memória superiores. Usar um tipo de instância com CPU ou

memória inferior, em comparação com m5.xlarge, pode levar à falha do trabalho devido à insuficiência de recursos disponíveis no cluster. Para obter todos os recursos criados, visualize a pilha chamada `eksctl-my-cluster-cluster` no console do AWS Cloud Formation.

O processo de criação de cluster e nó demora alguns minutos. Você visualizará diversas linhas de saída quando o cluster e os nós forem criados. O exemplo a seguir demonstra a última linha de saída.

```
...
[#]  EKS cluster "my-cluster" in "us-west-2" region is ready
```

O `eksctl` criou um arquivo de configuração `kubectl` em `~/.kube` ou adicionou a configuração do novo cluster em um arquivo de configuração existente em `~/.kube`.

3. Visualize e valide os recursos.

   Execute o comando apresentado a seguir para visualizar os nós do cluster.

   ```
   kubectl get nodes -o wide
   ```

   Veja a seguir um exemplo de saída.

   ```
   Amazon EC2 node output

   NAME                                           STATUS   ROLES    AGE    VERSION
             INTERNAL-IP      EXTERNAL-IP     OS-IMAGE         KERNEL-VERSION
            CONTAINER-RUNTIME
   ip-192-168-12-49.us-west-2.compute.internal    Ready    none     6m7s
    v1.18.9-eks-d1db3c   192.168.12.49    52.35.116.65    Amazon Linux 2
    4.14.209-160.335.amzn2.x86_64   docker://19.3.6
   ip-192-168-72-129.us-west-2.compute.internal   Ready    none     6m4s
    v1.18.9-eks-d1db3c   192.168.72.129   44.242.140.21   Amazon Linux 2
    4.14.209-160.335.amzn2.x86_64   docker://19.3.6
   ```

   Para obter mais informações, consulte View nodes.

   Use o comando apresentado a seguir para visualizar as workloads em execução no cluster.

```
kubectl get pods --all-namespaces -o wide
```

Veja a seguir um exemplo de saída.

```
Amazon EC2 output

NAMESPACE     NAME                       READY   STATUS    RESTARTS   AGE     IP
               NODE                                                NOMINATED NODE
 READINESS GATES
kube-system   aws-node-6ctpm             1/1     Running   0          7m43s
 192.168.72.129   ip-192-168-72-129.us-west-2.compute.internal   none
 none
kube-system   aws-node-cbntg             1/1     Running   0          7m46s
 192.168.12.49    ip-192-168-12-49.us-west-2.compute.internal    none
 none
kube-system   coredns-559b5db75d-26t47   1/1     Running   0          14m
 192.168.78.81    ip-192-168-72-129.us-west-2.compute.internal   none
 none
kube-system   coredns-559b5db75d-9rvnk   1/1     Running   0          14m
 192.168.29.248   ip-192-168-12-49.us-west-2.compute.internal    none
 none
kube-system   kube-proxy-l8pbd           1/1     Running   0          7m46s
 192.168.12.49    ip-192-168-12-49.us-west-2.compute.internal    none
 none
kube-system   kube-proxy-zh85h           1/1     Running   0          7m43s
 192.168.72.129   ip-192-168-72-129.us-west-2.compute.internal   none
 none
```

Para obter mais informações sobre o que você visualiza aqui, consulte View workloads.

Criação de um cluster do EKS usando o AWS Management Console e a AWS CLI

Você também pode usar o AWS Management Console e a AWS CLI para criar um cluster do EKS. Siga as etapas em Conceitos básicos do Amazon EKS: AWS Management Console e AWS CLI. Dessa forma, você obtém visibilidade de como cada recurso é criado para o cluster do EKS e de como os recursos interagem entre si.

> **Important**
>
> Ao criar nós para um cluster do EKS, use m5.xlarge como tipo de instância ou qualquer outro tipo de instância com CPU e memória superiores.

Criação de um cluster do EKS com o AWS Fargate

Você também pode criar um cluster do EKS com pods em execução no AWS Fargate.

1. Para criar um cluster do EKS com pods em execução no Fargate, siga as etapas descritas em Conceitos básicos do AWS Fargate usando o Amazon EKS.

   > **Note**
   >
   > O Amazon EMR no EKS precisa do CoreDNS para executar trabalhos no cluster do EKS. Se desejar executar seus pods somente no Fargate, deverá seguir as etapas em Atualizar CoreDNS.

2. Execute o comando apresentado a seguir para visualizar os nós do cluster.

   ```
   kubectl get nodes -o wide
   ```

   Veja a seguir um exemplo de saída do Fargate.

   ```
   Fargate node output

   NAME                                                  STATUS   ROLES    AGE
      VERSION              INTERNAL-IP       EXTERNAL-IP   OS-IMAGE         KERNEL-
   VERSION                   CONTAINER-RUNTIME
   fargate-ip-192-168-141-147.us-west-2.compute.internal   Ready    none
    8m3s    v1.18.8-eks-7c9bda   192.168.141.147   none          Amazon Linux 2
    4.14.209-160.335.amzn2.x86_64   containerd://1.3.2
   fargate-ip-192-168-164-53.us-west-2.compute.internal    Ready    none
    7m30s   v1.18.8-eks-7c9bda   192.168.164.53    none          Amazon Linux 2
    4.14.209-160.335.amzn2.x86_64   containerd://1.3.2
   ```

   Para obter mais informações, consulte View nodes.

3. Execute o comando apresentado a seguir para visualizar as workloads em execução no cluster.

```
kubectl get pods --all-namespaces -o wide
```

Veja a seguir um exemplo de saída do Fargate.

```
Fargate output

NAMESPACE     NAME                       READY   STATUS    RESTARTS   AGE   IP
           NODE                                                  NOMINATED NODE
 READINESS GATES
kube-system   coredns-69dfb8f894-9z95l   1/1     Running   0          18m
 192.168.164.53    fargate-ip-192-168-164-53.us-west-2.compute.internal    none
         none
kube-system   coredns-69dfb8f894-c8v66   1/1     Running   0          18m
 192.168.141.147   fargate-ip-192-168-141-147.us-west-2.compute.internal   none
         none
```

Para obter mais informações, consulte View workloads.

Habilitação do acesso ao cluster para o Amazon EMR no EKS

Você deve permitir o acesso do Amazon EMR no EKS a um namespace específico em seu cluster ao executar as seguintes ações: criar um perfil do Kubernetes, associar o perfil a um usuário do Kubernetes e mapear o usuário do Kubernetes com o perfil vinculado ao serviço `AWSServiceRoleForAmazonEMRContainers`. Essas ações são automatizadas em `eksctl` quando o comando de mapeamento de identidade do IAM é usado com `emr-containers` como nome do serviço. Você pode executar essas operações facilmente ao usar o comando apresentado a seguir.

```
eksctl create iamidentitymapping \
    --cluster my_eks_cluster \
    --namespace kubernetes_namespace \
    --service-name "emr-containers"
```

Substitua *`my_eks_cluster`* pelo nome do cluster do Amazon EKS e *`kubernetes_namespace`* pelo namespace do Kubernetes criado para executar workloads do Amazon EMR.

> **Important**
>
> Você deve fazer o download do eksctl mais recente usando a etapa anterior Instalar o `eksctl` para usar esta funcionalidade.

Etapas manuais para habilitar o acesso ao cluster para o Amazon EMR no EKS

Você também pode usar as etapas manuais apresentadas a seguir para habilitar o acesso ao cluster para o Amazon EMR no EKS.

1. Crie um perfil do Kubernetes em um namespace específico

   Amazon EKS 1.22 - 1.27

   Com o Amazon EKS, versões 1.22 a 1.27, execute o comando a seguir para criar um perfil do Kubernetes em um namespace específico. Esse perfil concede as permissões de RBAC obrigatórias ao Amazon EMR no EKS.

```
namespace=my-namespace
cat - >>EOF | kubectl apply -f - >>namespace "${namespace}"
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: emr-containers
  namespace: ${namespace}
rules:
  - apiGroups: [""]
    resources: ["namespaces"]
    verbs: ["get"]
  - apiGroups: [""]
    resources: ["serviceaccounts", "services", "configmaps", "events", "pods",
 "pods/log"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
  - apiGroups: [""]
    resources: ["secrets"]
    verbs: ["create", "patch", "delete", "watch"]
  - apiGroups: ["apps"]
    resources: ["statefulsets", "deployments"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
```

```
  - apiGroups: ["batch"]
    resources: ["jobs"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
  - apiGroups: ["extensions", "networking.k8s.io"]
    resources: ["ingresses"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
  - apiGroups: ["rbac.authorization.k8s.io"]
    resources: ["roles", "rolebindings"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
  - apiGroups: [""]
    resources: ["persistentvolumeclaims"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
EOF
```

Amazon EKS 1.21 and below

Com o Amazon EKS, versões 1.21 e anteriores, execute o comando a seguir para criar um perfil do Kubernetes em um namespace específico. Esse perfil concede as permissões de RBAC obrigatórias ao Amazon EMR no EKS.

```
namespace=my-namespace
cat - >>EOF | kubectl apply -f - >>namespace "${namespace}"
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: emr-containers
  namespace: ${namespace}
rules:
  - apiGroups: [""]
    resources: ["namespaces"]
    verbs: ["get"]
  - apiGroups: [""]
    resources: ["serviceaccounts", "services", "configmaps", "events", "pods",
 "pods/log"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
  - apiGroups: [""]
    resources: ["secrets"]
```

```
    verbs: ["create", "patch", "delete", "watch"]
  - apiGroups: ["apps"]
    resources: ["statefulsets", "deployments"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
  - apiGroups: ["batch"]
    resources: ["jobs"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
  - apiGroups: ["extensions"]
    resources: ["ingresses"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
  - apiGroups: ["rbac.authorization.k8s.io"]
    resources: ["roles", "rolebindings"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
  - apiGroups: [""]
    resources: ["persistentvolumeclaims"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
EOF
```

2. Crie uma associação de perfil do Kubernetes com escopo definido para o namespace

   Execute o comando apresentado a seguir para criar uma associação de perfil do Kubernetes no namespace especificado. Essa associação de perfil concede as permissões definidas no perfil criado na etapa anterior a um usuário chamado `emr-containers`. Esse usuário identifica perfis vinculados ao serviço para o Amazon EMR no EKS e, dessa forma, permite que o Amazon EMR no EKS execute ações conforme definido pelo perfil criado.

```
namespace=my-namespace

cat - <<EOF | kubectl apply -f - --namespace "${namespace}"
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: emr-containers
  namespace: ${namespace}
subjects:
- kind: User
  name: emr-containers
```

```
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: emr-containers
  apiGroup: rbac.authorization.k8s.io
EOF
```

3. Atualize mapa de configuração **`aws-auth`** do Kubernetes

   Você pode usar uma das opções apresentadas a seguir para mapear o perfil vinculado ao serviço do Amazon EMR no EKS com o usuário `emr-containers` que foi associado ao perfil do Kubernetes na etapa anterior.

   Opção 1: usar o **`eksctl`**

   Execute o comando `eksctl` apresentado a seguir para mapear o perfil vinculado ao serviço do Amazon EMR no EKS com o usuário `emr-containers`.

   ```
   eksctl create iamidentitymapping \
       --cluster my-cluster-name \
       --arn "arn:aws:iam::my-account-id:role/AWSServiceRoleForAmazonEMRContainers" \
       --username emr-containers
   ```

   Opção 2: sem usar o eksctl

   1. Execute o comando apresentado a seguir para abrir o mapa de configuração `aws-auth` no editor de texto.

      ```
      kubectl edit -n kube-system configmap/aws-auth
      ```

      > **Note**
      >
      > Se você receber um erro informando `Error from server (NotFound): configmaps "aws-auth" not found`, consulte as etapas em Add user roles no Guia do usuário do Amazon EKS para aplicar o ConfigMap padrão.

   2. Adicione detalhes do perfil vinculado ao serviço do Amazon EMR no EKS à seção `mapRoles` do `ConfigMap`, em `data`. Adicione essa seção se ela ainda não existir no arquivo. A seção `mapRoles` atualizada em dados deve ser semelhante ao exemplo a seguir.

```
apiVersion: v1
data:
  mapRoles: |
    - rolearn: arn:aws:iam::<your-account-id>:role/
AWSServiceRoleForAmazonEMRContainers
      username: emr-containers
    - ... <other previously existing role entries, if there's any>.
```

3. Salve o arquivo e saia do seu editor de texto.

Habilitação de perfis do IAM para contas de serviço (IRSA) no cluster do EKS

O recurso de perfis do IAM para contas de serviço está disponível nas versões 1.14 e posteriores do Amazon EKS e para clusters do EKS atualizados para as versões 1.13 ou posteriores ou após 3 de setembro de 2019. Para usar esse recurso, é possível atualizar os clusters do EKS existentes para a versão 1.14 ou posterior. Para obter mais informações, consulte Atualizar uma versão do Kubernetes do cluster do Amazon EKS.

Se o seu cluster oferecer suporte para perfis do IAM para contas de serviço, ele terá um URL do emissor OpenID Connect associado a ele. Você pode visualizar esse URL no console do Amazon EKS ou usar o comando da AWS CLI apresentado a seguir para recuperá-lo.

> **Important**
>
> Você deve usar a versão mais recente da AWS CLI para receber a saída adequada deste comando.

```
aws eks describe-cluster --name cluster_name --query "cluster.identity.oidc.issuer" --
output text
```

A saída esperada é semelhante à apresentada a seguir.

```
https://oidc.eks.<region-code>.amazonaws.com/id/EXAMPLED539D4633E53DE1B716D3041E
```

Para usar perfis do IAM para contas de serviço em seu cluster, é necessário criar um provedor de identidades OIDC usando o eksctl ou o AWS Management Console.

Para criar um provedor de identidade OIDC do IAM para o cluster com o **`eksctl`**

Verifique a versão do `eksctl` com o comando a seguir. Este procedimento pressupõe que você instalou o `eksctl` e que a versão do `eksctl` seja 0.32.0 ou posterior.

```
eksctl version
```

Para obter mais informações sobre como instalar ou atualizar o eksctl, consulte Instalar ou atualizar o eksctl.

Crie o provedor de identidade OIDC para o cluster com o seguinte comando. Substitua *`cluster_name`* por seu próprio valor.

```
eksctl utils associate-iam-oidc-provider --cluster cluster_name --approve
```

Para criar um provedor de identidade OIDC do IAM para o cluster com o AWS Management Console

Recupere o URL do emissor OIDC na descrição do console do Amazon EKS do seu cluster ou use o comando da AWS CLI apresentado a seguir.

Use o comando apresentado a seguir para recuperar o URL do emissor OIDC da AWS CLI.

```
aws eks describe-cluster --name <cluster_name> --query "cluster.identity.oidc.issuer"
 --output text
```

Use as etapas apresentadas a seguir para recuperar o URL do emissor OIDC do console do Amazon EKS.

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Provedores de identidade e, em seguida, selecione Criar provedor.
    1. Para Tipo de provedor, escolha Escolher um tipo de provedor e escolha OpenID Connect.
    2. Em Provider URL (URL do provedor), cole o URL emissor OIDC do cluster.
    3. Para Público, digite sts.amazonaws.com e escolha Próxima etapa.
3. Verifique se as informações do provedor estão corretas e escolha Create (Criar) para criar seu provedor de identidade.

Criação de um perfil de execução de trabalho

Para executar workloads no Amazon EMR no EKS, você precisa criar um perfil do IAM. Referimo-nos a esse perfil como perfil de execução de trabalho nesta documentação. Para obter mais informações sobre como criar perfis do IAM, consulte Criação de perfis do IAM no Guia do usuário do IAM.

Você também deve criar uma política do IAM que especifique as permissões para o perfil de execução de trabalho e, em seguida, anexar a política do IAM ao perfil de execução de trabalho.

A política a seguir para o perfil de execução de trabalho permite acesso aos destinos de recursos, ao Amazon S3 e ao CloudWatch. Essas permissões são necessárias para monitorar trabalhos e acessar logs. Para seguir o mesmo processo usando a AWS CLI, você também pode configurar seu perfil usando as etapas na seção Create IAM Role for job execution do Amazon EMR on EKS Workshop.

> **Note**
>
> O acesso deve ter um escopo adequado e não ser concedido a todos os objetos do S3 no perfil de execução de trabalho.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::example-bucket"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams"
            ],
```

```
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        }
    ]
}
```

Para obter mais informações, consulte Uso de perfis de execução de trabalho, Configuração de uma execução de trabalho para usar logs do S3 e Configuração de uma execução de trabalho para usar o CloudWatch Logs.

Atualização da política de confiança do perfil de execução de trabalho

Ao usar perfis do IAM para contas de serviço (IRSA) com a finalidade de executar trabalhos em um namespace do Kubernetes, um administrador deve criar uma relação de confiança entre o perfil de execução de trabalho e a identidade da conta de serviço gerenciado do EMR. A relação de confiança pode ser criada ao atualizar a política de confiança do perfil de execução de trabalho. Observe que a conta de serviço gerenciado do EMR é criada automaticamente no envio do trabalho, com escopo definido para o namespace no qual o trabalho é enviado.

Execute o comando apresentado a seguir para atualizar a política de confiança.

```
aws emr-containers update-role-trust-policy \
      --cluster-name cluster \
      --namespace namespace \
      --role-name iam_role_name_for_job_execution
```

Para obter mais informações, consulte Uso de perfis de execução de trabalho com o Amazon EMR no EKS.

> **Important**
>
> O operador que executa o comando acima deve ter estas permissões: `eks:DescribeCluster`, `iam:GetRole` e `iam:UpdateAssumeRolePolicy`.

Concessão de acesso ao Amazon EMR no EKS para os usuários

Para as ações executadas no Amazon EMR no EKS, uma permissão do IAM correspondente para essa ação é necessária. Você deve criar uma política do IAM que permita executar ações do Amazon EMR no EKS e anexá-la ao perfil ou ao usuário do IAM que você usa.

Este tópico fornece etapas para a criação de uma nova política e para o anexo dela a um usuário. Ele também abrange as permissões básicas necessárias para configurar o ambiente do Amazon EMR no EKS. Recomendamos redefinir as permissões para recursos específicos sempre que possível com base nas suas necessidades de negócios.

Criação de uma nova política do IAM e anexo dela a um usuário no console do IAM

Crie uma nova política do IAM

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação esquerdo do console do IAM, escolha Políticas.
3. Na página Policies (Políticas), escolha Create Policy (Criar política).
4. Na janela Criar política, navegue até a guia Editar JSON. Crie um documento de política com uma ou mais instruções JSON, conforme mostrado nos exemplos que seguem este procedimento. Em seguida, escolha Analisar política.
5. Na tela Review Policy (Revisar política), insira o Policy Name (Nome da política), por exemplo, `AmazonEMROnEKSPolicy`. Insira uma descrição opcional e, em seguida, escolha Criar política.

Anexe a política a um usuário ou a um perfil

1. Faça login no AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Policies (Políticas).
3. Na lista de políticas, marque a caixa de seleção ao lado da política criada na seção anterior. Você pode usar o menu Filtro e a caixa de pesquisa para filtrar a lista de políticas.
4. Selecione Ações da política e escolha Anexar.
5. Escolha o usuário ou o perfil ao qual a política será anexada. Você pode usar o menu Filter (Filtro) e a caixa de pesquisa para filtrar a lista de entidades principais. Após escolher o usuário ou o perfil ao qual a política será anexada, selecione Anexar política.

Permissões para o gerenciamento de clusters virtuais

Para gerenciar clusters virtuais em sua conta da AWS, crie uma política do IAM com as permissões apresentadas a seguir. Essas permissões permitem criar, listar, descrever e excluir clusters virtuais em sua conta da AWS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "emr-containers.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "emr-containers:CreateVirtualCluster",
                "emr-containers:ListVirtualClusters",
                "emr-containers:DescribeVirtualCluster",
                "emr-containers:DeleteVirtualCluster"
            ],
            "Resource": "*"
        }
    ]
}
```

Quando a operação `CreateVirtualCluster` é invocada pela primeira vez usando uma conta da AWS, você também precisa de permissões `CreateServiceLinkedRole` para criar o perfil vinculado ao serviço para o Amazon EMR no EKS. Para obter mais informações, consulte Uso de perfis vinculados ao serviço para o Amazon EMR no EKS.

Permissões para o envio de trabalhos

Para enviar trabalhos nos clusters virtuais em sua conta da AWS, crie uma política do IAM com as permissões apresentadas a seguir. Essas permissões permitem iniciar, listar, descrever e cancelar execuções de trabalhos para todos os clusters virtuais em sua conta. Você deve considerar adicionar permissões para listar ou descrever clusters virtuais, o que permite verificar o estado do cluster virtual antes de enviar trabalhos.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "emr-containers:StartJobRun",
                "emr-containers:ListJobRuns",
                "emr-containers:DescribeJobRun",
                "emr-containers:CancelJobRun"
            ],
            "Resource": "*"
        }
    ]
}
```

Permissões para a depuração e o monitoramento

Para obter acesso aos logs enviados ao Amazon S3 e ao CloudWatch ou para visualizar os logs de eventos da aplicação no console do Amazon EMR, crie uma política do IAM com as permissões apresentadas a seguir. Recomendamos redefinir as permissões para recursos específicos sempre que possível com base nas suas necessidades de negócios.

> **Important**
>
> Se você não criou um bucket do Amazon S3, será necessário adicionar a permissão `s3:CreateBucket` à instrução de política. Se você não criou um grupo de logs, será necessário adicionar `logs:CreateLogGroup` à instrução de política.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "emr-containers:DescribeJobRun",
                "elasticmapreduce:CreatePersistentAppUI",
                "elasticmapreduce:DescribePersistentAppUI",
                "elasticmapreduce:GetPersistentAppUIPresignedURL"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:Get*",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams"
            ],
            "Resource": "*"
        }
    ]
}
```

Para obter mais informações sobre como configurar uma execução de trabalho para enviar logs ao Amazon S3 e ao CloudWatch, consulte Configuração de uma execução de trabalho para usar logs do S3 e Configuração de uma execução de trabalho para usar o CloudWatch Logs.

Registro do cluster do Amazon EKS com o Amazon EMR

Registrar o cluster é a etapa final obrigatória para configurar o Amazon EMR no EKS para executar workloads.

Use o comando apresentado a seguir para criar um cluster virtual com um nome de sua escolha para o cluster e para o namespace do Amazon EKS configurados nas etapas anteriores.

> **Note**
>
> Cada cluster virtual deve ter um nome exclusivo em todos os clusters do EKS. Se dois clusters virtuais tiverem o mesmo nome, o processo de implantação falhará mesmo que os dois clusters virtuais pertençam a clusters do EKS diferentes.

```
aws emr-containers create-virtual-cluster \
--name virtual_cluster_name \
--container-provider '{
    "id": "cluster_name",
    "type": "EKS",
    "info": {
        "eksInfo": {
            "namespace": "namespace_name"
        }
    }
}'
```

Como alternativa, você pode criar um arquivo JSON que inclua os parâmetros obrigatórios para o cluster virtual e, em seguida, executar o comando `create-virtual-cluster` com o caminho para o arquivo JSON. Para obter mais informações, consulte Gerenciamento de clusters virtuais.

> **Note**
>
> Para validar a criação com êxito de um cluster virtual, visualize o status dos clusters virtuais ao usar a operação `list-virtual-clusters` ou acessar a página Clusters virtuais no console do Amazon EMR.

Envio de uma execução de trabalho com **`StartJobRun`**

Enviar uma execução de trabalho com um arquivo JSON com parâmetros especificados

1. Crie um arquivo `start-job-run-request.json` e especifique os parâmetros obrigatórios para a execução de trabalho, como demonstrado pelo arquivo JSON de exemplo a seguir. Para obter mais informações sobre os parâmetros, consulte Opções para a configuração de uma execução de trabalho.

```
{
  "name": "myjob",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "emr-6.2.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "entryPoint_location",
      "entryPointArguments": ["argument1", "argument2", ...],
       "sparkSubmitParameters": "--class <main_class> --conf
 spark.executor.instances=2 --conf spark.executor.memory=2G --conf
 spark.executor.cores=2 --conf spark.driver.cores=1"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.driver.memory":"2G"
         }
      }
    ],
    "monitoringConfiguration": {
      "persistentAppUI": "ENABLED",
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "my_log_group",
        "logStreamNamePrefix": "log_stream_prefix"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://my_s3_log_location"
      }
    }
  }
}
```

2. Use o comando `start-job-run` com um caminho para o arquivo `start-job-run-request.json` armazenado localmente.

```
aws emr-containers start-job-run \
--cli-input-json file://./start-job-run-request.json
```

Iniciar uma execução de trabalho usando o comando **`start-job-run`**

1. Forneça todos os parâmetros especificados no comando `StartJobRun`, como demonstrado pelo exemplo a seguir.

```
aws emr-containers start-job-run \
--virtual-cluster-id 123456 \
--name myjob \
--execution-role-arn execution-role-arn \
--release-label emr-6.2.0-latest \
--job-driver '{"sparkSubmitJobDriver": {"entryPoint": "entryPoint_location",
 "entryPointArguments": ["argument1", "argument2", ...], "sparkSubmitParameters":
 "--class <main_class> --conf spark.executor.instances=2 --conf
 spark.executor.memory=2G --conf spark.executor.cores=2 --conf
 spark.driver.cores=1"}}' \
--configuration-overrides '{"applicationConfiguration": [{"classification":
 "spark-defaults", "properties": {"spark.driver.memory": "2G"}}],
 "monitoringConfiguration": {"cloudWatchMonitoringConfiguration":
 {"logGroupName": "log_group_name", "logStreamNamePrefix": "log_stream_prefix"},
 "persistentAppUI":"ENABLED",  "s3MonitoringConfiguration": {"logUri":
 "s3://my_s3_log_location" }}}'
```

2. Para o Spark SQL, forneça todos os parâmetros especificados no comando `StartJobRun`, como demonstrado pelo exemplo a seguir.

```
aws emr-containers start-job-run \
--virtual-cluster-id 123456 \
--name myjob \
--execution-role-arn execution-role-arn \
--release-label emr-6.7.0-latest \
--job-driver '{"sparkSqlJobDriver": {"entryPoint": "entryPoint_location",
 "sparkSqlParameters": "--conf spark.executor.instances=2 --conf
 spark.executor.memory=2G --conf spark.executor.cores=2 --conf
 spark.driver.cores=1"}}' \
--configuration-overrides '{"applicationConfiguration": [{"classification":
 "spark-defaults", "properties": {"spark.driver.memory": "2G"}}],
 "monitoringConfiguration": {"cloudWatchMonitoringConfiguration":
 {"logGroupName": "log_group_name", "logStreamNamePrefix": "log_stream_prefix"},
 "persistentAppUI":"ENABLED",  "s3MonitoringConfiguration": {"logUri":
 "s3://my_s3_log_location" }}}'
```

Execução de trabalhos do Spark com o operador do Spark

As versões 6.10.0 e superiores do Amazon EMR oferecem suporte ao operador do Kubernetes para Apache Spark, ou ao operador do Spark, como um modelo de envio de trabalho para o Amazon EMR no EKS. Com o operador do Spark, você pode implantar e gerenciar aplicações do Spark com o runtime da versão do Amazon EMR em seus próprios clusters do Amazon EKS. Após implantar o operador do Spark no cluster do Amazon EKS, você poderá enviar aplicações do Spark diretamente ao operador. O operador gerencia o ciclo de vida das aplicações do Spark.

> **Note**
>
> O Amazon EMR calcula os preços do Amazon EKS com base nos recursos de vCPU e de memória usados no pod do operador desde o momento em que o download da imagem da aplicação do Amazon EMR tem início até o encerramento do pod do Amazon EKS, arredondado para o segundo mais próximo.

Tópicos

- Configuração do operador do Spark para o Amazon EMR no EKS
- Conceitos básicos do operador do Spark para o Amazon EMR no EKS
- Usar ajuste de escala automático vertical do Amazon EMR no EKS
- Desinstalação do operador do Spark para o Amazon EMR no EKS
- Segurança e o operador do Spark com o Amazon EMR no EKS

Configuração do operador do Spark para o Amazon EMR no EKS

Conclua as tarefas apresentadas a seguir para se preparar antes de instalar o operador do Spark no Amazon EKS. Se você já se inscreveu na Amazon Web Services (AWS) e usou o Amazon EKS, está com quase tudo pronto para usar o Amazon EMR no EKS. Conclua as tarefas apresentadas a seguir para se preparar para usar o operador do Spark no Amazon EKS. Se você já completou algum dos pré-requisitos, pode ignorá-los e passar para os próximos.

- Instalar a AWS CLI: se você já instalou a AWS CLI, confirme se tem a versão mais recente.
- Instale o eksctl: o eksctl é uma ferramenta de linha de comando que você usa para se comunicar com o Amazon EKS.

- Instale o Helm: o gerenciador de pacotes Helm para o Kubernetes ajuda a instalar e gerenciar aplicações em seu cluster do Kubernetes.
- Configure um cluster do Amazon EKS: siga as etapas para criar um novo cluster do Kubernetes com nós no Amazon EKS.
- Selecione um URI de imagem base do Amazon EMR (versão 6.10.0 ou superiores): o operador do Spark é compatível com as versões 6.10.0 e superiores do Amazon EMR.

Conceitos básicos do operador do Spark para o Amazon EMR no EKS

Este tópico ajuda você a começar a usar o operador do Spark no Amazon EKS ao implantar uma aplicação do Spark e uma aplicação programada do Spark.

Instalação do operador do Spark

Use as etapas apresentadas a seguir para instalar o operador do Kubernetes para Apache Spark.

1. Caso ainda não tenha feito, conclua as etapas em Configuração do operador do Spark para o Amazon EMR no EKS.
2. Autentique seu cliente Helm no registro do Amazon ECR. No comando apresentado a seguir, substitua os valores de *region-id* pela Região da AWS de sua preferência e pelo valor *ECR-registry-account* correspondente para a região, estabelecido na página Contas de registro do Amazon ECR por região.

   ```
   aws ecr get-login-password \
   --region region-id | helm registry login \
   --username AWS \
   --password-stdin ECR-registry-account.dkr.ecr.region-id.amazonaws.com
   ```

3. Instale o operador do Spark com o comando apresentado a seguir.

   Para o parâmetro `--version` do chart do Helm, use o rótulo de versão do Amazon EMR com o prefixo `emr-` e o sufixo de data removidos. Por exemplo, com a versão `emr-6.12.0-java17-latest`, especifique `6.12.0-java17`. O exemplo com o comando apresentado a seguir usa a versão `emr-6.14.0-latest`, portanto, especifica `6.14.0` para o chart do Helm `--version`.

   ```
   helm install spark-operator-demo \
     oci://895885662937.dkr.ecr.region-id.amazonaws.com/spark-operator \
     --set emrContainers.awsRegion=region-id \
     --version 6.14.0 \
   ```

```
  --namespace spark-operator \
  --create-namespace
```

Por padrão, o comando cria uma conta de serviço `emr-containers-sa-spark-operator` para o operador do Spark. Para usar uma conta de serviço diferente, forneça o argumento `serviceAccounts.sparkoperator.name`. Por exemplo:

```
--set serviceAccounts.sparkoperator.name my-service-account-for-spark-operator
```

Para usar o escalonamento automático vertical com o operador Spark, adicione a seguinte linha ao comando de instalação para permitir webhooks para o operador:

```
--set webhook.enable=true
```

4. Verifique se você instalou o chart do Helm com o comando `helm list`:

```
helm list --namespace spark-operator -o yaml
```

O comando `helm list` deve retornar as informações da versão do chart do Helm recém-implantado:

```
app_version: v1beta2-1.3.8-3.1.1
chart: spark-operator-6.14.0
name: spark-operator-demo
namespace: spark-operator
revision: "1"
status: deployed
updated: 2023-03-14 18:20:02.721638196 +0000 UTC
```

5. Conclua a instalação com quaisquer opções adicionais necessárias. Para obter mais informações, consulte a documentação `spark-on-k8s-operator` no GitHub.

Execução de uma aplicação do Spark

O operador do Spark é compatível com o Amazon EMR 6.10.0 ou com versões superiores. Ao instalar o operador do Spark, ele cria a conta de serviço `emr-containers-sa-spark` para executar aplicações do Spark por padrão. Use as etapas apresentadas a seguir para executar uma aplicação do Spark com o operador do Spark no Amazon EMR no EKS 6.10.0 ou em versões superiores.

1. Antes de executar uma aplicação do Spark com o operador do Spark, conclua as etapas em Configuração do operador do Spark para o Amazon EMR no EKS e Instalação do operador do Spark.
2. Crie um arquivo de definição `SparkApplication` `spark-pi.yaml` com o seguinte conteúdo de exemplo:

```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: spark-pi
  namespace: spark-operator
spec:
  type: Scala
  mode: cluster
  image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.10.0:latest"
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "local:///usr/lib/spark/examples/jars/spark-examples.jar"
  sparkVersion: "3.3.1"
  restartPolicy:
    type: Never
  volumes:
    - name: "test-volume"
      hostPath:
        path: "/tmp"
        type: Directory
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.3.1
    serviceAccount: emr-containers-sa-spark
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
  executor:
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      version: 3.3.1
```

```
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
```

3. Agora, envie a aplicação do Spark com o comando apresentado a seguir. Isso também criará um objeto `SparkApplication` chamado `spark-pi`:

```
kubectl apply -f spark-pi.yaml
```

4. Verifique os eventos do objeto `SparkApplication` com o seguinte comando:

```
kubectl describe sparkapplication spark-pi --namespace spark-operator
```

Para obter mais informações sobre o envio de aplicações ao Spark usando o operador do Spark, consulte Using a `SparkApplication` na documentação `spark-on-k8s-operator` no GitHub.

Usar ajuste de escala automático vertical do Amazon EMR no EKS

O Ajuste de Escala Automático vertical do Amazon EMR no EKS simplifica o gerenciamento de recursos. Ela ajusta automaticamente os recursos de memória e de CPU para se adaptar às necessidades da workload fornecida para aplicações do Spark no Amazon EMR. Para obter mais informações, consulte Uso da escalabilidade automática vertical com trabalhos do Spark no Amazon EMR .

Esta seção descreve como configurar o operador Spark para usar o ajuste de escala automático vertical.

Pré-requisitos

Antes de usar continuar, verifique se você concluiu estas configurações:

- Siga as etapas em Configuração do operador do Spark para o Amazon EMR no EKS.
- Siga as etapas em Instalação do operador do Spark. Na etapa 3, adicione a seguinte linha ao comando de instalação para permitir webhooks para o operador:

```
--set webhook.enable=true
```

Execute um trabalho com ajuste de escala automático vertical no operador do Spark

Antes de executar uma aplicação do Spark com o operador do Spark, conclua as etapas em Pré-requisitos.

Para usar o ajuste de escala automático vertical com o operador do Spark, há várias configurações necessárias que devem ser adicionadas à aplicação. Você pode encontrar um exemplo abaixo desta lista.

Configurações do driver

Adicione a seguinte anotação à configuração do driver:

```
emr-containers.amazonaws.com/dynamic.sizing.signature: "YOUR_JOB_SIGNATURE"
```

Adicione o seguinte rótulo à configuração da aplicação:

```
emr-containers.amazonaws.com/dynamic.sizing: "true"
```

Configurações do executor

Adicione o seguinte rótulo à configuração do executor:

```
emr-containers.amazonaws.com/dynamic.sizing.signature: "YOUR_JOB_SIGNATURE"
```

Defina as seguintes variáveis de ambiente para a configuração do executor:

```
- name: DYNAMIC_SIZING_ENABLED
        value: "true" # true, not set otherwise
- name: OVERHEAD_FACTOR
        value: "0.1" # value of spark.kubernetes.memoryOverheadFactor
                     # or spark.driver.memoryOverheadFactor, not set otherwise
- name: PYSPARK_MEM
        value: "0.1" # value of spark.executor.pyspark.memory, not set otherwise
- name: EXEC_POD_CPU_REQUEST
        valueFrom:
          resourceFieldRef:
            containerName: spark-kubernetes-executor
            resource: requests.cpu
            divisor: 1
- name: EXEC_POD_CPU_LIMIT
```

```
        valueFrom:
          resourceFieldRef:
            containerName: spark-kubernetes-executor
            resource: limits.cpu
            divisor: 1
- name: EXEC_POD_MEM_REQUEST
        valueFrom:
          resourceFieldRef:
            containerName: spark-kubernetes-executor
            resource: requests.memory
            divisor: 1
- name: EXEC_POD_MEM_LIMIT
        valueFrom:
          resourceFieldRef:
            containerName: spark-kubernetes-executor
            resource: limits.memory
            divisor: 1
```

O exemplo a seguir mostra um arquivo de definição `SparkApplication` `spark-pi.yaml` com as configurações necessárias para usar ajuste de escala automático vertical:

```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: spark-pi
  namespace: spark-operator
spec:
  type: Scala
  mode: cluster
  image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.10.0:latest"
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "local:///usr/lib/spark/examples/jars/spark-examples.jar"
  arguments:
    - "10000"
  sparkVersion: "3.3.1"
  sparkConf:
    spark.kubernetes.executor.deleteOnTermination: "true"
  restartPolicy:
    type: Never
  volumes:
    - name: "test-volume"
```

```
      hostPath:
        path: "/tmp"
        type: Directory
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    annotations:
       emr-containers.amazonaws.com/dynamic.sizing.signature: "my-signature"
    labels:
       emr-containers.amazonaws.com/dynamic.sizing: "true"
       version: 3.3.1
    serviceAccount: emr-containers-sa-spark
    volumeMounts:
       - name: "test-volume"
         mountPath: "/tmp"
  executor:
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      emr-containers.amazonaws.com/dynamic.sizing.signature: "my-signature"
      version: 3.3.1
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
    env:
      - name: "DYNAMIC_SIZING_ENABLED"
        value: "true" # true, not set otherwise
      - name: "EXEC_POD_CPU_REQUEST"
        valueFrom:
          resourceFieldRef:
            containerName: "spark-kubernetes-executor"
            resource: "requests.cpu"
            divisor: "1"
      - name: "EXEC_POD_CPU_LIMIT"
        valueFrom:
          resourceFieldRef:
            containerName: "spark-kubernetes-executor"
            resource: "limits.cpu"
            divisor: "1"
      - name: "EXEC_POD_MEM_REQUEST"
        valueFrom:
          resourceFieldRef:
```

```
            containerName: "spark-kubernetes-executor"
            resource: "requests.memory"
            divisor: "1"
      - name: "EXEC_POD_MEM_LIMIT"
        valueFrom:
          resourceFieldRef:
            containerName: "spark-kubernetes-executor"
            resource: "limits.memory"
            divisor: "1"
```

Agora, envie a aplicação do Spark com o comando apresentado a seguir. Isso também criará um objeto `SparkApplication` chamado `spark-pi`:

```
kubectl apply -f spark-pi.yaml
```

Para obter mais informações sobre o envio de aplicações ao Spark usando o operador do Spark, consulte Using a `SparkApplication` na documentação `spark-on-k8s-operator` no GitHub.

Verificação de funcionalidade da escalabilidade automática vertical

Para verificar se a escalabilidade automática vertical está funcionando corretamente para o trabalho enviado, use kubectl para obter o recurso personalizado `verticalpodautoscaler` e visualizar suas recomendações de escalabilidade.

```
kubectl get verticalpodautoscalers --all-namespaces \
-l=emr-containers.amazonaws.com/dynamic.sizing.signature=my-signature
```

A saída desta consulta deve ser semelhante à seguinte:

```
NAMESPACE        NAME                                                    MODE
 CPU   MEM         PROVIDED   AGE
spark-operator   ds-p73j6mkosvc4xeb3gr7x4xol2bfcw5evqimzqojrlysvj3giozuq-vpa   Off
     580026651   True       15m
```

Se a saída não for semelhante ou tiver um código de erro, consulte Solução de problemas de escalabilidade automática vertical do Amazon EMR no EKS para obter passos que ajudam a resolver o problema.

Desinstalação do operador do Spark para o Amazon EMR no EKS

Use as etapas apresentadas a seguir para desinstalar o operador do Spark.

1. Exclua o operador do Spark usando o namespace correto. Para este exemplo, o namespace é `spark-operator-demo`.

```
helm uninstall spark-operator-demo -n spark-operator
```

2. Exclua a conta de serviço do operador do Spark:

```
kubectl delete sa emr-containers-sa-spark-operator -n spark-operator
```

3. Exclua o `CustomResourceDefinitions` (CRDs) do operador do Spark:

```
kubectl delete crd sparkapplications.sparkoperator.k8s.io
kubectl delete crd scheduledsparkapplications.sparkoperator.k8s.io
```

Segurança e o operador do Spark com o Amazon EMR no EKS

Tópicos

- Configuração de permissões de acesso ao cluster com o controle de acesso por perfil (RBAC)
- Configuração de permissões de acesso ao cluster com perfis do IAM para contas de serviço (IRSA)

Configuração de permissões de acesso ao cluster com o controle de acesso por perfil (RBAC)

Para implantar o operador do Spark, o Amazon EMR no EKS cria dois perfis e duas contas de serviço para o operador do Spark e para as aplicações do Spark.

Tópicos

- Perfil e conta de serviço do operador
- Perfil e conta de serviço do Spark

Perfil e conta de serviço do operador

O Amazon EMR no EKS cria o perfil e a conta de serviço do operador para gerenciar `SparkApplications` para trabalhos do Spark e para outros recursos, como serviços.

O nome padrão para esta conta de serviço é `emr-containers-sa-spark-operator`.

As seguintes regras se aplicam para este perfil de serviço:

```
 rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - services
  - configmaps
  - secrets
  verbs:
  - create
  - get
  - delete
  - update
- apiGroups:
  - extensions
  - networking.k8s.io
  resources:
  - ingresses
  verbs:
  - create
  - get
  - delete
- apiGroups:
  - ""
  resources:
  - nodes
  verbs:
  - get
- apiGroups:
  - ""
  resources:
  - events
  verbs:
  - create
  - update
  - patch
- apiGroups:
```

```
  - ""
  resources:
  - resourcequotas
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - apiextensions.k8s.io
  resources:
  - customresourcedefinitions
  verbs:
  - create
  - get
  - update
  - delete
- apiGroups:
  - admissionregistration.k8s.io
  resources:
  - mutatingwebhookconfigurations
  - validatingwebhookconfigurations
  verbs:
  - create
  - get
  - update
  - delete
- apiGroups:
  - sparkoperator.k8s.io
  resources:
  - sparkapplications
  - sparkapplications/status
  - scheduledsparkapplications
  - scheduledsparkapplications/status
  verbs:
  - "*"
  {{- if .Values.batchScheduler.enable }}
  # required for the `volcano` batch scheduler
- apiGroups:
  - scheduling.incubator.k8s.io
  - scheduling.sigs.dev
  - scheduling.volcano.sh
  resources:
  - podgroups
  verbs:
```

```
  - "*"
  {{- end }}
  {{ if .Values.webhook.enable }}
- apiGroups:
  - batch
  resources:
  - jobs
  verbs:
  - delete
  {{- end }}
```

Perfil e conta de serviço do Spark

Um pod do driver do Spark precisa de uma conta de serviço do Kubernetes no mesmo namespace que o pod. Esta conta de serviço precisa de permissões para criar, obter, listar, aplicar patches e excluir pods de executores e para criar um serviço descentralizado do Kubernetes para o driver. O driver apresentará falhas e será encerrado sem a conta de serviço, a menos que a conta de serviço padrão no namespace do pod tenha as permissões obrigatórias.

O nome padrão para esta conta de serviço é `emr-containers-sa-spark`.

As seguintes regras se aplicam para este perfil de serviço:

```
 rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - configmaps
  verbs:
  - "*"
- apiGroups:
```

```
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - "*"
```

Configuração de permissões de acesso ao cluster com perfis do IAM para contas de serviço (IRSA)

Esta seção usa um exemplo para demonstrar como configurar uma conta de serviço do Kubernetes para assumir um perfil do AWS Identity and Access Management. Os pods que usam a conta de serviço podem acessar qualquer serviço da AWS ao qual o perfil tenha permissões de acesso.

O exemplo apresentado a seguir executa uma aplicação do Spark para contar as palavras de um arquivo no Amazon S3. Para fazer isso, você pode configurar perfis do IAM para contas de serviço (IRSA) com a finalidade de autenticar e autorizar as contas de serviço do Kubernetes.

> **Note**
>
> Este exemplo usa o namespace “spark-operator” para o operador do Spark e para o namespace no qual a aplicação do Spark é enviada.

Pré-requisitos

Antes de testar o exemplo apresentado nesta página, complete os seguintes pré-requisitos:

- Prepare-se para usar o operador do Spark.
- Instalação do operador do Spark.
- Crie um bucket do Amazon S3.
- Salve seu poema favorito em um arquivo de texto chamado `poem.txt` e faça o upload do arquivo em seu bucket do S3. A aplicação do Spark criada nesta página realizará a leitura do conteúdo do arquivo de texto. Para obter mais informações sobre como fazer o upload de arquivos para o S3, consulte Fazer upload de um objeto para o seu bucket no Guia do usuário do Amazon Simple Storage Service.

Configuração de uma conta de serviço do Kubernetes para assumir um perfil do IAM

Use as etapas apresentadas a seguir para configurar uma conta de serviço do Kubernetes para assumir um perfil do IAM que os pods poderão usar para acessar os serviços da AWS aos quais o perfil tem permissões de acesso.

1. Após concluir os Pré-requisitos, use a AWS Command Line Interface para criar um arquivo `example-policy.json` que permita acesso somente leitura ao arquivo que você fez upload no Amazon S3:

```
cat >example-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::my-pod-bucket",
                "arn:aws:s3:::my-pod-bucket/*"
            ]
        }
    ]
}
EOF
```

2. Em seguida, crie uma política do IAM `example-policy`:

```
aws iam create-policy --policy-name example-policy --policy-document file://
example-policy.json
```

3. Depois disso, crie um perfil do IAM `example-role` e associe-o a uma conta de serviço do Kubernetes para o driver do Spark:

```
eksctl create iamserviceaccount --name driver-account-sa --namespace spark-operator
 \
--cluster my-cluster --role-name "example-role" \
--attach-policy-arn arn:aws:iam::111122223333:policy/example-policy --approve
```

4. Crie um arquivo em YAML com as associações de perfil do cluster que são obrigatórias para a conta de serviço do driver do Spark:

```
cat >spark-rbac.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  name: driver-account-sa
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: spark-role
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: edit
subjects:
  - kind: ServiceAccount
    name: driver-account-sa
    namespace: spark-operator
EOF
```

5. Aplique as configurações de associação de perfil do cluster:

```
kubectl apply -f spark-rbac.yaml
```

O comando kubectl deve confirmar a criação com êxito da conta:

```
serviceaccount/driver-account-sa created
clusterrolebinding.rbac.authorization.k8s.io/spark-role configured
```

Execução de uma aplicação do operador do Spark

Após configurar a conta de serviço do Kubernetes, você poderá executar uma aplicação do Spark que conta o número de palavras no arquivo de texto carregado como parte dos Pré-requisitos.

1. Crie um novo arquivo `word-count.yaml`, com uma definição `SparkApplication` para sua aplicação de contagem de palavras.

```
cat >word-count.yaml <<EOF
```

```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: word-count
  namespace: spark-operator
spec:
  type: Java
  mode: cluster
  image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.10.0:latest"
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.JavaWordCount
  mainApplicationFile: local:///usr/lib/spark/examples/jars/spark-examples.jar
  arguments:
    - s3://my-pod-bucket/poem.txt
  hadoopConf:
   # EMRFS filesystem
    fs.s3.customAWSCredentialsProvider:
 com.amazonaws.auth.WebIdentityTokenCredentialsProvider
    fs.s3.impl: com.amazon.ws.emr.hadoop.fs.EmrFileSystem
    fs.AbstractFileSystem.s3.impl: org.apache.hadoop.fs.s3.EMRFSDelegate
    fs.s3.buffer.dir: /mnt/s3
    fs.s3.getObject.initialSocketTimeoutMilliseconds: "2000"

 mapreduce.fileoutputcommitter.algorithm.version.emr_internal_use_only.EmrFileSystem:
 "2"
    mapreduce.fileoutputcommitter.cleanup-
failures.ignored.emr_internal_use_only.EmrFileSystem: "true"
  sparkConf:
    # Required for EMR Runtime
    spark.driver.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/
hadoop-aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/
share/aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-
glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*
    spark.driver.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/
lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/native
    spark.executor.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/
hadoop-aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/
share/aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-
glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
```

```
serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*
    spark.executor.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-
lzo/lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/
native
  sparkVersion: "3.3.1"
  restartPolicy:
    type: Never
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.3.1
    serviceAccount: my-spark-driver-sa
  executor:
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      version: 3.3.1
EOF
```

2. Envie a aplicação do Spark.

```
kubectl apply -f word-count.yaml
```

O comando kubectl deve retornar a confirmação de que você criou com êxito um objeto `SparkApplication` chamado `word-count`.

```
sparkapplication.sparkoperator.k8s.io/word-count configured
```

3. Para verificar os eventos do objeto `SparkApplication`, execute o seguinte comando:

```
kubectl describe sparkapplication word-count -n spark-operator
```

O comando kubectl deve retornar a descrição da `SparkApplication` com os eventos:

```
Events:
  Type     Reason                                Age                    From
    Message
```

```
  ----     ------                                  ----                  ----
    -------
  Normal   SparkApplicationSpecUpdateProcessed  3m2s (x2 over 17h)     spark-
operator  Successfully processed spec update for SparkApplication word-count
  Warning  SparkApplicationPendingRerun         3m2s (x2 over 17h)     spark-
operator  SparkApplication word-count is pending rerun
  Normal   SparkApplicationSubmitted            2m58s (x2 over 17h)    spark-
operator  SparkApplication word-count was submitted successfully
  Normal   SparkDriverRunning                   2m56s (x2 over 17h)    spark-
operator  Driver word-count-driver is running
  Normal   SparkExecutorPending                 2m50s                  spark-
operator  Executor [javawordcount-fdd1698807392c66-exec-1] is pending
  Normal   SparkExecutorRunning                 2m48s                  spark-
operator  Executor [javawordcount-fdd1698807392c66-exec-1] is running
  Normal   SparkDriverCompleted                2m31s (x2 over 17h)    spark-
operator  Driver word-count-driver completed
  Normal   SparkApplicationCompleted            2m31s (x2 over 17h)    spark-
operator  SparkApplication word-count completed
  Normal   SparkExecutorCompleted               2m31s (x2 over 2m31s)  spark-
operator  Executor [javawordcount-fdd1698807392c66-exec-1] completed
```

Agora, a aplicação está realizando a contagem das palavras em seu arquivo do S3. Para localizar a contagem de palavras, consulte os arquivos de log do seu driver:

```
kubectl logs pod/word-count-driver -n spark-operator
```

O comando kubectl deve retornar o conteúdo do arquivo de log com os resultados da sua aplicação de contagem de palavras.

```
INFO DAGScheduler: Job 0 finished: collect at JavaWordCount.java:53, took 5.146519 s
               Software: 1
```

Para obter mais informações sobre como enviar aplicações ao Spark usando o operador do Spark, consulte Using a SparkApplication na documentação Kubernetes Operator for Apache Spark (spark-on-k8s-operator) no GitHub.

Execução de trabalhos do Spark com spark-submit

As versões 6.10.0 e superiores do Amazon EMR oferecem suporte ao `spark-submit` como uma ferramenta de linha de comando que você pode usar para enviar e executar aplicações do Spark para um cluster do Amazon EMR no EKS.

> **Note**
>
> O Amazon EMR calcula os preços do Amazon EKS com base nos recursos de vCPU e de memória usados no pod do operador desde o momento em que o download da imagem da aplicação do Amazon EMR tem início até o encerramento do pod do Amazon EKS, arredondado para o segundo mais próximo.

Tópicos

- Configuração do spark-submit para o Amazon EMR no EKS
- Conceitos básicos do spark-submit para o Amazon EMR no EKS
- Requisitos de segurança da conta de serviço do driver do Spark para spark-submit

Configuração do spark-submit para o Amazon EMR no EKS

Conclua as tarefas apresentadas a seguir para se preparar antes de executar uma aplicação com spark-submit no Amazon EMR no EKS. Se você já se inscreveu na Amazon Web Services (AWS) e usou o Amazon EKS, está com quase tudo pronto para usar o Amazon EMR no EKS. Se você já completou algum dos pré-requisitos, pode ignorá-los e passar para os próximos.

- Instalar a AWS CLI: se você já instalou a AWS CLI, confirme se tem a versão mais recente.
- Instale o eksctl: o eksctl é uma ferramenta de linha de comando que você usa para se comunicar com o Amazon EKS.
- Configure um cluster do Amazon EKS: siga as etapas para criar um novo cluster do Kubernetes com nós no Amazon EKS.
- Selecione um URI de imagem base do Amazon EMR (versão 6.10.0 ou superiores): o comando `spark-submit` é compatível com as versões 6.10.0 e superiores do Amazon EMR.
- Confirme se a conta de serviço do driver tem permissões apropriadas para criar e monitorar os pods de executores. Para obter mais informações, consulte Requisitos de segurança da conta de serviço do driver do Spark para spark-submit.

- Configure seu perfil de credenciais locais da AWS.
- Obtenha o endpoint do cluster do Amazon EKS.

Conceitos básicos do spark-submit para o Amazon EMR no EKS

Execução de uma aplicação do Spark

O Amazon EMR 6.10.0 e versões superiores oferecem suporte ao spark-submit para a execução de aplicações do Spark em um cluster do Amazon EKS. Para executar a aplicação do Spark, siga estas etapas:

1. Antes de executar uma aplicação do Spark com o comando `spark-submit`, conclua as etapas em Configuração do spark-submit para o Amazon EMR no EKS.
2. Defina os valores para as seguintes variáveis de ambiente:

```
export SPARK_HOME=spark-home
export MASTER_URL=k8s://Amazon EKS-cluster-endpoint
```

3. Agora, envie a aplicação do Spark com o seguinte comando:

```
$SPARK_HOME/bin/spark-submit \
 --class org.apache.spark.examples.SparkPi \
 --master $MASTER_URL \
 --conf spark.kubernetes.container.image=895885662937.dkr.ecr.us-
west-2.amazonaws.com/spark/emr-6.10.0:latest \
 --conf spark.kubernetes.authenticate.driver.serviceAccountName=spark \
 --deploy-mode cluster \
 --conf spark.kubernetes.namespace=spark-operator \
 local:///usr/lib/spark/examples/jars/spark-examples.jar 20
```

Para obter mais informações sobre o envio de aplicações para o Spark, consulte Submitting applications na documentação do Apache Spark.

> **Important**
>
> O `spark-submit` oferece suporte somente para o modo de cluster como o mecanismo de envio.

Requisitos de segurança da conta de serviço do driver do Spark para spark-submit

O pod do driver do Spark usa uma conta de serviço do Kubernetes para acessar o servidor da API do Kubernetes para criar e monitorar pods de executores. A conta de serviço do driver deve ter as permissões apropriadas para listar, criar, editar, aplicar patches e excluir pods em seu cluster. É possível verificar se consegue listar esses recursos ao executar o seguinte comando:

```
kubectl auth can-i list|create|edit|delete|patch pods
```

As seguintes regras se aplicam para este perfil de serviço:

```
 rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - configmaps
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - "*"
```

Gerenciamento de execuções de trabalhos do Amazon EMR no EKS

As seções a seguir abordam tópicos que ajudam você a gerenciar suas execuções de trabalhos do Amazon EMR no EKS.

Tópicos

- Gerenciamento de execuções de trabalhos com a AWS CLI
- Execução de scripts do Spark SQL por meio da API StartJobRun
- Estados de execução de trabalho
- Visualização de trabalhos no console do Amazon EMR
- Erros comuns ao executar trabalhos

Gerenciamento de execuções de trabalhos com a AWS CLI

Esta página aborda como gerenciar execuções de trabalhos com a AWS Command Line Interface (AWS CLI).

Opções para a configuração de uma execução de trabalho

Use as seguintes opções para configurar os parâmetros de execução de trabalho:

- `--execution-role-arn`: você deve fornecer um perfil do IAM que é usado para a execução de trabalhos. Para obter mais informações, consulte Uso de perfis de execução de trabalho com o Amazon EMR no EKS.
- `--release-label`: você pode implantar o Amazon EMR no EKS com as versões 5.32.0 e 6.2.0 e posteriores do Amazon EMR. O Amazon EMR no EKS não é compatível com as versões anteriores do Amazon EMR. Para obter mais informações, consulte Versões do Amazon EMR no EKS.
- `--job-driver`: o driver de trabalho usado para fornecer entradas sobre o trabalho principal. Este é um campo do tipo união no qual você só pode transferir um dos valores para o tipo de trabalho que deseja executar. Os tipos de trabalho com suporte incluem:
  - Trabalhos do Spark-submit: usados para executar um comando por meio do spark-submit. Você pode usar este tipo de trabalho para executar o Scala, o PySpark, o Spark R, o Spark SQL e quaisquer outros trabalhos com suporte por meio do spark-submit. Esse tipo de trabalho tem os seguintes parâmetros:

- Entrypoint: esta é a referência do HCFS (sistema de arquivos compatível com Hadoop) para o arquivo principal em JAR/PY que você deseja executar.
- EntryPointArguments: esta é uma matriz de argumentos que você deseja transferir para seu arquivo principal em JAR/PY. Você deve realizar a leitura desses parâmetros usando seu código de Entrypoint. Cada argumento na matriz deve ser separado por uma vírgula. Os argumentos do EntryPointArguments não podem conter colchetes ou parênteses, como (), {} ou [].
- SparkSubmitParameters: esses são os parâmetros adicionais do Spark que você deseja enviar para o trabalho. Use este parâmetro para substituir as propriedades padrão do Spark, como a memória do driver ou o número de executores, como —conf ou —class. Para obter informações adicionais, consulte Launching Applications with spark-submit.

- Trabalhos do Spark SQL: usados para executar um arquivo de consulta SQL por meio do Spark SQL. Você pode usar esse tipo de trabalho para executar trabalhos do Spark SQL. Esse tipo de trabalho tem os seguintes parâmetros:
  - Entrypoint: esta é a referência do HCFS (sistema de arquivos compatível com Hadoop) para o arquivo de consulta SQL que você deseja executar.

    Para obter uma lista de parâmetros adicionais do Spark que você pode usar para um trabalho do Spark SQL, consulte Execução de scripts do Spark SQL por meio da API StartJobRun.

- `--configuration-overrides`: você pode substituir as configurações padrão para as aplicações ao fornecer um objeto de configuração. Você pode usar uma sintaxe abreviada para fornecer a configuração ou fazer referência ao objeto de configuração em um arquivo JSON. Os objetos de configuração consistem em uma classificação, propriedades e configurações opcionais aninhadas As propriedades consistem nas configurações que você deseja substituir neste arquivo. Você pode especificar várias classificações para diversas aplicações em um único objeto JSON. As classificações de configuração disponíveis variam de acordo com a versão do Amazon EMR. Para obter uma lista das classificações de configuração disponíveis para cada versão de liberação do Amazon EMR, consulte Versões do Amazon EMR no EKS.

  Se você transferir a mesma configuração em uma substituição de aplicação e nos parâmetros do spark-submit, os parâmetros do spark-submit terão precedência. A lista completa de prioridades de configuração é apresentada a seguir, na ordem da prioridade mais alta para a prioridade mais baixa.

  - Configuração fornecida ao criar `SparkSession`.
  - Configuração fornecida como parte do `sparkSubmitParameters` usando `–conf`.

- Configuração fornecida como parte das substituições de aplicações.
- Configurações otimizadas escolhidas pelo Amazon EMR para a versão.
- Configurações padrão de código aberto para a aplicação.

Para monitorar as execuções de trabalhos usando o Amazon CloudWatch ou o Amazon S3, você deve fornecer os detalhes de configuração do CloudWatch. Para obter mais informações, consulte Configuração de uma execução de trabalho para usar logs do Amazon S3 e Configuração de uma execução de trabalho para usar o Amazon CloudWatch Logs. Se o bucket do S3 ou o grupo de logs do CloudWatch não existir, o Amazon EMR realizará a criação antes de fazer o upload dos logs para o bucket.

- Para obter uma lista adicional de opções de configuração do Kubernetes, consulte Spark Properties on Kubernetes.

  As configurações do Spark apresentadas a seguir não têm suporte.

  - `spark.kubernetes.authenticate.driver.serviceAccountName`
  - `spark.kubernetes.authenticate.executor.serviceAccountName`
  - `spark.kubernetes.namespace`
  - `spark.kubernetes.driver.pod.name`
  - `spark.kubernetes.container.image.pullPolicy`
  - `spark.kubernetes.container.image`

  > **Note**
  >
  > Você pode usar `spark.kubernetes.container.image` para imagens do Docker personalizadas. Para obter mais informações, consulte Personalização de imagens do Docker para o Amazon EMR no EKS.

Configuração de uma execução de trabalho para usar logs do Amazon S3

Para poder monitorar o progresso do trabalho e solucionar falhas, você deve configurar os trabalhos para enviar informações de log ao Amazon S3, ao Amazon CloudWatch Logs ou a ambos. Este tópico ajuda você a começar a publicar logs de aplicações no Amazon S3 em trabalhos iniciados com o Amazon EMR no EKS.

Política do IAM para logs do S3

Antes que os trabalhos possam enviar dados de log ao Amazon S3, as permissões apresentadas a seguir devem ser inclusas na política de permissões para o perfil de execução do trabalho. Substitua *DOC-EXAMPLE-BUCKET-LOGGING* pelo nome do bucket de registro em log.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-LOGGING",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-LOGGING/*",
            ]
        }
    ]
}
```

> **Note**
>
> O Amazon EMR no EKS também pode criar um bucket do Amazon S3. Se um bucket do Amazon S3 não estiver disponível, inclua a permissão "`s3:CreateBucket`" na política do IAM.

Depois de conceder ao perfil de execução as permissões adequadas para enviar logs ao Amazon S3, os dados de log serão enviados para os locais do Amazon S3 apresentados a seguir quando `s3MonitoringConfiguration` for transmitida na seção `monitoringConfiguration` de uma solicitação `start-job-run`, conforme mostrado em Gerenciamento de execuções de trabalhos com a AWS CLI.

- Logs do controlador: /*logUri*/*virtual-cluster-id*/jobs/*job-id*/containers/*pod-name*/(stderr.gz/stdout.gz)
- Logs do driver: /*logUri*/*virtual-cluster-id*/jobs/*job-id*/containers/*spark-application-id*/spark-*job-id*-driver/(stderr.gz/stdout.gz)

- Logs do executor: /*logUri*/*virtual-cluster-id*/jobs/*job-id*/containers/*spark-application-id*/*executor-pod-name*/(stderr.gz/stdout.gz)

Configuração de uma execução de trabalho para usar o Amazon CloudWatch Logs

Para monitorar o progresso do trabalho e solucionar falhas, você deve configurar os trabalhos para enviar informações de log ao Amazon S3, ao Amazon CloudWatch Logs ou a ambos. Este tópico ajuda você a começar a usar o CloudWatch Logs em trabalhos iniciados com o Amazon EMR no EKS. Para obter mais informações sobre o CloudWatch Logs, consulte Monitoring Log Files no Guia do usuário do Amazon CloudWatch.

Política do IAM para o CloudWatch Logs

Para que seus trabalhos enviem dados de log ao CloudWatch Logs, as permissões apresentadas a seguir devem ser inclusas na política de permissões para o perfil de execução do trabalho. Substitua *my_log_group_name* e *my_log_stream_prefix* pelos nomes do grupo de logs do CloudWatch e pelos nomes do fluxo de logs, respectivamente. O Amazon EMR no EKS cria o grupo de logs e o fluxo de logs se eles não existirem, desde que o ARN do perfil de execução tenha as permissões apropriadas.

```
{
     "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
```

```
                "arn:aws:logs:*:*:log-group:my_log_group_name:log-
stream:my_log_stream_prefix/*"
            ]
        }
    ]
}
```

> **Note**
>
> O Amazon EMR no EKS também pode criar um fluxo de logs. Se um fluxo de logs não existir, a política do IAM deverá incluir a permissão `"logs:CreateLogGroup"`.

Depois que você conceder as permissões adequadas ao perfil de execução, a aplicação enviará os dados de log para o CloudWatch Logs quando `cloudWatchMonitoringConfiguration` for transmitida na seção `monitoringConfiguration` de uma solicitação `start-job-run`, conforme mostrado em Gerenciamento de execuções de trabalhos com a AWS CLI.

Na API `StartJobRun`, *`log_group_name`* é o nome do grupo de logs do CloudWatch e *`log_stream_prefix`* é o prefixo do nome do fluxo de logs do CloudWatch. Você pode visualizar e pesquisar esses logs no AWS Management Console.

- Logs do controlador: *`logGroup`*/*`logStreamPrefix`*/*`virtual-cluster-id`*/jobs/*`job-id`*/containers/*`pod-name`*/(stderr/stdout)
- Logs do driver: *`logGroup`*/*`logStreamPrefix`*/*`virtual-cluster-id`*/jobs/*`job-id`*/containers/*`spark-application-id`*/spark-*`job-id`*-driver/(stderrstdout)
- Logs do executor: *`logGroup`*/*`logStreamPrefix`*/*`virtual-cluster-id`*/jobs/*`job-id`*/containers/*`spark-application-id`*/*`executor-pod-name`*/(stderr/stdout)

Listagem de execuções de trabalhos

Você pode executar `list-job-run` para mostrar os estados das execuções de trabalhos, como demonstra o exemplo a seguir.

```
aws emr-containers list-job-runs --virtual-cluster-id <cluster-id>
```

Descrição de uma execução de trabalho

Você pode executar `describe-job-run` para obter mais detalhes sobre o trabalho, como o estado do trabalho, os detalhes do estado e o nome do trabalho, como demonstra o exemplo a seguir.

```
aws emr-containers describe-job-run --virtual-cluster-id cluster-id --id job-run-id
```

Cancelamento de uma execução de trabalho

Você pode executar `cancel-job-run` para cancelar trabalhos em execução, como demonstra o exemplo a seguir.

```
aws emr-containers cancel-job-run --virtual-cluster-id cluster-id --id job-run-id
```

Execução de scripts do Spark SQL por meio da API StartJobRun

As versões 6.7.0 e posteriores do Amazon EMR no EKS incluem um driver de trabalho do Spark SQL para que você possa executar scripts do Spark SQL por meio da API `StartJobRun`. Você pode fornecer arquivos de ponto de entrada SQL para executar consultas do Spark SQL diretamente no Amazon EMR no EKS com a API `StartJobRun`, sem a necessidade de realizar modificações nos scripts do Spark SQL existentes. A tabela a seguir lista os parâmetros do Spark que têm suporte com os trabalhos do Spark SQL por meio da API StartJobRun.

Você pode escolher entre os parâmetros do Spark apresentados a seguir para enviar para um trabalho do Spark SQL. Use esses parâmetros para substituir as propriedades padrão do Spark.

Opção	Descrição
--name NAME	Nome do aplicativo
--jars JARS	Lista separada por vírgulas de arquivos em JARs a serem inclusos no driver e no caminho de classe de execução.
--packages	Lista separada por vírgulas de coordenadas do Maven de arquivos em JARs a serem inclusas nos caminhos de classe do driver e do executor.

Opção	Descrição
--exclude-packages	Lista separada por vírgulas de groupId:artifactId, para excluir ao resolver as dependências fornecidas em --packages para evitar conflitos de dependência.
--repositories	Lista separada por vírgulas de repositórios remotos adicionais para pesquisar as coordenadas do Maven fornecidas com --packages.
--files FILES	Lista separada por vírgulas de arquivos a serem colocados no diretório de trabalho de cada executor.
--conf PROP=VALUE	Propriedade de configuração do Spark.
--properties-file FILE	Caminho para um arquivo do qual as propriedades extras serão carregadas.
--driver-memory MEM	Memória para o driver. O padrão é de 1.024 MB.
--driver-java-options	Opções extras do Java a serem transferidas para o driver.
--driver-library-path	Entradas extras de caminhos da biblioteca a serem transferidas para o driver.
--driver-class-path	Entradas extras de caminhos de classe a serem transferidas para o driver.
--executor-memory MEM	Memória por executor. O padrão é de 1 GB.
--driver-cores NUM	Número de núcleos usados pelo driver.
--total-executor-cores NUM	Total de núcleos para todos os executores.
--executor-cores NUM	Número de núcleos usados por cada executor.

Opção	Descrição
--num-executors NUM	Número de executores para iniciar.
-hivevar <key=value>	Substituição de variável para a aplicação dos comandos do Hive, por exemplo, `-hivevar A=B`.
-hiveconf <property=value>	Valor a ser usado para a propriedade em questão.

Para um trabalho do Spark SQL, crie um arquivo start-job-run-request.json e especifique os parâmetros obrigatórios para sua execução de trabalho, como no seguinte exemplo:

```
{
  "name": "myjob",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "emr-6.7.0-latest",
  "jobDriver": {
    "sparkSqlJobDriver": {
      "entryPoint": "entryPoint_location",
       "sparkSqlParameters": "--conf spark.executor.instances=2 --conf
 spark.executor.memory=2G --conf spark.executor.cores=2 --conf spark.driver.cores=1"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.driver.memory":"2G"
         }
      }
    ],
    "monitoringConfiguration": {
      "persistentAppUI": "ENABLED",
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "my_log_group",
        "logStreamNamePrefix": "log_stream_prefix"
      },
```

```
      "s3MonitoringConfiguration": {
        "logUri": "s3://my_s3_log_location"
      }
    }
  }
}
```

Estados de execução de trabalho

Quando você envia uma execução de trabalho para uma fila de trabalhos do Amazon EMR no EKS, a execução de trabalho entra no estado `PENDING`. Em seguida, ela passa pelos estados apresentados a seguir até obter êxito (sair com o código `0`) ou falhar (sair com um código diferente de zero).

As execuções de trabalhos podem ter os seguintes estados:

- `PENDING`: o estado inicial do trabalho quando a execução de trabalho é enviada ao Amazon EMR no EKS. O trabalho está aguardando para ser enviado ao cluster virtual, e o Amazon EMR no EKS está trabalhando no envio desse trabalho.
- `SUBMITTED`: uma execução de trabalho que foi enviada com êxito ao cluster virtual. O programador do cluster tenta executar esse trabalho no cluster.
- `RUNNING`: uma execução de trabalho em execução no cluster virtual. Em aplicações do Spark, isso significa que o processo do driver do Spark está no estado `running`.
- `FAILED`: uma execução de trabalho que falhou ao ser enviada ao cluster virtual ou que foi concluída sem êxito. Consulte StateDetails e FailureReason para encontrar informações adicionais sobre essa falha de trabalho.
- `COMPLETED`: uma execução de trabalho que foi concluída com êxito.
- `CANCEL_PENDING`: uma execução de trabalho foi solicitada para cancelamento. O Amazon EMR no EKS está tentando cancelar o trabalho no cluster virtual.
- `CANCELLED`: uma execução de trabalho que foi cancelada com êxito.

Visualização de trabalhos no console do Amazon EMR

Para visualizar trabalhos no console do Amazon EMR, execute as etapas a seguir.

1. No menu à esquerda do console do Amazon EMR, em Amazon EMR no EKS, escolha Clusters virtuais.

2. Na lista de clusters virtuais, selecione o cluster virtual do qual deseja visualizar os trabalhos.
3. Na tabela Execuções de trabalhos, selecione Visualizar logs para visualizar os detalhes de uma execução de trabalho.

> **Note**
>
> O suporte para a experiência de um clique está habilitado por padrão. Ele pode ser desativado ao configurar `persistentAppUI` para `DISABLED` na `monitoringConfiguration` durante o envio do trabalho. Para obter mais informações, consulte View Persistent Application User Interfaces.

Erros comuns ao executar trabalhos

Os erros a seguir podem ocorrer ao executar a API `StartJobRun`.

Mensagem de erro	Condição de erro	Próxima etapa recomendada
error: argument --*argument* is required	Os parâmetros obrigatórios estão ausentes.	Adicione os argumentos ausentes à solicitação de API.
An error occurred (AccessDeniedException) when calling the StartJobRun operation: User: *ARN* is not authorized to perform: emr-containers:StartJobRun	O perfil de execução está ausente.	Consulte Uso de perfis de execução de trabalho com o Amazon EMR no EKS.
An error occurred (AccessDeniedException) when calling the StartJobRun operation: User: *ARN* is not authorized to perform: emr-containers:StartJobRun	O chamador não tem permissão para acessar o perfil de execução [formato válido ou inválido] por meio de chaves de condição.	Consulte Uso de perfis de execução de trabalho com o Amazon EMR no EKS.

Mensagem de erro	Condição de erro	Próxima etapa recomendada
An error occurred (AccessDeniedException) when calling the StartJobRun operation: User: *ARN* is not authorized to perform: emr-containers:StartJobRun	O emissor de trabalho e o ARN do perfil de execução são de contas diferentes.	Certifique-se de que o ARN do emissor de trabalho e do perfil de execução sejam da mesma conta da AWS.
1 validation error detected: Value *Role* at “executionRoleArn” failed to satisfy the ARN regular expression pattern: ^arn:(aws[a-zA-Z0-9-]*):iam::(\d{12})?:(role((\u002F)|(\u002F[\u0021-\u007F]+\u002F))[\w+=,.@-]+)	O chamador tem permissões para acessar o perfil de execução por meio de chaves de condição, mas o perfil não atende às restrições do formato de ARN.	Forneça o perfil de execução seguindo o formato de ARN. Consulte Uso de perfis de execução de trabalho com o Amazon EMR no EKS.
An error occurred (ResourceNotFoundException) when calling the StartJobRun operation: Virtual cluster *Virtual Cluster ID* doesn't exist.	O ID do cluster virtual não foi encontrado.	Forneça um ID do cluster virtual registrado no Amazon EMR no EKS.
An error occurred (ValidationException) when calling the StartJobRun operation: Virtual cluster state *state* is not valid to create resource JobRun.	O cluster virtual não está pronto para executar o trabalho.	Consulte Estados de um cluster virtual.
An error occurred (ResourceNotFoundException) when calling the StartJobRun operation: Release *RELEASE* doesn't exist.	A versão especificada no envio do trabalho está incorreta.	Consulte Versões do Amazon EMR no EKS.

Mensagem de erro	Condição de erro	Próxima etapa recomendada
An error occurred (AccessDeniedException) when calling the StartJobRun operation: User: *ARN* is not authorized to perform: emr-containers:StartJobRun on resource: *ARN* with an explicit deny. An error occurred (AccessDeniedException) when calling the StartJobRun operation: User: *ARN* is not authorized to perform: emr-containers:StartJobRun on resource: *ARN*	O usuário não está autorizado a chamar StartJobRun.	Consulte Uso de perfis de execução de trabalho com o Amazon EMR no EKS.
An error occurred (ValidationException) when calling the StartJobRun operation: configurationOverrides.monitoringConfiguration.s3MonitoringConfiguration.logUri failed to satisfy constraint : %s	A sintaxe do URI do caminho do S3 não é válida.	O logUri deve estar no formato s3://...

Os erros a seguir podem ocorrer ao executar a API `DescribeJobRun` antes da execução de trabalho.

Mensagem de erro	Condição de erro	Próxima etapa recomendada
stateDetails: JobRun submission failed. Classification *classification* not supported.	Os parâmetros em StartJobRun não são válidos.	Consulte Versões do Amazon EMR no EKS.

Mensagem de erro	Condição de erro	Próxima etapa recomendada
failureReason: VALIDATIO N_ERROR state: FAILED.		
stateDetails: Cluster *EKS Cluster ID* does not exist. failureReason: CLUSTER_U NAVAILABLE state: FAILED	O cluster do EKS não está disponível.	Verifique se o cluster do EKS existe e tem as permissõe s corretas. Para obter mais informações, consulte Configuração do Amazon EMR no EKS.
stateDetails: Cluster *EKS Cluster ID* does not have sufficient permissions. failureReason: CLUSTER_U NAVAILABLE state: FAILED	O Amazon EMR não tem permissões para acessar o cluster do EKS.	Verifique se as permissõe s estão configuradas para o Amazon EMR no namespace registrado. Para obter mais informações, consulte Configuração do Amazon EMR no EKS.
stateDetails: Cluster *EKS Cluster ID* is currently not reachable. failureReason: CLUSTER_U NAVAILABLE state: FAILED	O cluster do EKS não está acessível.	Verifique se o cluster do EKS existe e tem as permissõe s corretas. Para obter mais informações, consulte Configuração do Amazon EMR no EKS.
stateDetails: JobRun submission failed due to an internal error. failureReason: INTERNAL_ ERROR state: FAILED	Ocorreu um erro interno com o cluster do EKS.	N/D

Mensagem de erro	Condição de erro	Próxima etapa recomendada
stateDetails: Cluster *EKS Cluster ID* does not have sufficient resources. failureReason: USER_ERROR state: FAILED	Não há recursos suficientes no cluster do EKS para executar o trabalho.	Adicione mais capacidade ao grupo de nós do EKS ou configure o Autoscaler do EKS. Para obter mais informações, consulte Autoscaler do cluster.

Os erros a seguir podem ocorrer ao executar a API `DescribeJobRun` depois da execução de trabalho.

Mensagem de erro	Condição de erro	Próxima etapa recomendada
stateDetails: Trouble monitoring your JobRun. Cluster *EKS Cluster ID* does not exist. failureReason: CLUSTER_UNAVAILABLE state: FAILED	O cluster do EKS não existe.	Verifique se o cluster do EKS existe e tem as permissões corretas. Para obter mais informações, consulte Configuração do Amazon EMR no EKS.
stateDetails: Trouble monitoring your JobRun. Cluster *EKS Cluster ID* does not have sufficient permissions. failureReason: CLUSTER_UNAVAILABLE state: FAILED	O Amazon EMR não tem permissões para acessar o cluster do EKS.	Verifique se as permissões estão configuradas para o Amazon EMR no namespace registrado. Para obter mais informações, consulte Configuração do Amazon EMR no EKS.

Mensagem de erro	Condição de erro	Próxima etapa recomendada
stateDetails: Trouble monitoring your JobRun. Cluster *EKS Cluster ID* is currently not reachable. failureReason: CLUSTER_UNAVAILABLE state: FAILED	O cluster do EKS não está acessível.	Verifique se o cluster do EKS existe e tem as permissões corretas. Para obter mais informações, consulte Configuração do Amazon EMR no EKS.
stateDetails: Trouble monitoring your JobRun due to an internal error failureReason: INTERNAL_ERROR state: FAILED	Ocorreu um erro interno que está impedindo o monitoramento do JobRun.	N/D

O erro apresentado a seguir pode ocorrer quando um trabalho não pode ser iniciado e aguarda no estado SUBMITTED por 15 minutos. Isso pode ser causado pela falta de recursos do cluster.

Mensagem de erro	Condição de erro	Próxima etapa recomendada
cluster timeout	O trabalho está no estado SUBMITTED há 15 minutos ou mais.	Você pode substituir a configuração padrão de 15 minutos para este parâmetro com a substituição de configuração mostrada abaixo.

Use a configuração a seguir para alterar a configuração de tempo limite do cluster para 30 minutos. Observe que você fornece o novo valor para `job-start-timeout` em segundos:

```
{
"configurationOverrides": {
```

```
  "applicationConfiguration": [{
      "classification": "emr-containers-defaults",
      "properties": {
          "job-start-timeout":"1800"
      }
  }]
}
```

Uso da classificação de envio de trabalho

Visão geral

A solicitação `StartJobRun` do Amazon EMR no EKS cria um pod de envio de trabalho (também conhecido como pod job-runner) para gerar o driver do Spark. É possível configurar seletores de nós para seu pod de envio de trabalho com a classificação `emr-job-submitter`.

A seguinte configuração está disponível na classificação `emr-job-submitter`:

`jobsubmitter.node.selector.[labelKey]`

> É adicionada ao seletor de nó do pod de envio de trabalho, com a chave *`labelKey`* e o valor como o valor de configuração para a configuração. Por exemplo, você pode definir `jobsubmitter.node.selector.identifier` como `myIdentifier` e o pod de envio de trabalho terá um seletor de nó com um valor de identificador de chave `myIdentifier`. Para adicionar diversas chaves seletoras de nós, defina diversas configurações com esse prefixo.

Como prática recomendada, recomendamos que os pods de envio de trabalho estejam estabelecidos em nós em instâncias sob demanda, em vez de em instâncias spot. Um trabalho falhará se o pod de envio de trabalho estiver sujeito a interrupções da instância spot. Você também pode estabelecer o pod de envio de trabalho em uma única zona de disponibilidade ou usar quaisquer rótulos do Kubernetes aplicados aos nós.

Exemplos de classificação de envio de trabalho

Nesta seção

- Solicitação StartJobRun com nó sob demanda estabelecido para o pod de envio de trabalho
- Solicitação StartJobRun com nó em uma única zona de disponibilidade estabelecido para o pod de envio de trabalho

- Solicitação StartJobRun com tipo de instância do Amazon EC2 e uma única zona de disponibilidade estabelecidos para o pod de envio de trabalho

Solicitação **StartJobRun** com nó sob demanda estabelecido para o pod de envio de trabalho

```
cat >spark-python-in-s3-nodeselector-job-submitter.json << EOF
{
  "name": "spark-python-in-s3-nodeselector",
  "virtualClusterId": "virtual-cluster-id",
  "executionRoleArn": "execution-role-arn",
  "releaseLabel": "emr-6.11.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "s3://S3-prefix/trip-count.py",
       "sparkSubmitParameters": "--conf spark.driver.cores=5  --conf
 spark.executor.memory=20G --conf spark.driver.memory=15G --conf
 spark.executor.cores=6"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.dynamicAllocation.enabled":"false"
         }
      },
      {
        "classification": "emr-job-submitter",
        "properties": {
            "jobsubmitter.node.selector.eks.amazonaws.com/capacityType": "ON_DEMAND"
        }
      }
    ],
    "monitoringConfiguration": {
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "/emr-containers/jobs",
        "logStreamNamePrefix": "demo"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://joblogs"
```

```
      }
    }
  }
}
EOF
aws emr-containers start-job-run --cli-input-json file:///spark-python-in-s3-
nodeselector-job-submitter.json
```

Solicitação **StartJobRun** com nó em uma única zona de disponibilidade estabelecido para o pod de envio de trabalho

```
cat >spark-python-in-s3-nodeselector-job-submitter-az.json << EOF
{
  "name": "spark-python-in-s3-nodeselector",
  "virtualClusterId": "virtual-cluster-id",
  "executionRoleArn": "execution-role-arn",
  "releaseLabel": "emr-6.11.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "s3://S3-prefix/trip-count.py",
       "sparkSubmitParameters": "--conf spark.driver.cores=5  --conf
 spark.executor.memory=20G --conf spark.driver.memory=15G --conf
 spark.executor.cores=6"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.dynamicAllocation.enabled":"false"
         }
      },
      {
        "classification": "emr-job-submitter",
        "properties": {
            "jobsubmitter.node.selector.topology.kubernetes.io/zone": "Availability
 Zone"
        }
      }
    ],
    "monitoringConfiguration": {
      "cloudWatchMonitoringConfiguration": {
```

```
        "logGroupName": "/emr-containers/jobs",
        "logStreamNamePrefix": "demo"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://joblogs"
      }
    }
  }
}
EOF
aws emr-containers start-job-run --cli-input-json file:///spark-python-in-s3-
nodeselector-job-submitter-az.json
```

Solicitação **`StartJobRun`** com tipo de instância do Amazon EC2 e uma única zona de disponibilidade estabelecidos para o pod de envio de trabalho

```
{
  "name": "spark-python-in-s3-nodeselector",
  "virtualClusterId": "virtual-cluster-id",
  "executionRoleArn": "execution-role-arn",
  "releaseLabel": "emr-6.11.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "s3://S3-prefix/trip-count.py",
       "sparkSubmitParameters": "--conf spark.driver.cores=5  --conf
 spark.kubernetes.pyspark.pythonVersion=3 --conf spark.executor.memory=20G
 --conf spark.driver.memory=15G --conf spark.executor.cores=6 --conf
 spark.sql.shuffle.partitions=1000"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.dynamicAllocation.enabled":"false",
         }
      },
      {
        "classification": "emr-job-submitter",
        "properties": {
            "jobsubmitter.node.selector.topology.kubernetes.io/zone": "Availability
 Zone",
```

```
            "jobsubmitter.node.selector.node.kubernetes.io/instance-type":"m5.4xlarge"
        }
      }
    ],
    "monitoringConfiguration": {
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "/emr-containers/jobs",
        "logStreamNamePrefix": "demo"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://joblogs"
      }
    }
  }
}
```

Uso de modelos de trabalho

Um modelo de trabalho armazena valores que podem ser compartilhados entre invocações da API `StartJobRun` ao iniciar uma execução de trabalho. Ele oferece suporte a dois casos de uso:

- Para evitar valores de solicitação da API `StartJobRun` repetitivos e recorrentes.
- Para impor uma regra, determinados valores devem ser fornecidos por meio de solicitações da API `StartJobRun`.

Os modelos de trabalho possibilitam definir um modelo reutilizável para execuções de trabalho com a finalidade de aplicar personalizações adicionais, por exemplo:

- Configuração da capacidade de computação dos executores e dos drivers.
- Definição de propriedades de segurança e de governança, como perfis do IAM.
- Personalização de uma imagem do Docker para usar em diversas aplicações e pipelines de dados.

Criação e uso de um modelo de trabalho para iniciar uma execução de trabalho

Esta seção descreve a criação de um modelo de trabalho e o uso desse modelo para iniciar uma execução de trabalho com a AWS Command Line Interface (AWS CLI).

Criar um modelo de trabalho

1. Crie um arquivo `create-job-template-request.json` e especifique os parâmetros obrigatórios para seu modelo de trabalho, conforme mostrado no arquivo JSON de exemplo a seguir. Para obter informações sobre todos os parâmetros disponíveis, consulte a API CreateJobTemplate.

   A maioria dos valores obrigatórios para a API `StartJobRun` também são obrigatórios para `jobTemplateData`. Se você desejar usar espaços reservados para quaisquer parâmetros e fornecer valores ao invocar StartJobRun usando um modelo de trabalho, consulte a próxima seção sobre os parâmetros de modelos de trabalhos.

```
{
   "name": "mytemplate",
   "jobTemplateData": {
        "executionRoleArn": "iam_role_arn_for_job_execution",
        "releaseLabel": "emr-6.7.0-latest",
        "jobDriver": {
            "sparkSubmitJobDriver": {
                "entryPoint": "entryPoint_location",
                "entryPointArguments": [ "argument1","argument2",...],
                "sparkSubmitParameters": "--class <main_class> --conf
 spark.executor.instances=2 --conf spark.executor.memory=2G --conf
 spark.executor.cores=2 --conf spark.driver.cores=1"
            }
       },
       "configurationOverrides": {
            "applicationConfiguration": [
                {
                    "classification": "spark-defaults",
                    "properties": {
                         "spark.driver.memory":"2G"
                    }
                }
            ],
            "monitoringConfiguration": {
                "persistentAppUI": "ENABLED",
                "cloudWatchMonitoringConfiguration": {
                    "logGroupName": "my_log_group",
                    "logStreamNamePrefix": "log_stream_prefix"
                },
                "s3MonitoringConfiguration": {
```

```
                    "logUri": "s3://my_s3_log_location/"
                }
            }
        }
    }
}
```

2. Use o comando `create-job-template` com um caminho para o arquivo `create-job-template-request.json` armazenado localmente.

```
aws emr-containers create-job-template \
--cli-input-json file://./create-job-template-request.json
```

Iniciar uma execução de trabalho usando um modelo de trabalho

Forneça o ID do cluster virtual, o ID do modelo de trabalho e o nome do trabalho no comando `StartJobRun`, conforme mostrado no exemplo a seguir.

```
aws emr-containers start-job-run \
--virtual-cluster-id 123456 \
--name myjob \
--job-template-id 1234abcd
```

Definição de parâmetros de modelos de trabalhos

Os parâmetros de modelos de trabalhos permitem especificar variáveis no modelo de trabalho. Os valores para essas variáveis de parâmetro precisarão ser especificados ao iniciar uma execução de trabalho usando esse modelo de trabalho. Os parâmetros do modelo de trabalho são especificados no formato `${parameterName}`. Você pode optar por especificar qualquer valor em um campo `jobTemplateData` como um parâmetro do modelo de trabalho. Para cada uma das variáveis de parâmetro do modelo de trabalho, especifique seu tipo de dados (`STRING` ou `NUMBER`) e, opcionalmente, um valor padrão. O exemplo abaixo mostra como você pode especificar os parâmetros do modelo de trabalho para os valores de local do ponto de entrada, classe principal e local do log do S3.

Para especificar o local do ponto de entrada, a classe principal e o local do log do Amazon S3 como parâmetros do modelo de trabalho

1. Crie um arquivo `create-job-template-request.json` e especifique os parâmetros obrigatórios para seu modelo de trabalho, conforme mostrado no arquivo JSON de exemplo a seguir. Para obter mais informações sobre os parâmetros, consulte a API CreateJobTemplate.

```
{
   "name": "mytemplate",
   "jobTemplateData": {
        "executionRoleArn": "iam_role_arn_for_job_execution",
        "releaseLabel": "emr-6.7.0-latest",
        "jobDriver": {
            "sparkSubmitJobDriver": {
                "entryPoint": "${EntryPointLocation}",
                "entryPointArguments": [ "argument1","argument2",...],
                "sparkSubmitParameters": "--class ${MainClass} --conf
 spark.executor.instances=2 --conf spark.executor.memory=2G --conf
 spark.executor.cores=2 --conf spark.driver.cores=1"
            }
        },
        "configurationOverrides": {
            "applicationConfiguration": [
                {
                    "classification": "spark-defaults",
                    "properties": {
                         "spark.driver.memory":"2G"
                    }
                }
            ],
            "monitoringConfiguration": {
                "persistentAppUI": "ENABLED",
                "cloudWatchMonitoringConfiguration": {
                    "logGroupName": "my_log_group",
                    "logStreamNamePrefix": "log_stream_prefix"
                },
                "s3MonitoringConfiguration": {
                    "logUri": "${LogS3BucketUri}"
                }
            }
        },
        "parameterConfiguration": {
            "EntryPointLocation": {
                "type": "STRING"
            },
            "MainClass": {
```

```
                "type": "STRING",
                "defaultValue":"Main"
            },
            "LogS3BucketUri": {
                "type": "STRING",
                "defaultValue":"s3://my_s3_log_location/"
            }
        }
    }
}
```

2. Use o comando `create-job-template` com um caminho para o arquivo `create-job-template-request.json` armazenado localmente ou no Amazon S3.

```
aws emr-containers create-job-template \
--cli-input-json file://./create-job-template-request.json
```

Para iniciar uma execução de trabalho usando um modelo de trabalho com parâmetros de modelos de trabalhos

Para iniciar uma execução de trabalho com um modelo de trabalho contendo parâmetros de modelos de trabalhos, especifique o ID do modelo de trabalho, bem como os valores dos parâmetros do modelo de trabalho na solicitação da API `StartJobRun`, conforme mostrado abaixo.

```
aws emr-containers start-job-run \
--virtual-cluster-id 123456 \
--name myjob \
--job-template-id 1234abcd \
--job-template-parameters '{"EntryPointLocation": "entry_point_location","MainClass":
 "ExampleMainClass","LogS3BucketUri": "s3://example_s3_bucket/"}'
```

Controle do acesso aos modelos de trabalhos

A política `StartJobRun` permite impor que um usuário ou um perfil possa somente executar trabalhos usando modelos de trabalhos especificados e não possa executar operações `StartJobRun` sem usar os modelos de trabalhos especificados. Para conseguir isso, primeiro, certifique-se de conceder ao usuário ou ao perfil uma permissão de leitura para os modelos de trabalhos especificados, conforme mostrado abaixo.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "emr-containers:DescribeJobTemplate",
            "Resource": [
                "job_template_1_arn",
                "job_template_2_arn",
                ...
            ]
        }
    ]
}
```

Para garantir que um usuário ou que um perfil seja capaz de invocar a operação `StartJobRun` somente ao usar modelos de trabalhos especificados, você pode atribuir a permissão de política `StartJobRun` apresentada a seguir para um determinado usuário ou perfil.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "emr-containers:StartJobRun",
            "Resource": [
                "virtual_cluster_arn",
            ],
            "Condition": [
                "StringEquals": {
                    "emr-containers:JobTemplateArn": [
                        "job_template_1_arn",
                        "job_template_2_arn",
                        ...
                    ]
                 }
                ]
            }
        }
    ]
}
```

Se o modelo de trabalho especificar um parâmetro de modelo de trabalho dentro do campo ARN do perfil de execução, o usuário poderá fornecer um valor para esse parâmetro e, assim, invocar `StartJobRun` usando um perfil de execução arbitrário. Para restringir os perfis de execução que o usuário pode fornecer, consulte Controle do acesso ao perfil de execução em Uso de perfis de execução de trabalho com o Amazon EMR no EKS.

Se nenhuma condição for especificada na política de ação `StartJobRun` acima para um determinado usuário ou perfil, o usuário ou o perfil terá permissão para invocar a ação `StartJobRun` no cluster virtual especificado usando um modelo de trabalho arbitrário ao qual ele tenha acesso de leitura ou usando um perfil de execução arbitrário.

Uso de modelos de pod

A partir das versões 5.33.0 ou 6.3.0 do Amazon EMR, o Amazon EMR no EKS oferece suporte ao recurso de modelo de pod do Spark. Um pod corresponde a um grupo de um ou mais contêineres, com recursos de rede e armazenamento compartilhados, e uma especificação de como executar os contêineres. Os modelos de pod são especificações que determinam como ocorrerá a execução de cada pod. É possível usar arquivos de modelo de pod para definir as configurações de pods de drivers ou de executores para as quais as configurações do Spark não oferecem suporte. Para obter mais informações sobre o recurso de modelo de pod do Spark, consulte Pod Template.

> **Note**
>
> O recurso de modelo de pod funciona somente com pods de drivers e de executores. Não é possível configurar pods de controladores de trabalhos usando o modelo de pod.

Cenários comuns

Você pode definir como executar trabalhos do Spark em clusters do EKS compartilhados ao usar modelos de pod com o Amazon EMR no EKS, e economizar custos e aprimorar a utilização e a performance dos recursos.

- Para reduzir custos, você pode programar tarefas do driver do Spark para a execução em instâncias sob demanda do Amazon EC2 enquanto programa tarefas do executor do Spark para a execução em instâncias spot do Amazon EC2.
- Para aumentar a utilização de recursos, você pode oferecer suporte a diversas equipes que executam as workloads no mesmo cluster do EKS. Cada equipe receberá um grupo de nós do

Amazon EC2 designado para executar as workloads. Você pode usar os modelos de pod para aplicar uma tolerância correspondente à workload.

- Para aprimorar o monitoramento, você pode executar um contêiner de registro em log separado para encaminhar os logs para a aplicação de monitoramento existente.

Por exemplo, o arquivo de modelo de pod a seguir demonstra um cenário de uso comum.

```
apiVersion: v1
kind: Pod
spec:
  volumes:
    - name: source-data-volume
      emptyDir: {}
    - name: metrics-files-volume
      emptyDir: {}
  nodeSelector:
    eks.amazonaws.com/nodegroup: emr-containers-nodegroup
  containers:
  - name: spark-kubernetes-driver # This will be interpreted as driver Spark main
 container
    env:
      - name: RANDOM
        value: "random"
    volumeMounts:
      - name: shared-volume
        mountPath: /var/data
      - name: metrics-files-volume
        mountPath: /var/metrics/data
  - name: custom-side-car-container # Sidecar container
    image: <side_car_container_image>
    env:
      - name: RANDOM_SIDECAR
        value: random
    volumeMounts:
      - name: metrics-files-volume
        mountPath: /var/metrics/data
    command:
      - /bin/sh
      - '-c'
      -  <command-to-upload-metrics-files>
  initContainers:
  - name: spark-init-container-driver # Init container
```

```
    image: <spark-pre-step-image>
    volumeMounts:
      - name: source-data-volume # Use EMR predefined volumes
        mountPath: /var/data
    command:
      - /bin/sh
      - '-c'
      -  <command-to-download-dependency-jars>
```

O modelo de pod conclui as seguintes tarefas:

- Adiciona um novo contêiner Init que é executado antes do início do contêiner principal do Spark. O contêiner Init compartilha o volume EmptyDir chamado `source-data-volume` com o contêiner principal do Spark. Você pode fazer com que seu contêiner Init execute as etapas de inicialização, como fazer download de dependências ou gerar dados de entrada. Em seguida, o contêiner principal do Spark consumirá os dados.
- Adiciona outro contêiner sidecar que é executado junto com o contêiner principal do Spark. Os dois contêineres estão compartilhando outro volume `EmptyDir` chamado `metrics-files-volume`. Seu trabalho do Spark pode gerar métricas, como as métricas do Prometheus. Em seguida, o trabalho do Spark pode colocar as métricas em um arquivo e fazer com que o contêiner sidecar carregue os arquivos em seu próprio sistema de BI para análise futura.
- Adiciona uma nova variável de ambiente ao contêiner principal do Spark. Você pode fazer com que seu trabalho consuma a variável de ambiente.
- Defina um seletor de nó, para que o pod seja programado somente no grupo de nós `emr-containers-nodegroup`. Isso ajuda a isolar os recursos de computação entre trabalhos e equipes.

Habilitação de modelos de pod com o Amazon EMR no EKS

Para habilitar o recurso de modelo de pod com o Amazon EMR no EKS, configure as propriedades do Spark `spark.kubernetes.driver.podTemplateFile` e `spark.kubernetes.executor.podTemplateFile` para direcionar aos arquivos de modelo de pod no Amazon S3. O Spark fará download do arquivo de modelo de pod e o usará para estruturar pods de drivers e de executores.

> **Note**
>
> O Spark usa o perfil de execução de trabalho para carregar o modelo de pod, portanto, o perfil de execução de trabalho deve ter permissões para acessar o Amazon S3 com a finalidade de carregar os modelos de pod. Para obter mais informações, consulte Criação de um perfil de execução de trabalho.

Você pode usar `SparkSubmitParameters` para especificar o caminho do Amazon S3 para o modelo de pod, como demonstra o arquivo JSON de execução de trabalho a seguir.

```
{
  "name": "myjob",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "release_label",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "entryPoint_location",
      "entryPointArguments": ["argument1", "argument2", ...],
       "sparkSubmitParameters": "--class <main_class> \
        --conf
 spark.kubernetes.driver.podTemplateFile=s3://path_to_driver_pod_template \
        --conf
 spark.kubernetes.executor.podTemplateFile=s3://path_to_executor_pod_template \
        --conf spark.executor.instances=2 \
        --conf spark.executor.memory=2G \
        --conf spark.executor.cores=2 \
        --conf spark.driver.cores=1"
    }
  }
}
```

Como alternativa, você pode usar `configurationOverrides` para especificar o caminho do Amazon S3 para o modelo de pod, como demonstra o arquivo JSON de execução de trabalho a seguir.

```
{
  "name": "myjob",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
```

```
  "releaseLabel": "release_label",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "entryPoint_location",
      "entryPointArguments": ["argument1", "argument2", ...],
       "sparkSubmitParameters": "--class <main_class> \
        --conf spark.executor.instances=2 \
        --conf spark.executor.memory=2G \
        --conf spark.executor.cores=2 \
        --conf spark.driver.cores=1"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.driver.memory":"2G",
          "spark.kubernetes.driver.podTemplateFile":"s3://path_to_driver_pod_template",

 "spark.kubernetes.executor.podTemplateFile":"s3://path_to_executor_pod_template"
         }
      }
    ]
  }
}
```

> **Note**
>
> 1. Você precisa seguir as diretrizes de segurança ao usar o recurso de modelo de pod com o Amazon EMR no EKS, por exemplo, como isolar códigos de aplicações não confiáveis. Para obter mais informações, consulte Práticas recomendadas de segurança para o Amazon EMR no EKS.
> 2. Não é possível alterar os nomes dos contêineres principais do Spark ao usar `spark.kubernetes.driver.podTemplateContainerName` e `spark.kubernetes.executor.podTemplateContainerName`, porque esses nomes são codificados como `spark-kubernetes-driver` e `spark-kubernetes-executors`. Se desejar personalizar o contêiner principal do Spark, deverá especificar o contêiner em um modelo de pod com esses nomes codificados.

Campos do modelo de pod

Considere as restrições para campos apresentadas a seguir ao configurar um modelo de pod com o Amazon EMR no EKS.

- O Amazon EMR no EKS permite somente os campos a seguir em um modelo de pod para habilitar a programação adequada de trabalhos.

  Estes são os campos permitidos em nível de pod:

  - `apiVersion`
  - `kind`
  - `metadata`
  - `spec.activeDeadlineSeconds`
  - `spec.affinity`
  - `spec.containers`
  - `spec.enableServiceLinks`
  - `spec.ephemeralContainers`
  - `spec.hostAliases`
  - `spec.hostname`
  - `spec.imagePullSecrets`
  - `spec.initContainers`
  - `spec.nodeName`
  - `spec.nodeSelector`
  - `spec.overhead`
  - `spec.preemptionPolicy`
  - `spec.priority`
  - `spec.priorityClassName`
  - `spec.readinessGates`
  - `spec.runtimeClassName`
  - `spec.schedulerName`
  - `spec.subdomain`
  - `spec.terminationGracePeriodSeconds`

- spec.tolerations
- spec.topologySpreadConstraints
- spec.volumes

Estes são os campos permitidos em nível de contêiner principal do Spark:

- env
- envFrom
- name
- lifecycle
- livenessProbe
- readinessProbe
- resources
- startupProbe
- stdin
- stdinOnce
- terminationMessagePath
- terminationMessagePolicy
- tty
- volumeDevices
- volumeMounts
- workingDir

Quando você usa qualquer campo não permitido no modelo de pod, o Spark gera uma exceção e o trabalho falha. O exemplo a seguir mostra uma mensagem de erro no log do controlador do Spark devido ao uso de campos não permitidos.

```
Executor pod template validation failed.
Field container.command in Spark main container not allowed but specified.
```

- O Amazon EMR no EKS define previamente os parâmetros apresentados a seguir em um modelo de pod. Os campos especificados em um modelo de pod não devem se sobrepor a esses campos.

  Estes são os nomes de volumes definidos previamente:

  - emr-container-communicate

- `config-volume`
- `emr-container-application-log-dir`
- `emr-container-event-log-dir`
- `temp-data-dir`
- `mnt-dir`
- `home-dir`
- `emr-container-s3`

Estas são as montagens de volume definidas previamente que se aplicam somente ao contêiner principal do Spark:

- Nome: `emr-container-communicate`; MountPath: `/var/log/fluentd`
- Nome: `emr-container-application-log-dir`; MountPath: `/var/log/spark/user`
- Nome: `emr-container-event-log-dir`; MountPath: `/var/log/spark/apps`
- Nome: `mnt-dir`; MountPath: `/mnt`
- Nome: `temp-data-dir`; MountPath: `/tmp`
- Nome: `home-dir`; MountPath: `/home/hadoop`

Estas são as variáveis de ambiente definidas previamente que se aplicam somente ao contêiner principal do Spark:

- `SPARK_CONTAINER_ID`
- `K8S_SPARK_LOG_URL_STDERR`
- `K8S_SPARK_LOG_URL_STDOUT`
- `SIDECAR_SIGNAL_FILE`

> **Note**
>
> Você ainda pode usar esses volumes definidos previamente e montá-los em seus contêineres sidecar adicionais. Por exemplo, é possível usar `emr-container-application-log-dir` e montá-lo em seu próprio contêiner sidecar definido no modelo de pod.

Se os campos especificados entrarem em conflito com qualquer um dos campos definidos previamente no modelo de pod, o Spark gerará uma exceção e o trabalho falhará. O exemplo a

seguir mostra uma mensagem de erro no log da aplicação do Spark devido a conflitos com os campos definidos previamente.

```
Defined volume mount path on main container must not overlap with reserved mount
 paths: [<reserved-paths>]
```

Considerações sobre contêineres sidecar

O Amazon EMR controla o ciclo de vida dos pods provisionados pelo Amazon EMR no EKS. Os contêineres sidecar devem seguir um ciclo de vida semelhante ao do contêiner principal do Spark. Se você injetar contêineres sidecar adicionais em seus pods, recomendamos a integração com o gerenciamento do ciclo de vida do pod definido pelo Amazon EMR para que o contêiner sidecar possa ser interrompido sozinho quando o contêiner principal do Spark sair.

Para reduzir custos, recomendamos implementar um processo que impeça que pods de drivers com contêineres sidecar continuem em execução após a conclusão do seu trabalho. O driver do Spark exclui os pods de executores quando o executor é encerrado. No entanto, quando um programa de driver é concluído, os contêineres sidecar adicionais continuam em execução. O pod é faturado até que o Amazon EMR no EKS limpe o pod do driver. Geralmente, isso ocorre menos de um minuto após a conclusão do contêiner principal do driver do Spark. Para reduzir custos, você pode integrar os contêineres sidecar adicionais ao mecanismo de gerenciamento do ciclo de vida que o Amazon EMR no EKS define para os pods de drivers e de executores, conforme descrito na seção a seguir.

O contêiner principal do Spark nos pods de drivers e de executores envia `heartbeat` para um arquivo `/var/log/fluentd/main-container-terminated` a cada dois segundos. Ao adicionar a montagem de volume `emr-container-communicate` definida previamente do Amazon EMR ao contêiner sidecar, você pode definir um subprocesso do contêiner sidecar para monitorar periodicamente o horário da última modificação desse arquivo. O subprocesso é interrompido sozinho caso descubra que o contêiner principal do Spark interrompeu a `heartbeat` por um período mais longo.

O exemplo apresentado a seguir demonstra um subprocesso que rastreia o arquivo de pulsação e é interrompido sozinho. Substitua *`your_volume_mount`* pelo caminho no qual você monta o volume definido previamente. O script é empacotado dentro da imagem usada pelo contêiner sidecar. Em um arquivo de modelo de pod, você pode especificar um contêiner sidecar com os seguintes comandos: `sub_process_script.sh` e `main_command`.

```
MOUNT_PATH="your_volume_mount"
```

```
FILE_TO_WATCH="$MOUNT_PATH/main-container-terminated"
INITIAL_HEARTBEAT_TIMEOUT_THRESHOLD=60
HEARTBEAT_TIMEOUT_THRESHOLD=15
SLEEP_DURATION=10

function terminate_main_process() {
  # Stop main process
}

# Waiting for the first heartbeat sent by Spark main container
echo "Waiting for file $FILE_TO_WATCH to appear..."
start_wait=$(date +%s)
while ! [[ -f "$FILE_TO_WATCH" ]]; do
    elapsed_wait=$(expr $(date +%s) - $start_wait)
    if [ "$elapsed_wait" -gt "$INITIAL_HEARTBEAT_TIMEOUT_THRESHOLD" ]; then
        echo "File $FILE_TO_WATCH not found after $INITIAL_HEARTBEAT_TIMEOUT_THRESHOLD
 seconds; aborting"
        terminate_main_process
        exit 1
    fi
    sleep $SLEEP_DURATION;
done;
echo "Found file $FILE_TO_WATCH; watching for heartbeats..."

while [[ -f "$FILE_TO_WATCH" ]]; do
    LAST_HEARTBEAT=$(stat -c %Y $FILE_TO_WATCH)
    ELAPSED_TIME_SINCE_AFTER_HEARTBEAT=$(expr $(date +%s) - $LAST_HEARTBEAT)
    if [ "$ELAPSED_TIME_SINCE_AFTER_HEARTBEAT" -gt "$HEARTBEAT_TIMEOUT_THRESHOLD" ];
 then
        echo "Last heartbeat to file $FILE_TO_WATCH was more than
 $HEARTBEAT_TIMEOUT_THRESHOLD seconds ago at $LAST_HEARTBEAT; terminating"
        terminate_main_process
        exit 0
    fi
    sleep $SLEEP_DURATION;
done;
echo "Outside of loop, main-container-terminated file no longer exists"

# The file will be deleted once the fluentd container is terminated

echo "The file $FILE_TO_WATCH doesn't exist any more;"
terminate_main_process
exit 0
```

Uso de políticas de repetição de trabalho

Nas versões 6.9.0 e posteriores do Amazon EMR no EKS, você pode definir uma política de repetição para as execuções de trabalhos. As políticas de repetição fazem com que um pod do driver de trabalho seja reiniciado automaticamente se falhar ou for excluído. Isso torna os trabalhos de transmissão do Spark de execução prolongada mais resistentes a falhas.

Definição de uma política de repetição para um trabalho

Para configurar uma política de repetição, forneça um campo `RetryPolicyConfiguration` usando a API StartJobRun. Um exemplo de `retryPolicyConfiguration` é mostrado aqui:

```
aws emr-containers start-job-run \
--virtual-cluster-id cluster_id \
--name sample-job-name \
--execution-role-arn execution-role-arn \
--release-label emr-6.9.0-latest \
--job-driver '{
  "sparkSubmitJobDriver": {
    "entryPoint": "local:///usr/lib/spark/examples/src/main/python/pi.py",
    "entryPointArguments": [ "2" ],
    "sparkSubmitParameters": "--conf spark.executor.instances=2 --conf
 spark.executor.memory=2G --conf spark.executor.cores=2 --conf spark.driver.cores=1"
  }
}' \
--retry-policy-configuration '{
    "maxAttempts": 5
  }' \
--configuration-overrides '{
  "monitoringConfiguration": {
    "cloudWatchMonitoringConfiguration": {
      "logGroupName": "my_log_group_name",
      "logStreamNamePrefix": "my_log_stream_prefix"
    },
    "s3MonitoringConfiguration": {
       "logUri": "s3://DOC-EXAMPLE-BUCKET-LOGGING"
    }
  }
}'
```

> **Note**
>
> `retryPolicyConfiguration` está disponível somente a partir da versão 1.27.68 da AWS CLI. Para atualizar a AWS CLI para a versão mais recente, consulte Instalar ou atualize para a versão mais recente da AWS CLI.

Configure o campo `maxAttempts` com o número máximo de vezes que você deseja que o pod do driver de trabalho seja reiniciado em casos de falhas ou exclusões. O intervalo de execução entre duas tentativas de repetição do driver de trabalho é um intervalo de repetição exponencial de (10 segundos, 20 segundos, 40 segundos...) que é limitado a seis minutos, conforme descrito na documentação do Kubernetes.

> **Note**
>
> Cada execução adicional de um driver de trabalho será cobrada como outra execução de trabalho e estará sujeita aos preços do Amazon EMR no EKS.

Valores de configuração da política de repetição

- Política de repetição padrão para um trabalho: `StartJobRun` inclui uma política de repetição definida como uma tentativa máxima, por padrão. Você pode configurar a política de repetição conforme desejado.

  > **Note**
  >
  > Se `maxAttempts` da `retryPolicyConfiguration` for definido como uma, significa que nenhuma nova tentativa será feita para ativar o pod do driver em caso de falha.

- Desabilitação da política de repetição para um trabalho: para desabilitar uma política de repetição, defina o valor máximo de tentativas em retryPolicyConfiguration como um.

```
"retryPolicyConfiguration": {
    "maxAttempts": 1
}
```

- Definição de maxAttempts para um trabalho dentro do intervalo válido: a chamada `StartJobRun` falhará se o valor de `maxAttempts` estiver fora do intervalo válido. O intervalo válido de

`maxAttempts` é de 1 a 2.147.483.647 (número inteiro de 32 bits), que corresponde ao intervalo compatível com a configuração `backOffLimit` do Kubernetes. Para obter mais informações, consulte Pod backoff failure policy na documentação do Kubernetes. Se o valor de `maxAttempts` for inválido, a seguinte mensagem de erro será retornada:

```
{
 "message": "Retry policy configuration's parameter value of maxAttempts is invalid"
}
```

Recuperação de um status da política de repetição para um trabalho

Você pode visualizar o status das tentativas de repetição de um trabalho com as APIs `ListJobRuns` e `DescribeJobRun`. Depois de solicitar um trabalho com uma configuração de política de repetição habilitada, as respostas `ListJobRun` e `DescribeJobRun` conterão o status da política de repetição no campo `RetryPolicyExecution`. Além disso, a resposta `DescribeJobRun` conterá a `RetryPolicyConfiguration` que foi inserida na solicitação `StartJobRun` para o trabalho.

Exemplo de respostas

ListJobRuns response

```
{
  "jobRuns": [
    ...
    ...
    "retryPolicyExecution" : {
      "currentAttemptCount": 2
    }
    ...
    ...
  ]
}
```

DescribeJobRun response

```
{
  ...
  ...
  "retryPolicyConfiguration": {
    "maxAttempts": 5
```

```
    },
    "retryPolicyExecution" : {
     "currentAttemptCount": 2
   },
   ...
   ...
}
```

Esses campos não ficarão visíveis quando a política de repetição estiver desabilitada no trabalho, conforme descrito abaixo em Valores de configuração da política de repetição.

Monitoramento de um trabalho com uma política de repetição

Quando você habilita uma política de repetição, um evento do CloudWatch é gerado para cada driver de trabalho criado. Para assinar esses eventos, configure uma regra de evento do CloudWatch usando o seguinte comando:

```
aws events put-rule \
--name cwe-test \
--event-pattern '{"detail-type": ["EMR Job Run New Driver Attempt"]}'
```

O evento retornará informações sobre o `newDriverPodName`, o carimbo de data/hora `newDriverCreatedAt`, a `previousDriverFailureMessage` e o `currentAttemptCount` dos drivers de trabalho. Esses eventos não serão criados se a política de repetição estiver desabilitada.

Para obter mais informações sobre como monitorar seu trabalho com eventos do CloudWatch, consulte Monitoramento de trabalhos com o Amazon CloudWatch Events.

Descoberta de logs para drivers e executores

Os nomes dos pods de drivers seguem o formato `spark-<job id>-driver-<random-suffix>`. O mesmo `random-suffix` é adicionado aos nomes dos pods de executores que o driver gera. Ao usar esse `random-suffix`, você pode localizar logs de um driver e seus executores associados. O `random-suffix` estará presente somente se a política de repetição estiver habilitada para o trabalho. Caso contrário, o `random-suffix` estará ausente.

Para obter mais informações sobre como configurar trabalhos com uma configuração de monitoramento para o registro em log, consulte Execução de uma aplicação do Spark.

Uso da alternância de log de eventos do Spark

Com as versões 6.3.0 e posteriores do Amazon EMR, você pode ativar o recurso de alternância de log de eventos do Spark para o Amazon EMR no EKS. Em vez de gerar um único arquivo de log de eventos, esse recurso alterna o arquivo com base no intervalo de tempo configurado e remove os arquivos de log de eventos mais antigos.

A alternância de logs de eventos do Spark pode ajudar você a evitar possíveis problemas com um grande arquivo de log de eventos do Spark gerado para trabalhos de execução prolongada ou de transmissão. Por exemplo, você inicia um trabalho de execução prolongada do Spark com um log de eventos habilitado com o parâmetro `persistentAppUI`. O driver do Spark gera um arquivo de log de eventos. Se o trabalho for executado por horas ou por dias, e houver um espaço em disco limitado no nó do Kubernetes, o arquivo de log de eventos poderá consumir todo o espaço em disco disponível. Ativar o recurso de alternância de log de eventos do Spark resolve o problema ao dividir o arquivo de log em vários arquivos e remover os arquivos mais antigos.

> **Note**
>
> Esse recurso funciona somente com o Amazon EMR no EKS. O Amazon EMR em execução no Amazon EC2 não oferece suporte à alternância de logs de eventos do Spark.

Para ativar o recurso de alternância de log de eventos do Spark, configure os seguintes parâmetros do Spark:

- `spark.eventLog.rotation.enabled`: ativa a alternância de log. Por padrão, ele está desabilitado no arquivo de configuração do Spark. Defina-o como verdadeiro para ativar esse recurso.
- `spark.eventLog.rotation.interval`: especifica o intervalo de tempo para a alternância de log. O valor mínimo é 60 segundos. O valor de padrão é de 300 segundos.
- `spark.eventLog.rotation.minFileSize`: especifica um tamanho mínimo de arquivo para alternar o arquivo de log. O valor mínimo e padrão é de 1 MB.
- `spark.eventLog.rotation.maxFilesToRetain`: especifica quantos arquivos de log alternados serão mantidos durante a limpeza. O intervalo válido é de 1 a 10. O valor padrão é 2.

Você pode especificar esses parâmetros na seção `sparkSubmitParameters` da API `StartJobRun`, como mostra o exemplo a seguir.

```
"sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi --conf
 spark.eventLog.rotation.enabled=true --conf spark.eventLog.rotation.interval=300 --
conf spark.eventLog.rotation.minFileSize=1m --conf
 spark.eventLog.rotation.maxFilesToRetain=2"
```

Uso da alternância de log do contêiner do Spark

Com as versões 6.11.0 e posteriores do Amazon EMR, você pode ativar o recurso de alternância de log do contêiner do Spark para o Amazon EMR no EKS. Em vez de gerar um arquivo único de log `stdout` ou `stderr`, esse recurso alterna o arquivo com base na dimensão de alternância configurada e remove os arquivos de log mais antigos do contêiner.

A alternância de logs do contêiner do Spark pode ajudar você a evitar possíveis problemas com grandes arquivos de log do Spark gerados para trabalhos de execução prolongada ou de transmissão. Por exemplo, é possível iniciar um trabalho do Spark de execução prolongada e o driver do Spark gerar um arquivo de log do contêiner. Se o trabalho for executado por horas ou por dias, e houver um espaço em disco limitado no nó do Kubernetes, o arquivo de log do contêiner poderá consumir todo o espaço em disco disponível. Ao ativar a alternância de log do contêiner do Spark, você divide o arquivo de log em vários arquivos e remove os arquivos mais antigos.

Para ativar o recurso de alternância de log do contêiner do Spark, configure os seguintes parâmetros do Spark:

`containerLogRotationConfiguration`

Inclua esse parâmetro em `monitoringConfiguration` para ativar a alternância de log. O recurso é desabilitado por padrão. Você deve usar `containerLogRotationConfiguration` além de `s3MonitoringConfiguration`.

`rotationSize`

O parâmetro `rotationSize` especifica o tamanho do arquivo para a alternância de log. O intervalo de valores possíveis é de `2KB` a `2GB`. A parcela numérica da unidade do parâmetro `rotationSize` é transferida como um número inteiro. Como não há suporte para valores decimais, você pode especificar um tamanho de rotação de 1,5 GB, por exemplo, com o valor `1500MB`.

`maxFilesToKeep`

O parâmetro `maxFilesToKeep` especifica o número máximo de arquivos a serem retidos no contêiner após a alternância ter ocorrido. O valor mínimo é de 1 e o valor máximo é de 50.

Você pode especificar esses parâmetros na seção `monitoringConfiguration` da API `StartJobRun`, como mostra o exemplo a seguir. Neste exemplo, com `rotationSize = "10 MB"` e `maxFilesToKeep = 3`, o Amazon EMR no EKS alterna seus logs em 10 MB, gera um novo arquivo de log e, em seguida, limpa o arquivo de log mais antigo quando o número de arquivos de log atinge três.

```
{
  "name": "my-long-running-job",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "emr-6.11.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "entryPoint_location",
      "entryPointArguments": ["argument1", "argument2", ...],
       "sparkSubmitParameters": "--class main_class --conf spark.executor.instances=2
 --conf spark.executor.memory=2G --conf spark.executor.cores=2 --conf
 spark.driver.cores=1"
    }
 },
 "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.driver.memory":"2G"
         }
      }
    ],
    "monitoringConfiguration": {
      "persistentAppUI": "ENABLED",
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "my_log_group",
        "logStreamNamePrefix": "log_stream_prefix"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://my_s3_log_location"
      },
      "containerLogRotationConfiguration": {
        "rotationSize":"10MB",
        "maxFilesToKeep":"3"
      }
    }
```

```
  }
}
```

Para iniciar uma execução de trabalho com a alternância de log do contêiner do Spark, inclua um caminho para o arquivo JSON que você configurou com esses parâmetros no comando `StartJobRun`.

```
aws emr-containers start-job-run \
--cli-input-json file://path-to-json-request-file
```

Uso da escalabilidade automática vertical com trabalhos do Spark no Amazon EMR

A escalabilidade automática vertical do Amazon EMR no EKS ajusta automaticamente os recursos de memória e de CPU para se adaptar às necessidades da workload fornecida para aplicações do Spark no Amazon EMR. Isso simplifica o gerenciamento de recursos.

Para rastrear a utilização histórica e em tempo real dos recursos de suas aplicações do Spark no Amazon EMR, a escalabilidade automática vertical utiliza o Vertical Pod Autoscaler (VPA) do Kubernetes. A funcionalidade de escalabilidade automática vertical usa os dados que o VPA coleta para ajustar automaticamente os recursos de memória e de CPU atribuídos às aplicações do Spark. Este processo simplificado aumenta a confiabilidade e otimiza os custos.

Tópicos

- Configuração da escalabilidade automática vertical para o Amazon EMR no EKS
- Conceitos básicos da escalabilidade automática vertical para o Amazon EMR no EKS
- Configuração da escalabilidade automática vertical para o Amazon EMR no EKS
- Monitoramento da escalabilidade automática vertical para o Amazon EMR no EKS
- Desinstalação do operador de escalabilidade automática vertical do Amazon EMR no EKS

Configuração da escalabilidade automática vertical para o Amazon EMR no EKS

Este tópico ajuda você a preparar o cluster do Amazon EKS para enviar trabalhos do Spark no Amazon EMR com a escalabilidade automática vertical. O processo de configuração requer que você confirme ou conclua as tarefas apresentadas nas seguintes seções:

Tópicos

- Pré-requisitos
- Instalação do Operator Lifecycle Manager (OLM) no cluster do Amazon EKS
- Instalação do operador de escalabilidade automática vertical do Amazon EMR no EKS

Pré-requisitos

Conclua as tarefas apresentadas a seguir antes de instalar o operador do Kubernetes para escalabilidade automática vertical em seu cluster. Se você já completou algum dos pré-requisitos, pode ignorá-los e passar para os próximos.

- Instalar a AWS CLI: se você já instalou a AWS CLI, confirme se tem a versão mais recente.
- Instale o kubectl: o kubectl é uma ferramenta de linha de comando que você usa para se comunicar com o servidor de API do Kubernetes. Você precisa do kubectl para instalar e monitorar artefatos relacionados à escalabilidade automática vertical no cluster do Amazon EKS.
- Instale o Operator SDK: o Amazon EMR no EKS usa o Operator SDK como um gerenciador de pacotes durante a vida útil do operador de escalabilidade automática vertical instalado no cluster.
- Instale o Docker: você precisa de acesso à CLI do Docker para autenticar e buscar as imagens do Docker relacionadas à escalabilidade automática vertical para instalar no cluster do Amazon EKS.
- Configuração de um cluster do Amazon EKS (versão 1.24 ou superiores): a escalabilidade automática vertical é compatível com as versões 1.24 e superiores do Amazon EKS. Após criar o cluster, registre-o para uso com o Amazon EMR.
- Selecione um URI de imagem base do Amazon EMR (versão 6.10.0 ou superiores): a escalabilidade automática vertical é compatível com as versões 6.10.0 e superiores do Amazon EMR.

Instalação do Operator Lifecycle Manager (OLM) no cluster do Amazon EKS

Use a CLI do Operator SDK para instalar o Operator Lifecycle Manager (OLM) no cluster do Amazon EMR no EKS no qual você deseja configurar a escalabilidade automática vertical, conforme mostrado no exemplo a seguir. Após configurá-lo, será possível usar o OLM para instalar e gerenciar o ciclo de vida do operador de escalabilidade automática vertical do Amazon EMR.

```
operator-sdk olm install
```

Para validar a instalação, execute o comando `olm status`:

```
operator-sdk olm status
```

Verifique se o comando retorna um resultado com êxito semelhante ao seguinte exemplo de saída:

```
INFO[0007] Successfully got OLM status for version X.XX
```

Se a instalação não ocorrer com êxito, consulte Solução de problemas de escalabilidade automática vertical do Amazon EMR no EKS.

Instalação do operador de escalabilidade automática vertical do Amazon EMR no EKS

Use as seguintes etapas para instalar o operador de escalabilidade automática vertical no cluster do Amazon EKS:

1. Configure as seguintes variáveis de ambiente que serão usadas para concluir a instalação:
   - **$REGION** direciona para a Região da AWS do seu cluster. Por exemplo, `us-west-2`.
   - **$ACCOUNT_ID** direciona para o ID da conta do Amazon ECR para sua região. Para obter mais informações, consulte Contas de registro do Amazon ECR por região.
   - **$RELEASE** direciona para a versão do Amazon EMR que você deseja usar para o cluster. Com a escalabilidade automática vertical, você deve usar o Amazon EMR versão 6.10.0 ou superiores.
2. Em seguida, obtenha tokens de autenticação para o registro do Amazon ECR do operador.

   ```
   aws ecr get-login-password \
    --region region-id | docker login \
    --username AWS \
    --password-stdin $ACCOUNT_ID.dkr.ecr.region-id.amazonaws.com
   ```
3. Instale o operador de escalabilidade automática vertical do Amazon EMR no EKS com o seguinte comando:

   ```
   ECR_URL=$ACCOUNT_ID.dkr.ecr.$REGION.amazonaws.com && \
   REPO_DEST=dynamic-sizing-k8s-operator-olm-bundle && \
   BUNDLE_IMG=emr-$RELEASE-dynamic-sizing-k8s-operator && \
   operator-sdk run bundle \
   $ECR_URL/$REPO_DEST/$BUNDLE_IMG\:latest
   ```

Isso criará uma versão do operador de escalabilidade automática vertical no namespace padrão do cluster do Amazon EKS. Use este comando para realizar a instalação em um namespace diferente:

```
operator-sdk run bundle \
$ACCOUNT_ID.dkr.ecr.$REGION.amazonaws.com/dynamic-sizing-k8s-operator-olm-bundle/
emr-$RELEASE-dynamic-sizing-k8s-operator:latest \
-n operator-namespace
```

> **Note**
>
> Se o namespace especificado não existir, o OLM não instalará o operador. Para obter mais informações, consulte Namespace do Kubernetes não encontrado.

4. Verifique se você instalou o operador com êxito usando a ferramenta de linha de comando kubectl do Kubernetes.

```
kubectl get csv -n operator-namespace
```

O comando `kubectl` deve retornar o operador de escalabilidade automática vertical recém-implantado com um status de Fase como Com êxito. Se você tiver problemas com a instalação ou com a configuração, consulte Solução de problemas de escalabilidade automática vertical do Amazon EMR no EKS.

Conceitos básicos da escalabilidade automática vertical para o Amazon EMR no EKS

Envio de um trabalho do Spark com a escalabilidade automática vertical

Ao enviar um trabalho usando a API StartJobRun, adicione as duas seguintes configurações ao driver do trabalho do Spark para ativar a escalabilidade automática vertical:

```
"spark.kubernetes.driver.label.emr-containers.amazonaws.com/dynamic.sizing":"true",
"spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/
dynamic.sizing.signature":"YOUR_JOB_SIGNATURE"
```

No código acima, a primeira linha ativa a funcionalidade de escalabilidade automática vertical. A linha seguinte é uma configuração de assinatura obrigatória que permite que você escolha uma assinatura para o seu trabalho.

Para obter mais informações sobre essas configurações e valores de parâmetros aceitáveis, consulte Configuração da escalabilidade automática vertical para o Amazon EMR no EKS. Por padrão, seu trabalho é enviado com a opção somente monitoramento no modo Desativado para a escalabilidade automática vertical. Esse estado de monitoramento permite calcular e visualizar recomendações de recursos sem realizar a escalabilidade automática. Para obter mais informações, consulte Modos da escalabilidade automática vertical.

O seguinte exemplo mostra como concluir um comando `start-job-run` de exemplo com a escalabilidade automática vertical:

```
aws emr-containers start-job-run \
--virtual-cluster-id $VIRTUAL_CLUSTER_ID \
--name $JOB_NAME \
--execution-role-arn $EMR_ROLE_ARN \
--release-label emr-6.10.0-latest \
--job-driver '{
  "sparkSubmitJobDriver": {
     "entryPoint": "local:///usr/lib/spark/examples/src/main/python/pi.py"
   }
 }' \
--configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "spark-defaults",
        "properties": {
          "spark.kubernetes.driver.label.emr-containers.amazonaws.com/dynamic.sizing":
 "true",
          "spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/
dynamic.sizing.signature": "test-signature"
        }
    }]
  }'
```

Verificação de funcionalidade da escalabilidade automática vertical

Para verificar se a escalabilidade automática vertical está funcionando corretamente para o trabalho enviado, use kubectl para obter o recurso personalizado `verticalpodautoscaler` e visualizar suas recomendações de escalabilidade. Por exemplo, o seguinte comando consulta recomendações

sobre o trabalho de exemplo da seção Envio de um trabalho do Spark com a escalabilidade automática vertical:

```
kubectl get verticalpodautoscalers --all-namespaces \
-l=emr-containers.amazonaws.com/dynamic.sizing.signature=test-signature
```

A saída desta consulta deve ser semelhante à seguinte:

```
NAME                                                        MODE   CPU         MEM
 PROVIDED   AGE
ds-jceyefkxnhrvdzw6djum3naf2abm6o63a6dvjkkedqtkhlrf25eq-vpa   Off    3304504865  True
         87m
```

Se a saída não for semelhante ou tiver um código de erro, consulte Solução de problemas de escalabilidade automática vertical do Amazon EMR no EKS para obter passos que ajudam a resolver o problema.

Configuração da escalabilidade automática vertical para o Amazon EMR no EKS

Você pode configurar a escalabilidade automática vertical ao enviar trabalhos do Spark no Amazon EMR usando a API StartJobRun. Defina os parâmetros de configuração relacionados à escalabilidade automática no pod do driver do Spark, conforme mostrado no exemplo em Envio de um trabalho do Spark com a escalabilidade automática vertical.

O operador de escalabilidade automática vertical do Amazon EMR no EKS recebe os pods de drivers que têm escalabilidade automática e, em seguida, configura a integração com o Vertical Pod Autoscaler (VPA) do Kubernetes usando as configurações no pod do driver. Isso facilita o rastreamento de recursos e a escalabilidade automática de pods de executores do Spark.

As seções a seguir descrevem os parâmetros que você pode usar ao configurar a escalabilidade automática vertical para o cluster do Amazon EKS.

> **Note**
>
> Configure o parâmetro de alternância de recursos como um rótulo e configure os parâmetros restantes como anotações no pod do driver do Spark. Os parâmetros de escalabilidade automática pertencem ao domínio `emr-containers.amazonaws.com/` e têm o prefixo `dynamic.sizing`.

Parâmetros necessários

Você deve incluir os dois seguintes parâmetros no driver de trabalho do Spark ao enviar o trabalho:

Chave	Descrição	Valores aceitos	Valor padrão	Tipo	Parâmetro do Spark[1]
`dynamic.sizing`	Alternância de recursos	`true`, `false`	não definido	label	`spark.kubernetes.driver.label.emr-containers.amazonaws.com/dynamic.sizing`
`dynamic.sizing.signature`	Assinatura do trabalho	string	não definido	anotação	`spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.signature`

[1] Use esse parâmetro como um `SparkSubmitParameter` ou uma `ConfigurationOverride` na API `StartJobRun`.

- **`dynamic.sizing`**: é possível ativar e desativar a escalabilidade automática vertical com o rótulo `dynamic.sizing`. Para ativar a escalabilidade automática vertical, defina `dynamic.sizing` como `true` no pod do driver do Spark. Se você omitir esse rótulo ou defini-lo com qualquer valor diferente de `true`, a escalabilidade automática vertical será desativada.

- **`dynamic.sizing.signature`**: defina a assinatura do trabalho com a anotação `dynamic.sizing.signature` no pod do driver. A escalabilidade automática vertical agrega dados de uso de recursos em diferentes execuções de trabalhos do Spark no Amazon EMR para derivar recomendações de recursos. Você fornece o identificador exclusivo para unir os trabalhos.

> **Note**
>
> Se o seu trabalho se repetir em um intervalo fixo, como diariamente ou semanalmente, a assinatura do trabalho deverá permanecer a mesma para cada nova instância dele. Isso garante que a escalabilidade automática vertical possa computar e agregar recomendações em diferentes execuções do trabalho.

[1] Use esse parâmetro como um `SparkSubmitParameter` ou uma `ConfigurationOverride` na API `StartJobRun`.

Parâmetros opcionais

A escalabilidade automática vertical também oferece suporte aos parâmetros opcionais apresentados a seguir. Defina-os como anotações no pod do driver.

Chave	Descrição	Valores aceitos	Valor padrão	Tipo	Parâmetro do Spark[1]
dynamic.sizing.mode	Modo da escalabilidade automática vertical	`Off`, `Initial`, `Auto`	`Off`	anotação	`spark.kubernetes.driver.label.emr-containers.amazonaws.com/dynamic.sizing.mode`
dynamic.sizing.sca	Habilitação da escalabil	*`true`*, *`false`*	`true`	anotação	`spark.kubernetes.driver.lab`

Chave	Descrição	Valores aceitos	Valor padrão	Tipo	Parâmetro do Spark[1]
le.memory	idade de memória				el.emr-containers.amazonaws.com/dynamic.sizing.scale.memory
dynamic.sizing.scale.cpu	Ativação ou desativação da escalabilidade de CPU	*true*, *false*	false	anotação	spark.kubernetes.driver.label.emr-containers.amazonaws.com/dynamic.sizing.scale.cpu
dynamic.sizing.scale.memory.min	Limite mínimo para a escalabilidade de memória	string, K8s resource. Quantity, por exemplo: 1G	não definido	anotação	spark.kubernetes.driver.label.emr-containers.amazonaws.com/dynamic.sizing.scale.memory.min

Chave	Descrição	Valores aceitos	Valor padrão	Tipo	Parâmetro do Spark[1]
dynamic.sizing.scale.memory.max	Limite máximo para a escalabilidade de memória	string, K8s resource.Quantity, por exemplo: 4G	não definido	anotação	spark.kubernetes.driver.label.emr-containers.amazonaws.com/dynamic.sizing.scale.memory.max
dynamic.sizing.scale.cpu.min	Limite mínimo para a escalabilidade de CPU	string, K8s resource.Quantity, por exemplo: 1	não definido	anotação	spark.kubernetes.driver.label.emr-containers.amazonaws.com/dynamic.sizing.scale.cpu.min
dynamic.sizing.scale.cpu.max	Limite máximo para a escalabilidade de CPU	string, K8s resource.Quantity, por exemplo: 2	não definido	anotação	spark.kubernetes.driver.label.emr-containers.amazonaws.com/dynamic.sizing.scale.cpu.max

Modos da escalabilidade automática vertical

O parâmetro `mode` é mapeado para os diferentes modos de escalabilidade automática compatíveis com o VPA. Use a anotação `dynamic.sizing.mode` no pod do driver para definir o modo. Os seguintes valores têm suporte para este parâmetro:

- Desativado: um modo de simulação no qual é possível monitorar recomendações, mas a escalabilidade automática não é executada. Este é o modo padrão para a escalabilidade automática vertical. Nesse modo, o recurso de escalabilidade automática vertical do pod associado computa recomendações e você pode monitorá-las usando ferramentas como o kubectl, o Prometheus e o Grafana.
- Inicial: neste modo, o VPA escala automaticamente os recursos quando o trabalho é iniciado, se houver recomendações disponíveis com base no histórico de execuções do trabalho, como no caso de um trabalho recorrente.
- Automático: neste modo, o VPA realiza a remoção de pods de executores do Spark e os escala automaticamente com as configurações de recursos recomendadas quando o pod do driver do Spark os reinicia. Às vezes, o VPA realiza a remoção de pods de executores do Spark em execução, portanto, isso pode resultar em latência adicional ao tentar novamente o executor interrompido.

Escalabilidade de recursos

Ao configurar a escalabilidade automática vertical, você pode escolher se deseja escalar os recursos de CPU e de memória. Defina as anotações `dynamic.sizing.scale.cpu` e `dynamic.sizing.scale.memory` como `true` ou `false`. Por padrão, a escalabilidade de CPU é definida como `false` e a escalabilidade de memória é definida como `true`.

Mínimos e máximos de recursos (limites)

Como opção, também é possível definir limites para os recursos de CPU e de memória. Escolha valores mínimo e máximo para esses recursos com as anotações `dynamic.sizing.[memory/cpu].[min/max]` ao ativar a escalabilidade automática. Por padrão, os recursos não têm limitações. Defina as anotações como valores de string que representam uma quantidade de recursos do Kubernetes. Por exemplo, defina `dynamic.sizing.memory.max` como `4G` para representar 4 GB.

Monitoramento da escalabilidade automática vertical para o Amazon EMR no EKS

Você pode usar a ferramenta de linha de comando kubectl do Kubernetes para listar as recomendações ativas relacionadas à escalabilidade automática vertical em seu cluster. Também é possível visualizar as assinaturas de trabalho rastreadas e limpar quaisquer recursos desnecessários associados às assinaturas.

Listagem das recomendações de escalabilidade automática vertical para o cluster

Use kubectl para obter o recurso `verticalpodautoscaler` e visualizar o status atual e as recomendações. O exemplo de consulta a seguir retorna todos os recursos ativos no cluster do Amazon EKS.

```
kubectl get verticalpodautoscalers \
-o custom-columns="NAME:.metadata.name,"\
"SIGNATURE:.metadata.labels.emr-containers\.amazonaws\.com/dynamic\.sizing
\.signature,"\
"MODE:.spec.updatePolicy.updateMode,"\
"MEM:.status.recommendation.containerRecommendations[0].target.memory" \
--all-namespaces
```

A saída desta consulta é semelhante à seguinte:

```
NAME                  SIGNATURE                MODE      MEM
ds-example-id-1-vpa   job-signature-1          Off       none
ds-example-id-2-vpa   job-signature-2          Initial   12936384283
```

Consulta e exclusão das recomendações de escalabilidade automática vertical para o cluster

Quando você exclui um recurso de execução de trabalho de escalabilidade automática vertical do Amazon EMR, ele exclui automaticamente o objeto VPA associado que rastreia e armazena recomendações.

O seguinte exemplo usa kubectl para limpar as recomendações para um trabalho identificado por uma assinatura:

```
kubectl delete jobrun -n emr -l=emr-containers\.amazonaws\.com/dynamic\.sizing
\.signature=integ-test
jobrun.dynamicsizing.emr.services.k8s.aws "ds-job-signature" deleted
```

Se você não souber a assinatura específica do trabalho ou desejar limpar todos os recursos do cluster, poderá usar `--all` ou `--all-namespaces` em seu comando, em vez do ID exclusivo do trabalho, conforme mostrado no seguinte exemplo:

```
kubectl delete jobruns --all --all-namespaces
jobrun.dynamicsizing.emr.services.k8s.aws "ds-example-id" deleted
```

Desinstalação do operador de escalabilidade automática vertical do Amazon EMR no EKS

Se você desejar remover o operador de escalabilidade automática vertical do cluster do Amazon EKS, use o comando `cleanup` com a CLI do Operator SDK, conforme mostrado no exemplo a seguir. Isso também excluirá as dependências upstream instaladas com o operador, como o Vertical Pod Autoscaler.

```
operator-sdk cleanup emr-dynamic-sizing
```

Se houver trabalhos em execução no cluster quando você excluir o operador, esses trabalhos continuarão em execução sem a escalabilidade automática vertical. Se você enviar trabalhos no cluster após a exclusão o operador, o Amazon EMR no EKS ignorará quaisquer parâmetros relacionados à escalabilidade automática vertical que você possa ter definido durante a configuração.

Execução de workloads interativas no Amazon EMR no EKS

Um endpoint interativo corresponde a um gateway que conecta o Amazon EMR Studio ao Amazon EMR no EKS para que você possa executar workloads interativas. É possível usar endpoints interativos com o EMR Studio para executar análises interativas com conjuntos de dados em armazenamentos de dados como o Amazon S3 e o Amazon DynamoDB.

Casos de uso

- Criação de um script de ETL com a experiência do IDE do EMR Studio. O IDE ingere dados on-premises e os armazena no Amazon S3 após as transformações para análises posteriores.
- Uso de cadernos para explorar conjuntos de dados e treinar um modelo de machine learning para detectar anomalias nos conjuntos de dados.
- Criação de scripts que geram relatórios diários para aplicações de análise, como painéis de negócios.

Tópicos

- Visão geral dos endpoints interativos
- Pré-requisitos para a criação de um endpoint interativo no Amazon EMR no EKS
- Criação de um endpoint interativo para o cluster virtual
- Definição de configurações para endpoints interativos
- Monitoramento de endpoints interativos
- Uso de cadernos Jupyter de hospedagem própria
- Outras operações em um endpoint interativo

Visão geral dos endpoints interativos

Um endpoint interativo fornece a funcionalidade para os clientes interativos, como os que usam o Amazon EMR Studio, de se conectarem ao Amazon EMR em clusters do EKS para executar workloads interativas. O endpoint interativo está respaldado pelo Jupyter Enterprise Gateway que fornece a funcionalidade de gerenciamento remoto do ciclo de vida do kernel de que os clientes interativos precisam. Os kernels são processos específicos de linguagem que interagem com o cliente do Amazon EMR Studio baseado em Jupyter para executar workloads interativas.

Os endpoints interativos oferecem suporte aos seguintes kernels:

- Python 3
- PySpark no Kubernetes
- Apache Spark com o Scala

> **Note**
>
> Os preços do Amazon EMR no EKS se aplicam aos endpoints e aos kernels interativos. Para obter mais informações, consulte a página Preços do Amazon EMR no EKS.

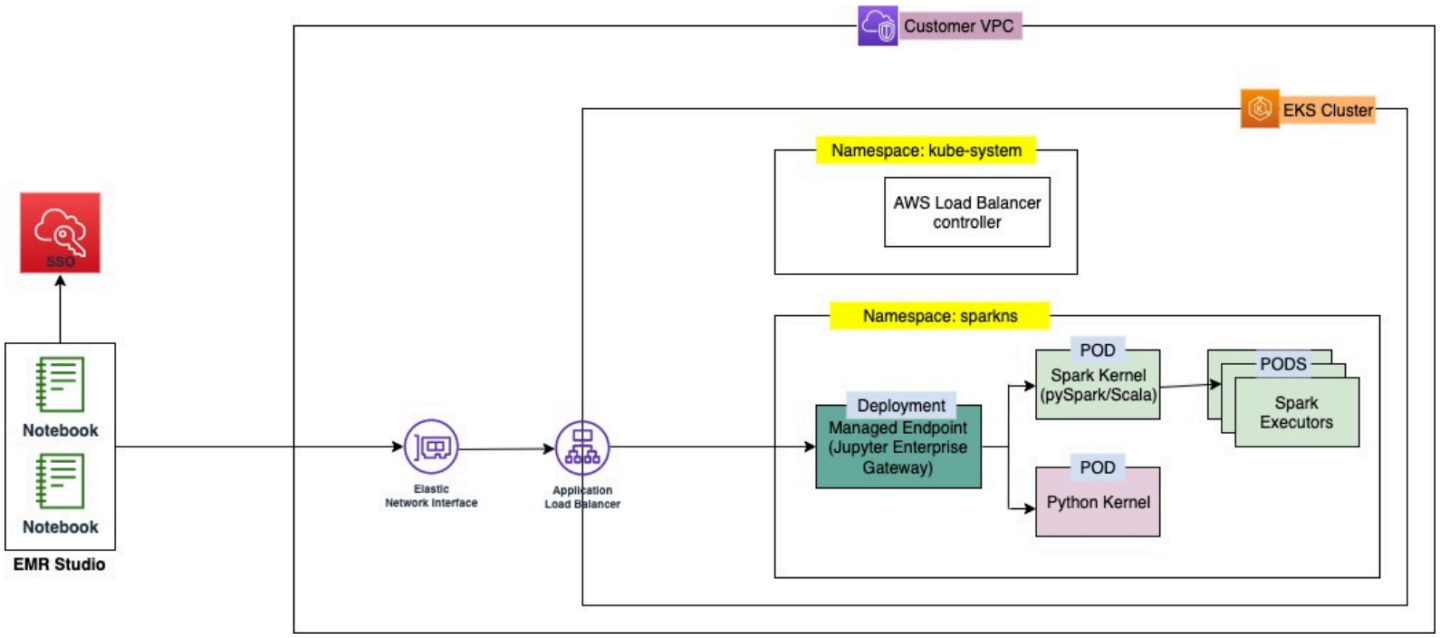
As entidades apresentadas a seguir são necessárias para que o EMR Studio se conecte ao Amazon EMR no EKS.

- Cluster virtual do Amazon EMR no EKS: um cluster virtual corresponde a um namespace do Kubernetes no qual você registra o Amazon EMR. O Amazon EMR usa clusters virtuais para executar trabalhos e hospedar endpoints. É possível fazer backup de vários clusters virtuais usando o mesmo cluster físico. No entanto, cada cluster virtual é mapeado para um namespace em um cluster do Amazon EKS. Os clusters virtuais não criam quaisquer recursos ativos que contribuam para o seu faturamento ou que requeiram gerenciamento do ciclo de vida de forma externa ao serviço.
- Endpoint interativo do Amazon EMR no EKS: um endpoint interativo corresponde a um endpoint HTTPS ao qual os usuários do EMR Studio podem conectar um Workspace. É possível acessar os endpoints HTTPS somente ao usar o EMR Studio e criá-los em uma sub-rede privada da Amazon Virtual Private Cloud (Amazon VPC) para o cluster do Amazon EKS.

  Os kernels Python, PySpark e Spark Scala usam as permissões definidas em seu perfil de execução de trabalho do Amazon EMR no EKS para invocar outros Serviços da AWS. Todos os kernels e os usuários que se conectam ao endpoint interativo utilizam o perfil que você especificou ao criar o endpoint. Recomendamos que você crie endpoints separados para usuários diferentes e que os usuários tenham perfis do AWS Identity and Access Management (IAM) diferentes.
- Controlador do AWS Application Load Balancer: o controlador do AWS Application Load Balancer gerencia o Elastic Load Balancing para um cluster do Kubernetes do Amazon EKS. O controlador provisiona um Application Load Balancer (ALB) quando você cria um recurso Ingress do Kubernetes. Um ALB revela um serviço do Kubernetes, como um endpoint interativo, de forma externa ao cluster do Amazon EKS, mas dentro da mesma Amazon VPC. Ao criar um endpoint interativo, também ocorre a implementação de um recurso Ingress que revela o endpoint interativo

por meio do ALB para os clientes interativos se conectarem. Você precisa somente instalar um controlador do AWS Application Load Balancer para cada cluster do Amazon EKS.

O diagrama a seguir descreve a arquitetura dos endpoints interativos no Amazon EMR no EKS. Um cluster do Amazon EKS compreende a computação para executar as workloads de análise e o endpoint interativo. O controlador do Application Load Balancer é executado no namespace `kube-system`, enquanto as workloads e os endpoints interativos são executados no namespace especificado ao criar o cluster virtual. Quando você cria um endpoint interativo, o ambiente de gerenciamento do Amazon EMR no EKS cria a implantação do endpoint interativo no cluster do Amazon EKS. Além disso, uma instância para o recurso Ingress do Application Load Balancer é criada pelo controlador do AWS Load Balancer. O Application Load Balancer fornece a interface externa para que os clientes, como o EMR Studio, se conectem ao cluster do Amazon EMR e executem workloads interativas.



Pré-requisitos para a criação de um endpoint interativo no Amazon EMR no EKS

Esta seção descreve os pré-requisitos para configurar um endpoint interativo que o EMR Studio poderá usar para se conectar a um cluster do Amazon EMR no EKS e executar workloads interativas.

AWS CLI

Siga as etapas descritas em Instalar a AWS CLI para instalar a versão mais recente da AWS Command Line Interface (AWS CLI).

Instalação do eksctl

Siga as etapas descritas em Instalar o `eksctl` para instalar a versão mais recente do eksctl. Se você estiver usando a versão 1.22 ou versões posteriores do Kubernetes para o cluster do Amazon EKS, use uma versão do eksctl superior a 0.117.0.

Cluster do Amazon EKS

Crie um cluster do Amazon EKS. Registre o cluster como um cluster virtual com o Amazon EMR no EKS. Confira abaixo os requisitos e as considerações para este cluster:

- O cluster deve estar na mesma Amazon Virtual Private Cloud (VPC) que seu EMR Studio.
- O cluster deve ter, no mínimo, uma sub-rede privada para ativar os endpoints interativos, vincular repositórios baseados em Git e iniciar o Application Load Balancer no modo privado.
- Deve existir, no mínimo, uma sub-rede privada em comum entre o EMR Studio e o cluster do Amazon EKS usado para registrar o cluster virtual. Isso garante que seu endpoint interativo apareça como uma opção nos WorkSpaces do Studio e ativa a conectividade do Studio com o Application Load Balancer.

  Existem dois métodos que você pode escolher para conectar o Studio e o cluster do Amazon EKS:

  - Criar um cluster do Amazon EKS e associá-lo às sub-redes que pertencem ao seu EMR Studio.
  - Como alternativa, crie um EMR Studio e especifique as sub-redes privadas para o cluster do Amazon EKS.
- O ARM de AMIs do Amazon Linux otimizadas para o Amazon EKS não são compatíveis com endpoints interativos do Amazon EMR no EKS.
- Se você usar um cluster do EKS somente para o AWS Fargate (Fargate), remova os taints do nó `eks.amazonaws.com/compute-type=fargate:NoSchedule` de todos os nós do Fargate.
- Os endpoints interativos funcionam com clusters do Amazon EKS que usam versões do Kubernetes até 1.27.
- Somente grupos de nós gerenciados do Amazon EKS são compatíveis.

Concessão de acesso ao cluster para o Amazon EMR no EKS

Use as etapas em Concessão de acesso ao cluster para o Amazon EMR no EKS para conceder acesso a um namespace específico em seu cluster para o Amazon EMR no EKS.

Ativação de IRSA no cluster do Amazon EKS

Para ativar os perfis do IAM para contas de serviço (IRSA) no cluster do Amazon EKS, siga as etapas em Habilitação de perfis do IAM para contas de serviço (IRSA).

Criação de um perfil de execução de trabalho do IAM

Você deve criar um perfil do IAM para executar workloads em endpoints interativos do Amazon EMR no EKS. Referimo-nos a esse perfil do IAM como perfil de execução de trabalho nesta documentação. Esse perfil do IAM é atribuído ao contêiner de endpoint interativo e aos contêineres de execução reais criados quando você envia trabalhos com o EMR Studio. Você precisará do nome do recurso da Amazon (ARN) do seu perfil de execução de trabalho para o Amazon EMR no EKS. Para isso, são necessárias duas etapas:

- Crie um perfil do IAM para execução de trabalhos.
- Atualize a política de confiança do perfil de execução de trabalho.

Concessão de acesso ao Amazon EMR no EKS para os usuários

A entidade do IAM (usuário ou perfil) que faz a solicitação para criar um endpoint interativo também deve ter as permissões do Amazon EC2 e de `emr-containers` apresentadas a seguir. Siga as etapas descritas em Concessão de acesso ao Amazon EMR no EKS para os usuários para conceder as permissões que permitem que o Amazon EMR no EKS crie, gerencie e exclua os grupos de segurança que limitam o tráfego de entrada para o balanceador de carga do seu endpoint interativo.

As seguintes permissões de `emr-containers` permitem que o usuário execute operações básicas do endpoint interativo:

```
"ec2:CreateSecurityGroup",
"ec2:DeleteSecurityGroup",
"ec2:AuthorizeSecurityGroupEgress",
"ec2:AuthorizeSecurityGroupIngress",
"ec2:RevokeSecurityGroupEgress",
"ec2:RevokeSecurityGroupIngress"
```

```
"emr-containers:CreateManagedEndpoint",
"emr-containers:ListManagedEndpoints",
"emr-containers:DescribeManagedEndpoint",
"emr-containers:DeleteManagedEndpoint"
```

Registro do cluster do Amazon EKS com o Amazon EMR

Configure um cluster virtual e mapeie-o para o namespace no cluster do Amazon EKS no qual você deseja executar os trabalhos. Para clusters somente do AWS Fargate, use o mesmo namespace para o cluster virtual do Amazon EMR no EKS e para o perfil do Fargate.

Para obter informações sobre como configurar um cluster virtual do Amazon EMR no EKS, consulte Registro do cluster do Amazon EKS com o Amazon EMR.

Implantação do AWS Load Balancer Controller no cluster do Amazon EKS

Um AWS Application Load Balancer é necessário para o cluster do Amazon EKS. Você precisa configurar somente um controlador do Application Load Balancer por cluster do Amazon EKS. Para obter informações sobre como configurar o controlador do AWS Application Load Balancer, consulte Instalar o complemento AWS Load Balancer Controller no Guia do usuário do Amazon EKS.

Criação de um endpoint interativo para o cluster virtual

Esta página descreve como criar um endpoint interativo usando a AWS Command Line Interface (AWS CLI).

Criação de um endpoint interativo com o comando **`create-managed-endpoint`**

Especifique os parâmetros no comando `create-managed-endpoint`, conforme apresentado a seguir. O Amazon EMR no EKS oferece suporte à criação de endpoints interativos com as versões 6.7.0 e superiores do Amazon EMR.

```
aws emr-containers create-managed-endpoint \
--type JUPYTER_ENTERPRISE_GATEWAY \
--virtual-cluster-id 1234567890abcdef0xxxxxxxx \
--name example-endpoint-name \
--execution-role-arn arn:aws:iam::444455556666:role/JobExecutionRole \
```

```
--release-label emr-6.9.0-latest \
--configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "spark-defaults",
        "properties": {
            "spark.driver.memory": "2G"
        }
    }],
    "monitoringConfiguration": {
        "cloudWatchMonitoringConfiguration": {
            "logGroupName": "log_group_name",
            "logStreamNamePrefix": "log_stream_prefix"
        },
        "persistentAppUI": "ENABLED",
        "s3MonitoringConfiguration": {
            "logUri": "s3://my_s3_log_location"
        }
    }
}'
```

Para obter mais informações, consulte Parâmetros para a criação de um endpoint interativo.

Criação de um endpoint interativo com parâmetros especificados em um arquivo JSON

1. Crie um arquivo `create-managed-endpoint-request.json` e especifique os parâmetros obrigatórios para o endpoint, conforme mostrado no seguinte arquivo JSON:

```
{
    "name": "MY_TEST_ENDPOINT",
    "virtualClusterId": "MY_CLUSTER_ID",
    "type": "JUPYTER_ENTERPRISE_GATEWAY",
    "releaseLabel": "emr-6.9.0-latest",
    "executionRoleArn": "arn:aws:iam::444455556666:role/JobExecutionRole",
    "configurationOverrides":
    {
        "applicationConfiguration":
        [
            {
                "classification": "spark-defaults",
                "properties":
                {
```

```
                    "spark.driver.memory": "8G"
                }
            }
        ],
        "monitoringConfiguration":
        {
            "persistentAppUI": "ENABLED",
            "cloudWatchMonitoringConfiguration":
            {
                "logGroupName": "my_log_group",
                "logStreamNamePrefix": "log_stream_prefix"
            },
            "s3MonitoringConfiguration":
            {
                "logUri": "s3://my_s3_log_location"
            }
        }
    }
}
```

2. Use o comando `create-managed-endpoint` com um caminho para o arquivo `create-managed-endpoint-request.json` armazenado localmente ou no Amazon S3.

```
aws emr-containers create-managed-endpoint \
--cli-input-json  file://./create-managed-endpoint-request.json --region AWS-Region
```

Saída da criação de endpoint interativo

Você deverá visualizar a saída a seguir no terminal. A saída inclui o nome e o identificador do seu novo endpoint interativo:

```
{
    "id": "1234567890abcdef0",
    "name": "example-endpoint-name",
    "arn": "arn:aws:emr-containers:us-west-2:111122223333:/
virtualclusters/444455556666/endpoints/444455556666",
    "virtualClusterId": "111122223333xxxxxxxx"
}
```

A execução de `aws emr-containers create-managed-endpoint` cria um certificado autoassinado que permite a comunicação HTTPS entre o EMR Studio e o servidor do endpoint interativo.

Se você executar `create-managed-endpoint` e não tiver concluído os pré-requisitos, o Amazon EMR retornará uma mensagem de erro com as ações que você deve realizar para continuar.

Parâmetros para a criação de um endpoint interativo

Tópicos

- Parâmetros obrigatórios para endpoints interativos
- Parâmetros opcionais para endpoints interativos

Parâmetros obrigatórios para endpoints interativos

Você deve especificar os seguintes parâmetros ao criar um endpoint interativo:

`--type`

Use `JUPYTER_ENTERPRISE_GATEWAY`. Este é o único tipo com suporte.

`--virtual-cluster-id`

O identificador do cluster virtual registrado com o Amazon EMR no EKS.

`--name`

Um nome descritivo para o endpoint interativo que auxilia os usuários do EMR Studio a selecioná-lo na lista suspensa.

`--execution-role-arn`

O nome do recurso da Amazon (ARN) do perfil de execução de trabalho do IAM para o Amazon EMR no EKS que foi criado como parte dos pré-requisitos.

`--release-label`

O rótulo da versão do Amazon EMR a ser usado para o endpoint. Por exemplo, `emr-6.9.0-latest`. O Amazon EMR no EKS oferece suporte a endpoints interativos com as versões 6.7.0 e superiores do Amazon EMR.

Parâmetros opcionais para endpoints interativos

Como opção, também é possível especificar os seguintes parâmetros ao criar um endpoint interativo:

`--configuration-overrides`

Para substituir as configurações padrão das aplicações, forneça um objeto de configuração. Você pode usar uma sintaxe abreviada para fornecer a configuração ou fazer referência ao objeto de configuração em um arquivo JSON.

Os objetos de configuração consistem em uma classificação, propriedades e configurações opcionais aninhadas As propriedades consistem nas configurações que você deseja substituir neste arquivo. Você pode especificar várias classificações para diversas aplicações em um único objeto JSON. As classificações de configuração disponíveis variam de acordo com a versão do Amazon EMR no EKS. Para obter uma lista das classificações de configuração disponíveis para cada versão do Amazon EMR no EKS, consulte Versões do Amazon EMR no EKS. Além das classificações de configuração listadas para cada versão, os endpoints interativos trazem a classificação adicional `jeg-config`. Para obter mais informações, consulte Opções de configuração do Jupyter Enterprise Gateway (JEG).

Definição de configurações para endpoints interativos

Monitorar trabalhos Spark do

Para que seja possível monitorar e solucionar falhas, configure os endpoints interativos para que os trabalhos iniciados com o endpoint possam enviar informações de log para o Amazon S3, para o Amazon CloudWatch Logs ou para ambos. As seções a seguir descrevem como enviar logs de aplicações do Spark para o Amazon S3 para os trabalhos do Spark executados com endpoints interativos do Amazon EMR no EKS.

Configuração da política do IAM para os logs do Amazon S3

Antes que seus kernels possam enviar dados de log ao Amazon S3, a política de permissões para o perfil de execução de trabalho deve incluir as permissões apresentadas a seguir. Substitua *DOC-EXAMPLE-BUCKET-LOGGING* pelo nome do bucket de registro em log.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-LOGGING",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-LOGGING/*",
            ]
        }
    ]
}
```

> **Note**
>
> O Amazon EMR no EKS também pode criar um bucket do S3. Se um bucket do S3 não estiver disponível, inclua a permissão `s3:CreateBucket` na política do IAM.

Após conceder as permissões necessárias para o envio de logs ao bucket do S3 ao perfil de execução, os dados de log serão enviados para os locais do Amazon S3 apresentados a seguir. Isso acontece quando `s3MonitoringConfiguration` é transferido na seção `monitoringConfiguration` de uma solicitação `create-managed-endpoint`.

- Logs de driver: `logUri/virtual-cluster-id/endpoints/endpoint-id/containers/spark-application-id/spark-application-id-driver/(stderr.gz/stdout.gz)`.
- Logs de executor: `logUri/virtual-cluster-id/endpoints/endpoint-id/containers/spark-application-id/executor-pod-name-exec-<Number>/(stderr.gz/stdout.gz)`.

> **Note**
>
> O Amazon EMR no EKS não faz o upload dos logs do endpoint para o bucket do S3.

Especificação de modelos de pod personalizados com endpoints interativos

É possível criar endpoints interativos nos quais você especifica modelos de pod personalizados para drivers e executores. Os modelos de pod são especificações que determinam como ocorrerá a execução de cada pod. Você pode usar arquivos de modelo de pod para definir as configurações de pods de drivers ou de executores para as quais as configurações do Spark não oferecem suporte. No momento, os modelos de pod são compatíveis com as versões 6.3.0 e posteriores do Amazon EMR.

Para obter mais informações sobre modelos de pod, consulte Using pod templates no Guia de desenvolvimento do Amazon EMR no EKS.

O seguinte exemplo mostra como criar um endpoint interativo com modelos de pod:

```
aws emr-containers create-managed-endpoint \
    --type JUPYTER_ENTERPRISE_GATEWAY \
    --virtual-cluster-id virtual-cluster-id \
    --name example-endpoint-name \
    --execution-role-arn arn:aws:iam::aws-account-id:role/EKSClusterRole \
    --release-label emr-6.9.0-latest \
    --configuration-overrides '{
        "applicationConfiguration": [
        {
            "classification": "spark-defaults",
            "properties": {
                "spark.kubernetes.driver.podTemplateFile": "path/to/driver/
template.yaml",
                "spark.kubernetes.executor.podTemplateFile": "path/to/executor/
template.yaml"
            }
        }]
    }'
```

Implantação de um pod do JEG em um grupo de nós

O posicionamento do pod do JEG (Jupyter Enterprise Gateway) é um recurso que permite implantar um endpoint interativo em um grupo de nós específico. Com esse recurso, você pode definir configurações, como `instance type`, para o endpoint interativo.

Associação de um pod do JEG a um grupo de nós gerenciado

A propriedade de configuração apresentada a seguir permite especificar o nome de um grupo de nós gerenciado no cluster do Amazon EKS em que o pod do JEG será implantado.

```
//payload
--configuration-overrides '{
      "applicationConfiguration": [
            {
                "classification": "endpoint-configuration",
                "properties": {
                    "managed-nodegroup-name": NodeGroupName
                }
            }
        ]
    }'
```

Um grupo de nós deve ter o rótulo `for-use-with-emr-containers-managed-endpoint-ng=`*NodeGroupName* do Kubernetes anexado a todos os nós que fazem parte do grupo de nós. Para listar todos os nós de um grupo de nós que têm essa etiqueta, use o seguinte comando:

```
kubectl get nodes --show-labels | grep for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
```

Se a saída do comando acima não retornar nós que fazem parte do seu grupo de nós gerenciado, então não há nós no grupo de nós que tenham o rótulo `for-use-with-emr-containers-managed-endpoint-ng=`*NodeGroupName* do Kubernetes anexado. Nesse caso, siga as etapas abaixo para anexar esse rótulo aos nós do seu grupo de nós.

1. Use o comando a seguir para adicionar o rótulo `for-use-with-emr-containers-managed-endpoint-ng=`*NodeGroupName* do Kubernetes a todos os nós em um grupo de nós gerenciado *NodeGroupName*:

   ```
   kubectl label nodes --selector eks:nodegroup-name=NodeGroupName for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
   ```

2. Verifique se os nós foram rotulados corretamente usando o seguinte comando:

   ```
   kubectl get nodes --show-labels | grep for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
   ```

Um grupo de nós gerenciado deve estar associado a um grupo de segurança do cluster do Amazon EKS, o que geralmente acontece se você criou o cluster e o grupo de nós gerenciado usando `eksctl`. Você pode verificar isso no Console da AWS usando as etapas a seguir.

1. Acesse o cluster no console do Amazon EKS.
2. Acesse a guia de redes do cluster e anote o grupo de segurança do cluster.
3. Acesse a guia de computação do seu cluster e clique no nome do grupo de nós gerenciado.
4. Na guia Detalhes do grupo de nós gerenciado, verifique se o grupo de segurança do cluster anotado anteriormente está listado em Grupos de segurança.

Se o grupo de nós gerenciado não estiver anexado ao grupo de segurança do cluster do Amazon EKS, será necessário anexar a etiqueta `for-use-with-emr-containers-managed-endpoint-sg=`*`ClusterName`*`/`*`NodeGroupName`* ao grupo de segurança do grupo de nós. Use as etapas abaixo para anexar essa etiqueta.

1. Acesse o console do Amazon EC2 e clique em grupos de segurança no painel de navegação à esquerda.
2. Selecione o grupo de segurança do seu grupo de nós gerenciado ao clicar na caixa de seleção.
3. Na guia Etiquetas, adicione a etiqueta `for-use-with-emr-containers-managed-endpoint-sg=`*`ClusterName`*`/`*`NodeGroupName`* usando o botão Gerenciar etiquetas.

Associação de um pod do JEG a um grupo de nós autogerenciado

A propriedade de configuração apresentada a seguir permite especificar o nome de um grupo de nós autogerenciado ou não gerenciado no cluster do Amazon EKS em que o pod do JEG será implantado.

```
//payload
--configuration-overrides '{
      "applicationConfiguration": [
            {
                "classification": "endpoint-configuration",
                "properties": {
                    "self-managed-nodegroup-name": NodeGroupName
                }
            }
        ]
    }'
```

O grupo de nós deve ter o rótulo `for-use-with-emr-containers-managed-endpoint-ng=`*`NodeGroupName`* do Kubernetes anexado a todos os nós que fazem parte do grupo de nós. Para listar todos os nós de um grupo de nós que têm essa etiqueta, use o seguinte comando:

```
kubectl get nodes --show-labels | grep for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
```

Se a saída do comando acima não retornar nós que fazem parte do seu grupo de nós autogerenciado, então não há nós no grupo de nós que tenham o rótulo `for-use-with-emr-containers-managed-endpoint-ng=`*`NodeGroupName`* do Kubernetes anexado. Nesse caso, siga as etapas abaixo para anexar esse rótulo aos nós do seu grupo de nós.

1. Se você criou o grupo de nós autogerenciado usando `eksctl`, use o comando a seguir para adicionar o rótulo `for-use-with-emr-containers-managed-endpoint-ng=`*`NodeGroupName`* do Kubernetes a todos os nós no grupo de nós autogerenciado *`NodeGroupName`* de uma vez só.

   ```
   kubectl label nodes --selector alpha.eksctl.io/nodegroup-name=NodeGroupName for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
   ```

   Caso não tenha usado `eksctl` para criar o grupo de nós autogerenciado, você precisará substituir o seletor no comando acima por um rótulo diferente do Kubernetes que esteja anexado a todos os nós do grupo de nós.
2. Use o seguinte comando para verificar se os nós foram rotulados corretamente:

   ```
   kubectl get nodes --show-labels | grep for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
   ```

O grupo de segurança do grupo de nós autogerenciado deve ter a etiqueta `for-use-with-emr-containers-managed-endpoint-sg=`*`ClusterName`*`/`*`NodeGroupName`* anexada. Use as etapas apresentadas a seguir para anexar a etiqueta ao grupo de segurança do AWS Management Console.

1. Navegue até o console do Amazon EC2. Selecione Grupos de segurança no painel de navegação à esquerda.
2. Selecione a caixa de seleção ao lado do grupo de segurança do seu grupo de nós autogerenciado.

3. Na guia Etiquetas, use o botão Gerenciar etiquetas para adicionar a etiqueta `for-use-with-emr-containers-managed-endpoint-sg=`*`ClusterName`*`/`*`NodeGroupName`*. Substitua *`ClusterName`* e *`NodeGroupName`* com os valores apropriados.

Associação de um pod do JEG a um grupo de nós gerenciado com instâncias sob demanda

Você também pode definir rótulos adicionais, conhecidos como seletores de rótulos do Kubernetes, para especificar limitações ou restrições adicionais à execução de um endpoint interativo em um determinado nó ou grupo de nós. O exemplo a seguir mostra como usar as instâncias sob demanda do Amazon EC2 para um pod do JEG.

```
--configuration-overrides '{
      "applicationConfiguration": [
            {
                "classification": "endpoint-configuration",
                "properties": {
                    "managed-nodegroup-name": NodeGroupName,
                    "node-labels": "eks.amazonaws.com/capacityType:ON_DEMAND"
                }
            }
        ]
    }'
```

> **Note**
>
> Você pode usar a propriedade `node-labels` somente com uma propriedade `managed-nodegroup-name` ou `self-managed-nodegroup-name`.

Opções de configuração do Jupyter Enterprise Gateway (JEG)

O Amazon EMR no EKS usa o Jupyter Enterprise Gateway (JEG) para ativar os endpoints interativos. É possível definir os valores a seguir para as configurações do JEG listadas como permitidas ao criar o endpoint.

- **`RemoteMappingKernelManager.cull_idle_timeout`**: tempo limite em segundos (inteiro), após o qual um kernel é considerado inativo e pronto para ser descartado. Valores iguais ou

inferiores a `0` desativam o descarte. Tempos limite curtos podem resultar no descarte de kernels para usuários com conexões de rede ruins.

- **`RemoteMappingKernelManager.cull_interval`**: o intervalo em segundos (inteiro) no qual verificar se há kernels inativos que excedem o valor do tempo limite de descarte.

Modificação de parâmetros de sessões do PySpark

A partir da versão 6.9.0 do Amazon EMR no EKS, no Amazon EMR Studio é possível ajustar a configuração do Spark associada a uma sessão do PySpark executando o comando mágico `%%configure` na célula de caderno do EMR.

O exemplo a seguir mostra uma carga útil de exemplo que você pode usar para modificar a memória, os núcleos e outras propriedades do driver e do executor do Spark. Para as configurações `conf`, você pode definir qualquer configuração do Spark mencionada na documentação de configuração do Apache Spark.

```
%%configure -f
{
  "driverMemory": "16G",
  "driverCores" 4,
  "executorMemory" : "32G"
  "executorCores": 2,
  "conf": {
     "spark.dynamicAllocation.maxExecutors" : 10,
     "spark.dynamicAllocation.minExecutors": 1
  }
}
```

O exemplo a seguir mostra uma carga útil de exemplo que você pode usar para adicionar arquivos, pyFiles e dependências em JAR a um runtime do Spark.

```
%%configure -f
{
  "files": "s3://test-bucket-emr-eks/sample_file.txt",
  "pyFiles": : "path-to-python-files",
  "jars" : "path-to-jars
}
```

Imagem de kernel personalizada com o endpoint interativo

Para garantir que você tenha as dependências corretas para sua aplicação ao executar as workloads interativas do Amazon EMR Studio, você pode personalizar as imagens do Docker para os endpoints interativos e executar imagens base de kernel personalizadas. Para criar um endpoint interativo e conectá-lo a uma imagem do Docker personalizada, execute as etapas a seguir.

> **Note**
>
> Você pode substituir somente as imagens base. Não é possível adicionar novos tipos de imagens de kernel.

1. Crie e publique uma imagem do Docker personalizada. A imagem base contém o runtime do Spark e os kernels do caderno que são executados com ele. Para criar a imagem, você pode seguir as etapas de 1 a 4 descritas em Como personalizar imagens do Docker. Na etapa 1, o URI da imagem base em seu arquivo do Docker deve usar `notebook-spark` no lugar de `spark`.

   ```
   ECR-registry-account.dkr.ecr.Region.amazonaws.com/notebook-spark/container-image-tag
   ```

   Para obter mais informações sobre como selecionar as Regiões da AWS e as etiquetas de imagem de contêiner, consulte Como selecionar um URI de imagem base.
2. Crie um endpoint interativo que possa ser usado com a imagem personalizada.

   a. Crie um arquivo JSON `custom-image-managed-endpoint.json` com o conteúdo apresentado a seguir. Este exemplo usa a versão 6.9.0 do Amazon EMR.

      Example

      ```
      {
          "name": "endpoint-name",
          "virtualClusterId": "virtual-cluster-id",
          "type": "JUPYTER_ENTERPRISE_GATEWAY",
          "releaseLabel": "emr-6.9.0-latest",
          "executionRoleArn": "execution-role-arn",
          "configurationOverrides": {
              "applicationConfiguration": [
      ```

```
            {
                "classification": "jupyter-kernel-overrides",
                "configurations": [
                    {
                        "classification": "python3",
                        "properties": {
                            "container-image": "123456789012.dkr.ecr.us-
west-2.amazonaws.com/custom-notebook-python:latest"
                        }
                    },
                    {
                        "classification": "spark-python-kubernetes",
                        "properties": {
                            "container-image": "123456789012.dkr.ecr.us-
west-2.amazonaws.com/custom-notebook-spark:latest"
                        }
                    }
                ]
            }
        ]
    }
}
```

b. Crie um endpoint interativo com as configurações especificadas no arquivo JSON, conforme mostrado no exemplo a seguir. Para obter mais informações, consulte Criação de um endpoint interativo com o comando `create-managed-endpoint`.

```
aws emr-containers create-managed-endpoint --cli-input-json custom-image-
managed-endpoint.json
```

3. Conecte-se ao endpoint interativo usando o EMR Studio. Para obter mais informações e etapas a serem concluídas, consulte Connecting from Studio na seção Amazon EMR no EKS da documentação do AWS Workshop Studio.

Monitoramento de endpoints interativos

Com a versão 6.10 e versões posteriores do Amazon EMR no EKS, os endpoints interativos emitem métricas do Amazon CloudWatch para monitorar e solucionar problemas de operações do ciclo de vida do kernel. As métricas são acionadas por clientes interativos, como o EMR Studio ou os cadernos Jupyter de hospedagem própria. Cada uma das operações compatíveis com os endpoints interativos tem métricas associadas a elas. As operações são modeladas como dimensões para

cada métrica, conforme mostrado na tabela abaixo. As métricas emitidas por endpoints interativos ficam visíveis em um namespace personalizado, denominado EMRContainers, em sua conta.

Métrica	Descrição	Unidade
RequestCount	Número cumulativo de solicitações de uma operação processada pelo endpoint interativo.	Contagem
RequestLatency	O horário entre uma solicitação chegar ao endpoint interativo e uma resposta ser enviada pelo endpoint interativo.	Milissegundo
4XXError	Emitido quando uma solicitação de uma operação resulta em um erro 4xx durante o processamento.	Contagem
5XXError	Emitido quando uma solicitação de uma operação resulta em um erro 5Xxx no lado do servidor.	Contagem
KernelLaunchSuccess	Aplicável somente para a operação CreateKernel. Indica o número cumulativo de inicializações de kernel que ocorreram com êxito até e incluindo esta solicitação.	Contagem
KernelLaunchFailure	Aplicável somente para a operação CreateKernel. Indica o número cumulativo de falhas de inicialização de kernel até e incluindo esta solicitação.	Contagem

Cada métrica do endpoint interativo tem as seguintes dimensões anexadas a ela:

- **`ManagedEndpointId`**: identificador para o endpoint interativo.
- **`OperationName`**: a operação acionada pelo cliente interativo.

Os valores possíveis para a dimensão **`OperationName`** são mostrados na seguinte tabela:

`operationName`	Descrição da operação
`CreateKernel`	Solicita que o endpoint interativo inicie um kernel.
`ListKernels`	Solicita que o endpoint interativo liste os kernels que foram iniciados anteriormente usando o mesmo token de sessão.
`GetKernel`	Solicita que o endpoint interativo obtenha detalhes sobre um kernel específico que foi iniciado anteriormente.
`ConnectKernel`	Solicita que o endpoint interativo estabeleça conectividade entre o cliente do caderno e o kernel.
`ConfigureKernel`	Publica `%%configure magic request` em um kernel do PySpark.
`ListKernelSpecs`	Solicita que o endpoint interativo liste as especificações de kernel disponíveis.
`GetKernelSpec`	Solicita que o endpoint interativo obtenha as especificações de kernel de um kernel que foi iniciado anteriormente.
`GetKernelSpecResource`	Solicita que o endpoint interativo obtenha recursos específicos associados às especific ações do kernel que foram iniciadas anteriorm ente.

Exemplos

Para acessar o número total de kernels iniciados para um endpoint interativo em um determinado dia:

1. Selecione o namespace personalizado: `EMRContainers`.
2. Selecione o `ManagedEndpointId` e `OperationName - CreateKernel`.
3. A métrica `RequestCount` com a estatística `SUM` e o período de `1 day` fornecerá todas as solicitações de inicialização do kernel realizadas nas últimas 24 horas.
4. A métrica KernelLaunchSuccess com estatística `SUM` e período de `1 day` fornecerá todas as solicitações de inicialização do kernel que ocorreram com êxito e foram realizadas nas últimas 24 horas.

Para acessar o número de falhas de kernel para um endpoint interativo em um determinado dia:

1. Selecione o namespace personalizado: EMRContainers.
2. Selecione o `ManagedEndpointId` e `OperationName - CreateKernel`.
3. A métrica `KernelLaunchFailure` com a estatística `SUM` e o período de `1 day` fornecerá todas as solicitações de inicialização do kernel com falha realizadas nas últimas 24 horas. Você também pode selecionar as métricas `4XXError` e `5XXError` para saber que tipo de falha na inicialização do kernel ocorreu.

Uso de cadernos Jupyter de hospedagem própria

Você pode hospedar e gerenciar cadernos Jupyter ou JupyterLab em uma instância do Amazon EC2 ou em seu próprio cluster do Amazon EKS como um caderno Jupyter de hospedagem própria. Em seguida, é possível executar workloads interativas com seus cadernos Jupyter de hospedagem própria. As seções apresentadas a seguir descrevem o processo de configuração e de implantação de um caderno Jupyter de hospedagem própria em um cluster do Amazon EKS.

Criação de um caderno Jupyter de hospedagem própria em um cluster do EKS

- Crie um grupo de segurança
- Criação de um endpoint interativo do Amazon EMR no EKS

- Recuperação do URL do servidor de gateway do endpoint interativo
- Recuperação de um token de autenticação para a conexão com o endpoint interativo
- Exemplo: implantação de um caderno JupyterLab
- Exclusão de um caderno Jupyter de hospedagem própria

Crie um grupo de segurança

Antes de poder criar um endpoint interativo e executar um caderno Jupyter ou JupyterLab de hospedagem própria, você deve criar um grupo de segurança para controlar o tráfego entre seu caderno e o endpoint interativo. Para usar o console do Amazon EC2 ou o SDK do Amazon EC2 para criar o grupo de segurança, consulte as etapas em Crie um grupo de segurança no Guia do usuário do Amazon EC2 para instâncias do Linux. Você deve criar o grupo de segurança na VPC em que deseja implantar seu servidor de cadernos.

Para seguir o exemplo deste guia, use a mesma VPC do cluster do Amazon EKS. Se desejar hospedar seu caderno em uma VPC diferente da VPC do cluster do Amazon EKS, pode ser necessário criar uma conexão de emparelhamento entre essas duas VPCs. Para obter as etapas de criação de uma conexão de emparelhamento entre duas VPCs, consulte Criar uma conexão de emparelhamento da VPC no Guia de conceitos básicos da Amazon VPC.

Você precisará do ID do grupo de segurança para criar um endpoint interativo do Amazon EMR no EKS na próxima etapa.

Criação de um endpoint interativo do Amazon EMR no EKS

Após criar o grupo de segurança para o seu caderno, use as etapas fornecidas em Criação de um endpoint interativo para o cluster virtual para criar um endpoint interativo. Você deve fornecer o ID do grupo de segurança criado para o seu caderno em Crie um grupo de segurança.

Insira o ID do grupo de segurança no lugar de *your-notebook-security-group-id* nas seguintes configurações de substituição:

```
--configuration-overrides '{
    "applicationConfiguration": [
        {
            "classification": "endpoint-configuration",
            "properties": {
                "notebook-security-group-id": "your-notebook-security-group-id"
```

```
            }
        }
    ],
    "monitoringConfiguration": {
    ...'
```

Recuperação do URL do servidor de gateway do endpoint interativo

Após criar um endpoint interativo, recupere o URL do servidor de gateway usando o comando `describe-managed-endpoint` na AWS CLI. Você precisa desse URL para conectar seu caderno ao endpoint. O URL do servidor de gateway é um endpoint privado.

```
aws emr-containers describe-managed-endpoint \
--region region \
--virtual-cluster-id virtualClusterId \
--id endpointId
```

Inicialmente, seu endpoint está no estado CREATING. Após alguns minutos, ele passa para o estado ACTIVE. Quando o endpoint estiver ACTIVE, ele estará pronto para uso.

Anote o atributo `serverUrl` que o comando `aws emr-containers describe-managed-endpoint` retorna do endpoint ativo. Você precisará desse URL para conectar seu caderno ao endpoint no momento da implantação do caderno Jupyter ou JupyterLab de hospedagem própria.

Recuperação de um token de autenticação para a conexão com o endpoint interativo

Para se conectar a um endpoint interativo de um caderno Jupyter ou JupyterLab, você deve gerar um token de sessão com a API `GetManagedEndpointSessionCredentials`. O token atua como prova de autenticação para a conexão com o servidor de endpoint interativo.

O comando apresentado a seguir é explicado com mais detalhes com um exemplo de saída abaixo.

```
aws emr-containers get-managed-endpoint-session-credentials \
--endpoint-identifier endpointArn \
--virtual-cluster-identifier virtualClusterArn \
--execution-role-arn executionRoleArn \
--credential-type "TOKEN" \
--duration-in-seconds durationInSeconds \
--region region
```

endpointArn

O ARN do seu endpoint. Você pode descobrir o ARN no resultado de uma chamada `describe-managed-endpoint`.

virtualClusterArn

O ARN do cluster virtual.

executionRoleArn

O ARN do perfil de execução.

durationInSeconds

A duração, em segundos, pela qual o token é válido. A duração padrão é de 15 minutos (`900`) e a duração máxima é de 12 horas (`43200`).

region

A mesma região do seu endpoint.

A saída deve ser semelhante ao exemplo apresentado a seguir. Anote o valor *`session-token`* que você usará ao implantar seu caderno Jupyter ou JupyterLab de hospedagem própria.

```
{
    "id": "credentialsId",
    "credentials": {
        "token": "session-token"
    },
    "expiresAt": "2022-07-05T17:49:38Z"
}
```

Exemplo: implantação de um caderno JupyterLab

Após concluir as etapas acima, você pode tentar este procedimento de exemplo para implantar um caderno JupyterLab no cluster do Amazon EKS com seu endpoint interativo.

1. Crie um namespace para executar o servidor de cadernos.
2. Crie um arquivo localmente, chamado `notebook.yaml`, com o conteúdo apresentado a seguir. O conteúdo para o arquivo é descrito abaixo.

```
apiVersion: v1
```

```
kind: Pod
metadata:
  name: jupyter-notebook
  namespace: namespace
spec:
  containers:
  - name: minimal-notebook
    image: jupyter/all-spark-notebook:lab-3.1.4 # open source image
    ports:
    - containerPort: 8888
    command: ["start-notebook.sh"]
    args: ["--LabApp.token=''"]
    env:
    - name: JUPYTER_ENABLE_LAB
      value: "yes"
    - name: KERNEL_LAUNCH_TIMEOUT
      value: "400"
    - name: JUPYTER_GATEWAY_URL
      value: "serverUrl"
    - name: JUPYTER_GATEWAY_VALIDATE_CERT
      value: "false"
    - name: JUPYTER_GATEWAY_AUTH_TOKEN
      value: "session-token"
```

Se você estiver implantando o caderno Jupyter em um cluster somente do Fargate, rotule o pod do Jupyter com um rótulo `role`, conforme mostrado no seguinte exemplo:

```
...
metadata:
  name: jupyter-notebook
  namespace: default
  labels:
    role: example-role-name-label
spec:
            ...
```

namespace

O namespace do Kubernetes no qual o caderno é implantado.

serverUrl

O atributo `serverUrl` que o comando `describe-managed-endpoint` retornou em Recuperação do URL do servidor de gateway do endpoint interativo .

session-token

O atributo `session-token` que o comando `get-managed-endpoint-session-credentials` retornou em Recuperação de um token de autenticação para a conexão com o endpoint interativo.

KERNEL_LAUNCH_TIMEOUT

A quantidade de tempo, em segundos, que o endpoint interativo aguarda até que o kernel chegue ao estado RUNNING. Garanta tempo suficiente para a inicialização do kernel ser concluída ao definir o tempo limite de inicialização do kernel para um valor apropriado (máximo de 400 segundos).

KERNEL_EXTRA_SPARK_OPTS

Como opção, você pode transferir configurações adicionais do Spark para os kernels do Spark. Defina essa variável de ambiente com os valores da propriedade de configuração do Spark, conforme mostrado no seguinte exemplo:

```
- name: KERNEL_EXTRA_SPARK_OPTS
  value: "--conf spark.driver.cores=2
          --conf spark.driver.memory=2G
          --conf spark.executor.instances=2
          --conf spark.executor.cores=2
          --conf spark.executor.memory=2G
          --conf spark.dynamicAllocation.enabled=true
          --conf spark.dynamicAllocation.shuffleTracking.enabled=true
          --conf spark.dynamicAllocation.minExecutors=1
          --conf spark.dynamicAllocation.maxExecutors=5
          --conf spark.dynamicAllocation.initialExecutors=1
          "
```

3. Implante a especificação do pod no cluster do Amazon EKS:

```
kubectl apply -f notebook.yaml -n namespace
```

Isso iniciará um caderno JupyterLab mínimo conectado ao endpoint interativo do Amazon EMR no EKS. Aguarde até que o pod esteja RUNNING. Você pode verificar seu status com o seguinte comando:

```
kubectl get pod jupyter-notebook -n namespace
```

Quando o pod estiver pronto, o comando `get pod` retornará uma saída semelhante a esta:

```
NAME              READY   STATUS    RESTARTS   AGE
jupyter-notebook  1/1     Running   0          46s
```

4. Anexe o grupo de segurança do caderno ao nó em que o caderno está programado.

   a. Primeiro, identifique o nó em que o pod `jupyter-notebook` está programado com o comando `describe pod`.

   ```
   kubectl describe pod jupyter-notebook -n namespace
   ```

   b. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.

   c. Navegue até a guia Computação do cluster do Amazon EKS e selecione o nó identificado pelo comando `describe pod`. Selecione o ID da instância para o nó.

   d. No menu Ações, selecione Segurança > Alterar grupos de segurança para anexar o grupo de segurança que você criou em Crie um grupo de segurança.

   e. Se você estiver implantando um pod de caderno Jupyter no AWS Fargate, crie um `SecurityGroupPolicy` para aplicar ao pod de caderno Jupyter com o rótulo de perfil:

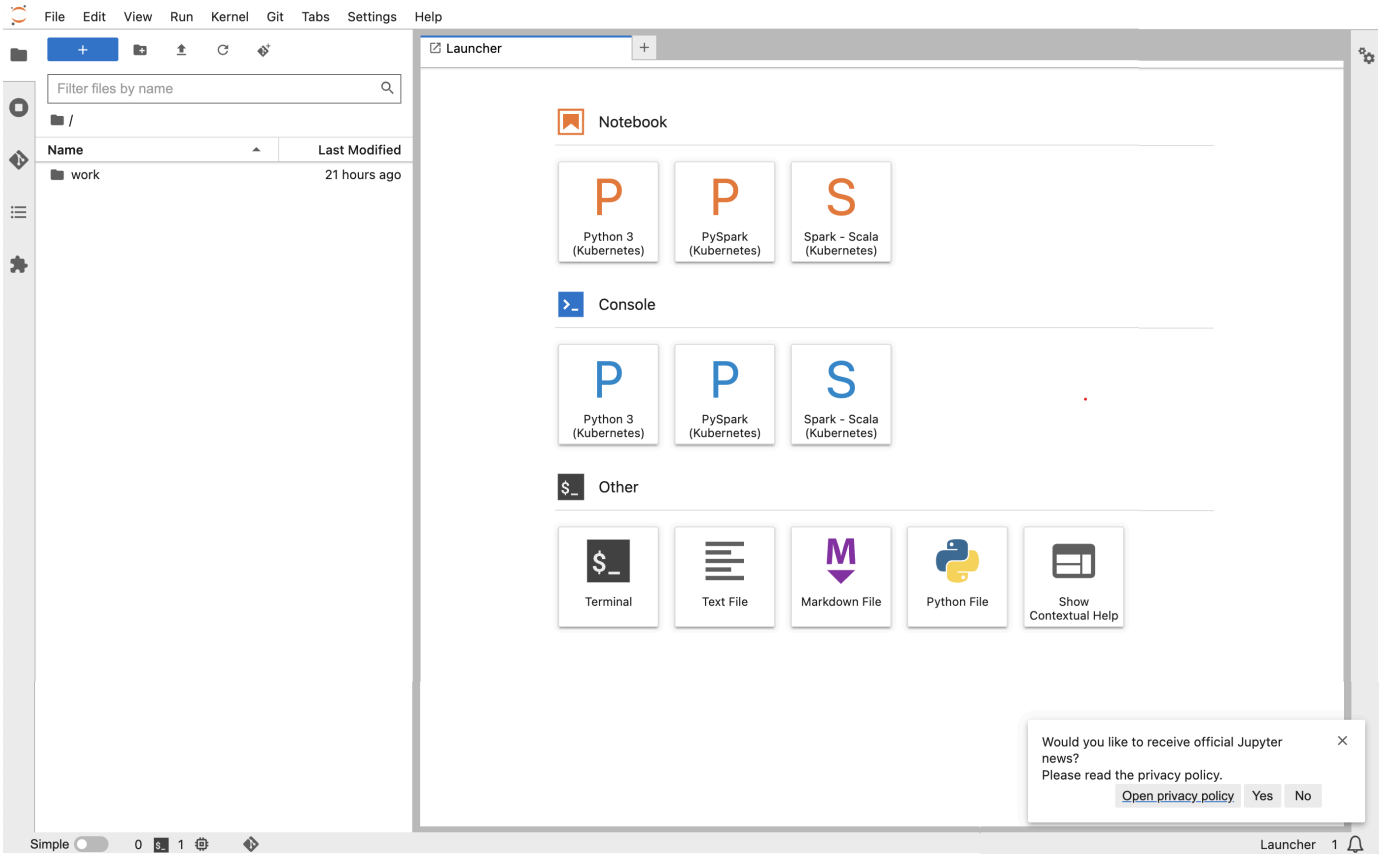
   ```
   cat >my-security-group-policy.yaml <<EOF
   apiVersion: vpcresources.k8s.aws/v1beta1
   kind: SecurityGroupPolicy
   metadata:
     name: example-security-group-policy-name
     namespace: default
   spec:
     podSelector:
       matchLabels:
         role: example-role-name-label
     securityGroups:
       groupIds:
         - your-notebook-security-group-id
   ```

```
EOF
```

5. Agora, realize o encaminhamento de porta para que você possa acessar localmente a interface do JupyterLab:

```
kubectl port-forward jupyter-notebook 8888:8888 -n namespace
```

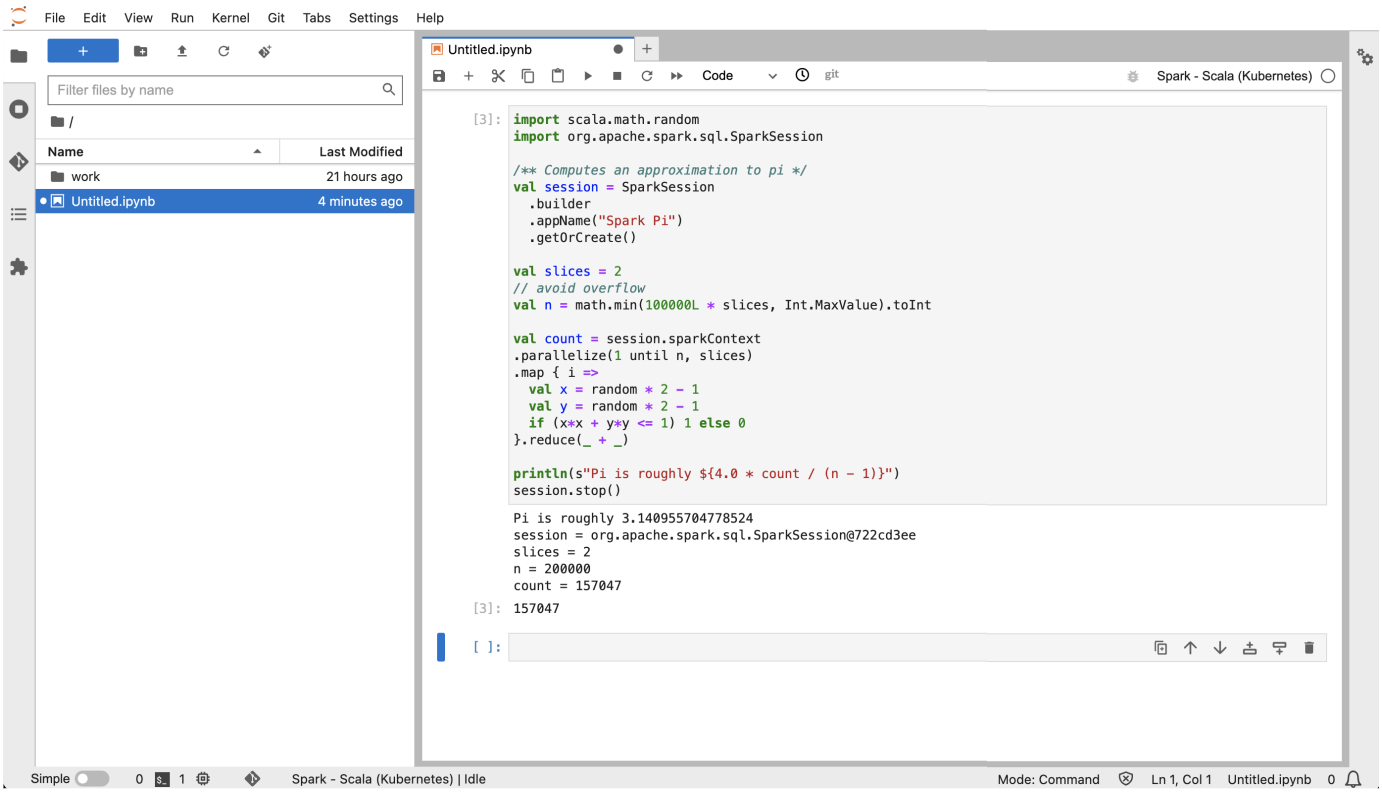
Quando estiver em execução, navegue até seu navegador local e acesse `localhost:8888` para visualizar a interface do JupyterLab:



6. No JupyterLab, crie um novo caderno do Scala. Confira um exemplo de trecho de código que você pode executar para aproximar o valor de Pi:

```
import scala.math.random
import org.apache.spark.sql.SparkSession

/** Computes an approximation to pi */
val session = SparkSession
  .builder
  .appName("Spark Pi")
  .getOrCreate()
```

```
val slices = 2
// avoid overflow
val n = math.min(100000L * slices, Int.MaxValue).toInt

val count = session.sparkContext
.parallelize(1 until n, slices)
.map { i =>
  val x = random * 2 - 1
  val y = random * 2 - 1
  if (x*x + y*y <= 1) 1 else 0
}.reduce(_ + _)

println(s"Pi is roughly ${4.0 * count / (n - 1)}")
session.stop()
```



Exclusão de um caderno Jupyter de hospedagem própria

Quando estiver com tudo pronto para excluir seu caderno de hospedagem própria, você também poderá excluir o endpoint interativo e o grupo de segurança. Execute as ações na seguinte ordem:

1. Use o seguinte comando para excluir o pod `jupyter-notebook`:

```
kubectl delete pod jupyter-notebook -n namespace
```

2. Em seguida, exclua o endpoint interativo com o comando `delete-managed-endpoint`. Para obter as etapas para a exclusão de um endpoint interativo, consulte Exclusão de um endpoint interativo. Inicialmente, seu endpoint estará no estado TERMINATING. Depois que todos os recursos forem limpos, ele transitará para o estado TERMINATED.
3. Caso não planeje usar o grupo de segurança de cadernos criado em Crie um grupo de segurança para outras implantações de caderno Jupyter, você poderá excluí-lo. Consulte Excluir um grupo de segurança no Guia do usuário do Amazon EC2 para obter mais informações.

Outras operações em um endpoint interativo

Este tópico aborda as operações com suporte em um endpoint interativo diferente de `create-managed-endpoint`.

Busca de detalhes do endpoint interativo

Após criar um endpoint interativo, você poderá recuperar os detalhes usando o comando `describe-managed-endpoint` da AWS CLI. Insira seus próprios valores para *managed-endpoint-id*, *virtual-cluster-id* e *region*:

```
aws emr-containers describe-managed-endpoint --id managed-endpoint-id \
 --virtual-cluster-id virtual-cluster-id --region region
```

A saída é semelhante à apresentada a seguir, com o endpoint especificado, como o ARN, o ID e o nome.

```
{
   "id": "as3ys2xxxxxxx",
   "name": "endpoint-name",
    "arn": "arn:aws:emr-containers:us-east-1:1828xxxxxxxx:/virtualclusters/
lbhl6kwwyoxxxxxxxxxxxxxx/endpoints/as3ysxxxxxxxx",
    "virtualClusterId": "lbhl6kwwyoxxxxxxxxxxxxxx",
    "type": "JUPYTER_ENTERPRISE_GATEWAY",
    "state": "ACTIVE",
    "releaseLabel": "emr-6.9.0-latest",
   "executionRoleArn": "arn:aws:iam::1828xxxxxxxx:role/RoleName",
```

```
    "certificateAuthority": {
        "certificateArn": "arn:aws:acm:us-east-1:1828xxxxxxxx:certificate/zzzzzzzz-
e59b-4ed0-aaaa-bbbbbbbbbbbb",
        "certificateData": "certificate-data"
    },
    "configurationOverrides": {
        "applicationConfiguration": [
            {
                "classification": "spark-defaults",
                "properties": {
                    "spark.driver.memory": "8G"
                }
            }
        ],
        "monitoringConfiguration": {
            "persistentAppUI": "ENABLED",
            "cloudWatchMonitoringConfiguration": {
                "logGroupName": "log-group-name",
                "logStreamNamePrefix": "log-stream-name-prefix"
            },
            "s3MonitoringConfiguration": {
                "logUri": "s3-bucket-name"
            }
        }
    },
   "serverUrl": "https://internal-k8s-namespace-ingressa-aaaaaaaaaa-
zzzzzzzzzz.us-east-1.elb.amazonaws.com:18888 (https://internal-k8s-nspluto-
ingressa-51e860abbd-1620715833.us-east-1.elb.amazonaws.com:18888/)",
    "createdAt": "2022-09-19T12:37:49+00:00",
    "securityGroup": "sg-aaaaaaaaaaaaaa",
    "subnetIds": [
        "subnet-11111111111",
        "subnet-22222222222",
        "subnet-33333333333"
    ],
    "stateDetails": "Endpoint created successfully. It took 3 Minutes 15 Seconds",
    "tags": {}
 }
```

Listagem de todos os endpoints interativos associados a um cluster virtual

Use o comando `list-managed-endpoints` da AWS CLI para buscar uma lista com todos os endpoints interativos associados a um cluster virtual especificado. Substitua `virtual-cluster-id` pelo ID do seu cluster virtual.

```
aws emr-containers list-managed-endpoints --virtual-cluster-id virtual-cluster-id
```

A saída do comando `list-managed-endpoint` é mostrada abaixo:

```
{
    "endpoints": [{
        "id": "as3ys2xxxxxxx",
        "name": "endpoint-name",
        "arn": "arn:aws:emr-containers:us-east-1:1828xxxxxxxx:/virtualclusters/
lbhl6kwwyoxxxxxxxxxxxxxxx/endpoints/as3ysxxxxxxxx",
        "virtualClusterId": "lbhl6kwwyoxxxxxxxxxxxxxxx",
        "type": "JUPYTER_ENTERPRISE_GATEWAY",
        "state": "ACTIVE",
        "releaseLabel": "emr-6.9.0-latest",
        "executionRoleArn": "arn:aws:iam::1828xxxxxxxx:role/RoleName",
        "certificateAuthority": {
            "certificateArn": "arn:aws:acm:us-east-1:1828xxxxxxxx:certificate/zzzzzzzz-
e59b-4ed0-aaaa-bbbbbbbbbbbb",
            "certificateData": "certificate-data"
        },
        "configurationOverrides": {
            "applicationConfiguration": [{
                "classification": "spark-defaults",
                "properties": {
                    "spark.driver.memory": "8G"
                }
            }],
            "monitoringConfiguration": {
                "persistentAppUI": "ENABLED",
                "cloudWatchMonitoringConfiguration": {
                    "logGroupName": "log-group-name",
                    "logStreamNamePrefix": "log-stream-name-prefix"
                },
                "s3MonitoringConfiguration": {
                    "logUri": "s3-bucket-name"
                }
```

```
            }
        },
        "serverUrl": "https://internal-k8s-namespace-ingressa-aaaaaaaaaa-
zzzzzzzzzz.us-east-1.elb.amazonaws.com:18888 (https://internal-k8s-nspluto-
ingressa-51e860abbd-1620715833.us-east-1.elb.amazonaws.com:18888/)",
        "createdAt": "2022-09-19T12:37:49+00:00",
        "securityGroup": "sg-aaaaaaaaaaaaa",
        "subnetIds": [
            "subnet-11111111111",
            "subnet-22222222222",
            "subnet-33333333333"
        ],
        "stateDetails": "Endpoint created successfully. It took 3 Minutes 15 Seconds",
        "tags": {}
    }]
}
```

Exclusão de um endpoint interativo

Para excluir um endpoint interativo associado a um cluster virtual do Amazon EMR no EKS, use o comando `delete-managed-endpoint` da AWS CLI. Quando você exclui um endpoint interativo, o Amazon EMR no EKS remove os grupos de segurança padrão criados para esse endpoint.

Especifique valores para os seguintes parâmetros do comando:

- --id: o identificador do endpoint interativo que você deseja excluir.
- --virtual-cluster-id: o identificador do cluster virtual associado ao endpoint interativo que você deseja excluir. Esse é o mesmo ID do cluster virtual que foi especificado quando o endpoint interativo foi criado.

```
aws emr-containers delete-managed-endpoint --id managed-endpoint-id --virtual-cluster-
id virtual-cluster-id
```

O comando retorna uma saída semelhante à seguinte para confirmar que você excluiu o endpoint interativo:

```
{
    "id":"8gai4l4exxxxx",
    "virtualClusterId":"0b0qvauoy3ch1nqodxxxxxxxx"
}
```

Monitorar trabalhos

Tópicos

- Monitoramento de trabalhos com o Amazon CloudWatch Events
- Automatização do Amazon EMR no EKS com o CloudWatch Events
- Exemplo: configuração de uma regra que invoque o Lambda
- Monitoramento do pod de drivers do trabalho com uma política de repetição usando o Amazon CloudWatch Events

Monitoramento de trabalhos com o Amazon CloudWatch Events

O Amazon EMR no EKS emite eventos quando o estado de uma execução de trabalho é alterado. Cada evento fornece informações, como a data e o horário em que o evento ocorreu, em conjunto com mais detalhes sobre o evento, como o ID do cluster virtual e o ID de uma execução de trabalho que foi afetada.

É possível usar os eventos para rastrear a atividade e a integridade de trabalhos executados em um cluster virtual. Você também pode usar o Amazon CloudWatch Events para definir uma ação a ser tomada quando uma execução de trabalho gerar um evento que corresponda a um padrão especificado por você. Os eventos são úteis para monitorar uma ocorrência específica durante o ciclo de vida de uma execução de trabalho. Por exemplo, é possível monitorar quando uma execução de trabalho sofre alteração de estado de `submitted` para `running`. Para obter mais informações sobre o CloudWatch Events, consulte o Guia do usuário do Amazon CloudWatch Events.

A tabela apresentada a seguir lista os eventos do Amazon EMR no EKS em conjunto com o estado ou a alteração de estado que o evento indica, a severidade do evento e as mensagens do evento. Cada evento é representado como um objeto JSON que é enviado automaticamente a um stream de evento. O objeto JSON inclui mais detalhes sobre o evento. O objeto JSON é particularmente importante quando você configura as regras para o processamento de eventos usando o CloudWatch Events, pois as regras buscam corresponder aos padrões no objeto JSON. Para obter mais informações, consulte Events and Event Patterns e os eventos do Amazon EMR no EKS no Guia do usuário do Amazon CloudWatch Events.

Eventos de alteração de estado em execuções de trabalhos

Estado	Severidade	Message
SUBMITTED	INFO	A execução de trabalho *JobRunId* (*JobRunName*) foi enviada com êxito ao cluster virtual *VirtualClusterId* às *Horário* UTC.
RUNNING (Em execução)	INFO	A execução de trabalho *JobRunId* (*JobRunName*) no cluster virtual *VirtualClusterId* começou a ser executada às *Horário*.
COMPLETED	INFO	A execução de trabalho *JobRunId* (*JobRunName*) no cluster virtual *VirtualClusterId* foi concluída às *Horário*. A execução de trabalho começou a ser executada às *Horário* e demorou *Número* minutos para ser concluída.
CANCELADO	WARN	A solicitação de cancelamento teve êxito para a execução de trabalho *JobRunId* (*JobRunName*) no cluster virtual *VirtualClusterId* às *Horário* e a execução de trabalho está cancelada, no momento.
FAILED (COM FALHA)	ERROR	A execução de trabalho *JobRunId* (*JobRunName*) no cluster virtual *VirtualClusterId* falhou às *Horário*.

Automatização do Amazon EMR no EKS com o CloudWatch Events

Você pode usar o Amazon CloudWatch Events para automatizar os serviços da AWS para responder a eventos do sistema, como problemas de disponibilidade de aplicações ou alterações de recursos. Os eventos dos produtos da AWS são entregues ao CloudWatch Events quase em tempo real.

Você pode criar regras simples para indicar quais eventos são de seu interesse, e quais ações automatizadas devem ser tomadas quando um evento corresponder a uma regra. As ações que podem ser automaticamente acionadas incluem as seguintes:

- Como invocar uma função do AWS Lambda
- Invocação do Run Command do Amazon EC2
- Retransmissão do evento para o Amazon Kinesis Data Streams
- Ativação da máquina de estado do AWS Step Functions
- Notificação de um tópico do Amazon Simple Notification Service (SNS) ou de uma fila do Amazon Simple Queue Service (SQS)

Alguns exemplos de uso do CloudWatch Events com o Amazon EMR no EKS incluem o seguinte:

- Ativação de uma função do Lambda quando uma execução de trabalho tiver êxito.
- Notificação de um tópico do Amazon SNS quando uma execução de trabalho falhar.

Os eventos do CloudWatch Events para "`detail-type:`" "`EMR Job Run State Change`" são gerados pelo Amazon EMR no EKS para as alterações de estado `SUBMITTED`, `RUNNING`, `CANCELLED`, `FAILED` e `COMPLETED`.

Exemplo: configuração de uma regra que invoque o Lambda

Use as etapas apresentadas a seguir para configurar uma regra do CloudWatch Events que invoca o Lambda quando há um evento de "alteração de estado em execuções de trabalhos do EMR".

```
aws events put-rule \
--name cwe-test \
--event-pattern '{"detail-type": ["EMR Job Run State Change"]}'
```

Adicione a função do Lambda de sua propriedade como um novo destino e conceda permissão ao CloudWatch Events para invocar a função do Lambda, como apresentado a seguir. Substitua *123456789012* pelo ID da sua conta.

```
aws events put-targets \
--rule cwe-test \
--targets Id=1,Arn=arn:aws:lambda:us-east-1:123456789012:function:MyFunction
```

```
aws lambda add-permission \
--function-name MyFunction \
--statement-id MyId \
--action 'lambda:InvokeFunction' \
--principal events.amazonaws.com
```

> **Note**
>
> Não é possível escrever um programa que dependa da ordem ou da existência de eventos de notificação, pois eles podem estar fora de sequência ou ausentes. Os eventos são emitidos com base no melhor esforço.

Monitoramento do pod de drivers do trabalho com uma política de repetição usando o Amazon CloudWatch Events

Ao usar eventos do CloudWatch, você pode monitorar pods de drivers que foram criados em trabalhos que têm políticas de repetição. Para obter mais informações, consulte Monitoramento de um trabalho com uma política de repetição neste guia.

Gerenciamento de clusters virtuais

Um cluster virtual corresponde a um namespace do Kubernetes no qual o Amazon EMR está registrado. Você pode criar, descrever, listar e excluir clusters virtuais. Eles não consomem quaisquer recursos adicionais em seu sistema. Um único cluster virtual mapeia para um único namespace do Kubernetes. Dado esse relacionamento, você pode modelar clusters virtuais da mesma forma que modela namespaces Kubernetes para atender aos seus requisitos. Confira os possíveis casos de uso na documentação de visão geral dos conceitos do Kubernetes.

Para registrar o Amazon EMR com um namespace do Kubernetes em um cluster do Amazon EKS, você precisa do nome do cluster do EKS e do namespace que foi configurado para executar sua workload. Esses clusters registrados no Amazon EMR são chamados de clusters virtuais porque não gerenciam computação ou armazenamento físicos, mas direcionam para um namespace do Kubernetes no qual sua workload está programada.

> **Note**
>
> Antes de criar um cluster virtual, você deve concluir as etapas de 1 a 8 em Configuração do Amazon EMR no EKS.

Tópicos

- Criação de um cluster virtual
- Listagem de clusters virtuais
- Descrição de um cluster virtual
- Exclusão de um cluster virtual
- Estados de um cluster virtual

Criação de um cluster virtual

Execute o comando apresentado a seguir para criar um cluster virtual ao registrar o Amazon EMR com um namespace em um cluster do EKS. Substitua *virtual_cluster_name* por um nome fornecido por você para o cluster virtual. Substitua *eks_cluster_name* pelo nome do cluster do EKS. Substitua *namespace_name* pelo namespace com o qual você deseja registrar o Amazon EMR.

```
aws emr-containers create-virtual-cluster \
--name virtual_cluster_name \
--container-provider '{
    "id": "eks_cluster_name",
    "type": "EKS",
    "info": {
        "eksInfo": {
            "namespace": "namespace_name"
        }
    }
}'
```

Como alternativa, você pode criar um arquivo JSON que inclua os parâmetros obrigatórios para o cluster virtual, como demonstra o exemplo a seguir.

```
{
    "name": "virtual_cluster_name",
    "containerProvider": {
        "type": "EKS",
        "id": "eks_cluster_name",
        "info": {
            "eksInfo": {
                "namespace": "namespace_name"
            }
        }
    }
}
```

Em seguida, execute o comando `create-virtual-cluster` apresentado a seguir com o caminho para o arquivo JSON.

```
aws emr-containers create-virtual-cluster \
--cli-input-json file://./create-virtual-cluster-request.json
```

> **Note**
>
> Para validar a criação com êxito de um cluster virtual, visualize o status dos clusters virtuais ao executar o comando `list-virtual-clusters` ou ao acessar a página Clusters virtuais no console do Amazon EMR.

Listagem de clusters virtuais

Execute o comando apresentado a seguir para visualizar o status dos clusters virtuais.

```
aws emr-containers list-virtual-clusters
```

Descrição de um cluster virtual

Execute o comando apresentado a seguir para obter mais detalhes sobre um cluster virtual, como o namespace, o status e a data de registro. Substitua *123456* pelo ID do seu cluster virtual.

```
aws emr-containers describe-virtual-cluster --id 123456
```

Exclusão de um cluster virtual

Execute o comando apresentado a seguir para excluir um cluster virtual. Substitua *123456* pelo ID do seu cluster virtual.

```
aws emr-containers delete-virtual-cluster --id 123456
```

Estados de um cluster virtual

A tabela a seguir descreve os quatro estados possíveis de um cluster virtual.

State	Descrição
RUNNING	O cluster virtual está no estado RUNNING.
TERMINATING	O encerramento solicitado para o cluster virtual está em andamento.
TERMINATED	O encerramento solicitado foi concluído.
ARRESTED	O encerramento solicitado falhou devido a permissões insuficientes.

Tutoriais para o Amazon EMR no EKS

Esta seção descreve casos de uso comuns para quando você trabalha com aplicações do Amazon EMR no EKS.

Tópicos

- Uso do Delta Lake com o Amazon EMR no EKS
- Uso do Apache Iceberg com o Amazon EMR no EKS
- Uso do acelerador RAPIDS para Apache Spark com o Amazon EMR no EKS
- Uso da integração do Amazon Redshift para Apache Spark no Amazon EMR no EKS
- Uso do Volcano como um programador personalizado para Apache Spark no Amazon EMR no EKS
- Uso do YuniKorn como um programador personalizado para Apache Spark no Amazon EMR no EKS

Uso do Delta Lake com o Amazon EMR no EKS

Para usar o Delta Lake com aplicações do Amazon EMR no EKS

1. Ao iniciar uma execução de trabalho para enviar um trabalho do Spark na configuração da aplicação, inclua os arquivos JAR do Delta Lake:

```
--job-driver '{"sparkSubmitJobDriver" : {
      "sparkSubmitParameters" : "--jars local:///usr/share/aws/delta/lib/delta-
core.jar,local:///usr/share/aws/delta/lib/delta-storage.jar,local:///usr/share/aws/
delta/lib/delta-storage-s3-dynamodb.jar"}}'
```

2. Inclua a configuração adicional do Delta Lake e use o Catálogo de Dados do AWS Glue como seu metastore.

```
--configuration-overrides '{
        "applicationConfiguration": [
        {
          "classification" : "spark-defaults",
          "properties" : {
            "spark.sql.extensions" : "io.delta.sql.DeltaSparkSessionExtension",
```

```
 "spark.sql.catalog.spark_catalog":"org.apache.spark.sql.delta.catalog.DeltaCatalog",
"spark.hadoop.hive.metastore.client.factory.class":"com.amazonaws.glue.catalog.metastore.AW
          }
       }]}'
```

Uso do Apache Iceberg com o Amazon EMR no EKS

Usar o Apache Iceberg com as aplicações do Amazon EMR no EKS

1. Ao iniciar uma execução de trabalho para enviar um trabalho do Spark na configuração da aplicação, inclua o arquivo JAR de runtime do Iceberg para o Spark:

```
--job-driver '{"sparkSubmitJobDriver" : {"sparkSubmitParameters" : "--jars
 local:///usr/share/aws/iceberg/lib/iceberg-spark3-runtime.jar"}}'
```

2. Inclua a configuração adicional do Iceberg:

```
--configuration-overrides '{
    "applicationConfiguration": [
    "classification" : "spark-defaults",
    "properties" : {
        "spark.sql.catalog.dev.warehouse" : "s3://DOC-EXAMPLE-BUCKET/EXAMPLE-
PREFIX/ ",
        "spark.sql.extensions ":"
 org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions ",
        "spark.sql.catalog.dev" : "org.apache.iceberg.spark.SparkCatalog",
        "spark.sql.catalog.dev.catalog-impl" :
 "org.apache.iceberg.aws.glue.GlueCatalog",
        "spark.sql.catalog.dev.io-impl": "org.apache.iceberg.aws.s3.S3FileIO"
        }
    ]
}'
```

Para saber mais sobre as versões de liberação do Apache Iceberg para o EMR, consulte Iceberg release history.

Uso do acelerador RAPIDS para Apache Spark com o Amazon EMR no EKS

Com o Amazon EMR no EKS, você pode executar trabalhos para o acelerador RAPIDS da Nvidia para Apache Spark. Este tutorial aborda como executar trabalhos do Spark usando o RAPIDS em tipos de instância de unidade de processamento gráfico (GPU) do EC2. O tutorial usa as seguintes versões:

- Amazon EMR no EKS na versão de liberação 6.9.0 e posteriores
- Apache Spark 3.x

Você pode acelerar o Spark com os tipos de instância de GPU do Amazon EC2 usando o plug-in do acelerador RAPIDS da Nvidia para Apache Spark. Ao usar essas tecnologias em conjunto, você acelera os pipelines de ciência de dados sem a necessidade de realizar alterações no código. Isso reduz o tempo de execução necessário para o processamento de dados e para o treinamento de modelos. Ao fazer mais em menos tempo, você gasta menos com o custo da infraestrutura.

Antes de começar, certifique-se de ter os recursos apresentados a seguir.

- Cluster virtual do Amazon EMR no EKS
- Cluster do Amazon EKS com um grupo de nós habilitado para GPU

Um cluster virtual do Amazon EKS corresponde a um manipulador registrado para o namespace do Kubernetes em um cluster do Amazon EKS e é gerenciado pelo Amazon EMR no EKS. O manipulador permite que o Amazon EMR use o namespace do Kubernetes como destino para a execução de trabalhos. Para obter mais informações sobre como configurar um cluster virtual, consulte Configuração do Amazon EMR no EKS neste guia.

Você deve configurar o cluster virtual do Amazon EKS com um grupo de nós que tenha instâncias de GPU. Você deve configurar os nós com um plug-in para dispositivos da Nvidia. Consulte Grupos de nós gerenciados para saber mais.

Para configurar o cluster do Amazon EKS para adicionar grupos de nós habilitados para GPU, execute o seguinte procedimento:

Adicionar grupos de nós habilitados para GPU

1. Crie um grupo de nós habilitado para GPU com o comando create-nodegroup apresentado a seguir. Certifique-se de realizar a substituição com os parâmetros corretos para o cluster do Amazon EKS. Use um tipo de instância compatível com RAPIDS para Spark, como P4, P3, G5 ou G4dn.

```
aws eks create-nodegroup \
 --cluster-name EKS_CLUSTER_NAME \
 --nodegroup-name NODEGROUP_NAME \
 --scaling-config minSize=0,maxSize=5,desiredSize=2 CHOOSE_APPROPRIATELY \
 --ami-type AL2_x86_64_GPU \
 --node-role NODE_ROLE \
 --subnets SUBNETS_SPACE_DELIMITED  \
 --remote-access ec2SshKey= SSH_KEY \
 --instance-types GPU_INSTANCE_TYPE \
 --disk-size DISK_SIZE \
 --region AWS_REGION
```

2. Instale o plug-in para dispositivos da Nvidia em seu cluster com a finalidade de emitir o número de GPUs em cada nó do cluster e de executar contêineres habilitados para GPU em seu cluster. Execute o seguinte código para instalar o plug-in:

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v0.9.0/
nvidia-device-plugin.yml
```

3. Para validar quantas GPUs estão disponíveis em cada nó do seu cluster, execute o seguinte comando:

```
kubectl get nodes  "-o=custom-
columns=NAME:.metadata.name,GPU:.status.allocatable.nvidia\.com/gpu"
```

Executar um trabalho do RAPIDS para Spark

1. Envie um trabalho do RAPIDS para Spark ao cluster do Amazon EMR no EKS. O código a seguir apresenta um exemplo de comando para iniciar o trabalho. Na primeira vez em que você executar o trabalho, poderá demorar alguns minutos para fazer download da imagem e armazená-la em cache no nó.

```
aws emr-containers start-job-run \
```

```
--virtual-cluster-id VIRTUAL_CLUSTER_ID \
--execution-role-arn JOB_EXECUTION_ROLE \
--release-label emr-6.9.0-spark-rapids-latest \
--job-driver '{"sparkSubmitJobDriver": {"entryPoint": "local:///usr/lib/
spark/examples/jars/spark-examples.jar","entryPointArguments":  ["10000"],
 "sparkSubmitParameters":"--class org.apache.spark.examples.SparkPi "}}' \
---configuration-overrides '{"applicationConfiguration": [{"classification":
 "spark-defaults","properties": {"spark.executor.instances":
 "2","spark.executor.memory": "2G"}}],"monitoringConfiguration":
 {"cloudWatchMonitoringConfiguration": {"logGroupName": "LOG_GROUP
 _NAME"},"s3MonitoringConfiguration": {"logUri": "LOG_GROUP_STREAM"}}}'
```

2. Para validar se o acelerador RAPIDS para Spark está habilitado, verifique os logs do driver do Spark. Esses logs são armazenados no CloudWatch ou no local do S3 que você especifica ao executar o comando `start-job-run`. O seguinte exemplo apresenta como geralmente é a aparência das linhas de log:

```
22/11/15 00:12:44 INFO RapidsPluginUtils: RAPIDS Accelerator build:
 {version=22.08.0-amzn-0, user=release, url=, date=2022-11-03T03:32:45Z, revision=,
 cudf_version=22.08.0, branch=}
22/11/15 00:12:44 INFO RapidsPluginUtils: RAPIDS Accelerator JNI build:
 {version=22.08.0, user=, url=https://github.com/NVIDIA/spark-rapids-jni.git,
 date=2022-08-18T04:14:34Z, revision=a1b23cd_sample, branch=HEAD}
22/11/15 00:12:44 INFO RapidsPluginUtils: cudf build: {version=22.08.0,
 user=, url=https://github.com/rapidsai/cudf.git, date=2022-08-18T04:14:34Z,
 revision=a1b23ce_sample, branch=HEAD}
22/11/15 00:12:44 WARN RapidsPluginUtils: RAPIDS Accelerator 22.08.0-amzn-0 using
 cudf 22.08.0.
22/11/15 00:12:44 WARN RapidsPluginUtils:
 spark.rapids.sql.multiThreadedRead.numThreads is set to 20.
22/11/15 00:12:44 WARN RapidsPluginUtils: RAPIDS Accelerator is enabled, to disable
 GPU support set `spark.rapids.sql.enabled` to false.
22/11/15 00:12:44 WARN RapidsPluginUtils: spark.rapids.sql.explain is set to
 `NOT_ON_GPU`. Set it to 'NONE' to suppress the diagnostics logging about the query
 placement on the GPU.
```

3. Para visualizar as operações que serão executadas em uma GPU, execute as etapas apresentadas a seguir para ativar o registro em log adicional. Observe a configuração "`spark.rapids.sql.explain : ALL`".

```
aws emr-containers start-job-run \
--virtual-cluster-id VIRTUAL_CLUSTER_ID \
```

```
--execution-role-arn JOB_EXECUTION_ROLE \
--release-label emr-6.9.0-spark-rapids-latest \
--job-driver '{"sparkSubmitJobDriver": {"entryPoint": "local:///usr/lib/
spark/examples/jars/spark-examples.jar","entryPointArguments":  ["10000"],
 "sparkSubmitParameters":"--class org.apache.spark.examples.SparkPi "}}' \
---configuration-overrides '{"applicationConfiguration":
 [{"classification": "spark-defaults","properties":
 {"spark.rapids.sql.explain":"ALL","spark.executor.instances":
 "2","spark.executor.memory": "2G"}}],"monitoringConfiguration":
 {"cloudWatchMonitoringConfiguration": {"logGroupName":
 "LOG_GROUP_NAME"},"s3MonitoringConfiguration": {"logUri": "LOG_GROUP_STREAM"}}}'
```

O comando anterior é um exemplo de trabalho que usa a GPU. Sua saída seria semelhante ao exemplo abaixo. Consulte esta legenda para obter ajuda para compreender a saída:

- *: marca uma operação que funciona em uma GPU.
- !: marca uma operação que não pode ser executada em uma GPU.
- @: marca uma operação que funciona em uma GPU, mas não será executada porque está em um plano que não pode ser executado em uma GPU.

```
22/11/15 01:22:58 INFO GpuOverrides: Plan conversion to the GPU took 118.64 ms
22/11/15 01:22:58 INFO GpuOverrides: Plan conversion to the GPU took 4.20 ms
22/11/15 01:22:58 INFO GpuOverrides: GPU plan transition optimization took 8.37 ms
22/11/15 01:22:59 WARN GpuOverrides:
   *Exec <ProjectExec> will run on GPU
     *Expression <Alias> substring(cast(date#149 as string), 0, 7) AS month#310
will run on GPU
       *Expression <Substring> substring(cast(date#149 as string), 0, 7) will run
on GPU
         *Expression <Cast> cast(date#149 as string) will run on GPU
     *Exec <SortExec> will run on GPU
       *Expression <SortOrder> date#149 ASC NULLS FIRST will run on GPU
       *Exec <ShuffleExchangeExec> will run on GPU
         *Partitioning <RangePartitioning> will run on GPU
           *Expression <SortOrder> date#149 ASC NULLS FIRST will run on GPU
         *Exec <UnionExec> will run on GPU
           !Exec <ProjectExec> cannot run on GPU because not all expressions can
be replaced
             @Expression <AttributeReference> customerID#0 could run on GPU
             @Expression <Alias> Charge AS kind#126 could run on GPU
```

```
                @Expression <Literal> Charge could run on GPU
              @Expression <AttributeReference> value#129 could run on GPU
              @Expression <Alias> add_months(2022-11-15, cast(-(cast(_we0#142 as
 bigint) + last_month#128L) as int)) AS date#149 could run on GPU
                ! <AddMonths> add_months(2022-11-15, cast(-
(cast(_we0#142 as bigint) + last_month#128L) as int)) cannot run
 on GPU because GPU does not currently support the operator class
 org.apache.spark.sql.catalyst.expressions.AddMonths
                  @Expression <Literal> 2022-11-15 could run on GPU
                  @Expression <Cast> cast(-(cast(_we0#142 as bigint) +
 last_month#128L) as int) could run on GPU
                    @Expression <UnaryMinus> -(cast(_we0#142 as bigint) +
 last_month#128L) could run on GPU
                      @Expression <Add> (cast(_we0#142 as bigint) +
 last_month#128L) could run on GPU
                        @Expression <Cast> cast(_we0#142 as bigint) could run on
 GPU
                          @Expression <AttributeReference> _we0#142 could run on
 GPU
                        @Expression <AttributeReference> last_month#128L could run
 on GPU
```

Uso da integração do Amazon Redshift para Apache Spark no Amazon EMR no EKS

Com as versões 6.9.0 e posteriores do Amazon EMR, cada imagem de versão inclui um conector entre o Apache Spark e o Amazon Redshift. Dessa forma, você pode usar o Spark no Amazon EMR no EKS para processar dados armazenados no Amazon Redshift. A integração é baseada no conector de código aberto `spark-redshift`. Para o Amazon EMR no EKS, a integração do Amazon Redshift para Apache Spark está inclusa como uma integração nativa.

Tópicos

- Inicialização de uma aplicação do Spark usando a integração do Amazon Redshift para Apache Spark
- Autenticação com a integração do Amazon Redshift para Apache Spark
- Leitura e gravação de e para o Amazon Redshift
- Considerações e limitações ao usar o conector do Spark

Inicialização de uma aplicação do Spark usando a integração do Amazon Redshift para Apache Spark

Para usar a integração, você deve transferir as dependências obrigatórias do Redshift para Spark com o trabalho do Spark. Você deve usar `--jars` para incluir as bibliotecas relacionadas ao conector do Redshift. Para visualizar outros locais de arquivo com suporte pela opção `--jars`, consulte a seção Advanced Dependency Management da documentação do Apache Spark.

- `spark-redshift.jar`
- `spark-avro.jar`
- `RedshiftJDBC.jar`
- `minimal-json.jar`

Para iniciar uma aplicação do Spark com a integração do Amazon Redshift para Apache Spark na versão 6.9.0 ou em versões posteriores do Amazon EMR no EKS, use o comando de exemplo a seguir. Observe que os caminhos listados com a opção `--conf spark.jars` são os caminhos padrão para os arquivos JAR.

```
aws emr-containers start-job-run \

--virtual-cluster-id cluster_id \
--execution-role-arn arn \
--release-label emr-6.9.0-latest\
--job-driver '{
    "sparkSubmitJobDriver": {
        "entryPoint": "s3://script_path",
            "sparkSubmitParameters":
            "--conf spark.kubernetes.file.upload.path=s3://upload_path
             --conf spark.jars=
                /usr/share/aws/redshift/jdbc/RedshiftJDBC.jar,
                /usr/share/aws/redshift/spark-redshift/lib/spark-redshift.jar,
                /usr/share/aws/redshift/spark-redshift/lib/spark-avro.jar,
                /usr/share/aws/redshift/spark-redshift/lib/minimal-json.jar"
                            }
            }'
```

Autenticação com a integração do Amazon Redshift para Apache Spark

Use o AWS Secrets Manager para recuperar credenciais e conectar-se ao Amazon Redshift

Você pode armazenar credenciais no Secrets Manager para realizar a autenticação com segurança no Amazon Redshift. É possível fazer com que seu trabalho do Spark chame a API `GetSecretValue` para buscar as credenciais:

```
from pyspark.sql import SQLContextimport boto3

sc = # existing SparkContext
sql_context = SQLContext(sc)

secretsmanager_client = boto3.client('secretsmanager',
 region_name=os.getenv('AWS_REGION'))
secret_manager_response = secretsmanager_client.get_secret_value(
    SecretId='string',
    VersionId='string',
    VersionStage='string'
)
username = # get username from secret_manager_response
password = # get password from secret_manager_response
url = "jdbc:redshift://redshifthost:5439/database?user=" + username + "&password="
 + password

# Access to Redshift cluster using Spark
```

Uso da autenticação baseada no IAM com o perfil de execução de trabalho do Amazon EMR no EKS

A partir da versão 6.9.0 do Amazon EMR no EKS, a versão 2.1 ou as versões superiores do driver JDBC do Amazon Redshift são empacotadas no ambiente. Com a versão 2.1 e versões superiores do driver JDBC, é possível especificar o URL do JDBC e não incluir o nome de usuário e a senha brutos. Em vez disso, você pode especificar o esquema `jdbc:redshift:iam://`. Isso comanda o driver JDBC para usar seu perfil de execução de trabalho do Amazon EMR no EKS para buscar as credenciais automaticamente.

Consulte Configurar uma conexão JDBC ou ODBC para usar credenciais do IAM no Guia de gerenciamento do Amazon Redshift para obter mais informações.

O exemplo de URL a seguir usa um esquema `jdbc:redshift:iam://`.

```
jdbc:redshift:iam://examplecluster.abc123xyz789.us-west-2.redshift.amazonaws.com:5439/
dev
```

As permissões apresentadas a seguir são obrigatórias para o seu perfil de execução de trabalho quando ele atende às condições fornecidas.

Permissão	Condições para se tornar obrigatória para o perfil de execução de trabalho
`redshift:GetClusterCredentials`	Obrigatória para que o driver JDBC busque as credenciais do Amazon Redshift.
`redshift:DescribeCluster`	Obrigatória se você especificar o cluster do Amazon Redshift e a Região da AWS no URL do JDBC em vez do endpoint.
`redshift-serverless:GetCredentials`	Obrigatória para que o driver JDBC busque as credenciais do Amazon Redshift sem servidor.
`redshift-serverless:GetWorkgroup`	Obrigatória se você estiver usando o Amazon Redshift sem servidor e especificar o URL em termos de nome e de região do grupo de trabalho.

Sua política de perfil de execução de trabalho deve ter as permissões apresentadas a seguir.

```
{
            "Effect": "Allow",
            "Action": [
                "redshift:GetClusterCredentials",
                "redshift:DescribeCluster",
                "redshift-serverless:GetCredentials",
                "redshift-serverless:GetWorkgroup"
            ],
            "Resource": [

 "arn:aws:redshift:AWS_REGION:ACCOUNT_ID:dbname:CLUSTER_NAME/DATABASE_NAME",
```

```
                "arn:aws:redshift:AWS_REGION:ACCOUNT_ID:dbuser:DATABASE_NAME/USER_NAME"
            ]
        }
```

Autenticação no Amazon Redshift com um driver JDBC

Definição de um nome de usuário e de uma senha no URL do JDBC

Para autenticar um trabalho do Spark em um cluster do Amazon Redshift, você pode especificar o nome e a senha do banco de dados do Amazon Redshift no URL do JDBC.

> **Note**
>
> Se você transferir as credenciais do banco de dados no URL, qualquer pessoa que tenha acesso ao URL também poderá acessar as credenciais. Este método geralmente não é recomendado porque não é uma opção segura.

Se a segurança não for uma preocupação para sua aplicação, você poderá usar o seguinte formato para definir o nome de usuário e a senha no URL do JDBC:

```
jdbc:redshift://redshifthost:5439/database?user=username&password=password
```

Leitura e gravação de e para o Amazon Redshift

Os exemplos de código apresentados a seguir usam o PySpark para realizar a leitura e a gravação de dados de exemplo de e para um banco de dados do Amazon Redshift com uma API de fonte de dados e com o Spark SQL.

Data source API

Use o PySpark para realizar a leitura e a gravação de dados de exemplo de e para um banco de dados do Amazon Redshift com uma API de fonte de dados.

```
import boto3
from pyspark.sql import SQLContext

sc = # existing SparkContext
sql_context = SQLContext(sc)
```

```
url = "jdbc:redshift:iam://redshifthost:5439/database"
aws_iam_role_arn = "arn:aws:iam::accountID:role/roleName"

df = sql_context.read \
    .format("io.github.spark_redshift_community.spark.redshift") \
    .option("url", url) \
    .option("dbtable", "tableName") \
    .option("tempdir", "s3://path/for/temp/data") \
    .option("aws_iam_role", "aws_iam_role_arn") \
    .load()

df.write \
    .format("io.github.spark_redshift_community.spark.redshift") \
    .option("url", url) \
    .option("dbtable", "tableName_copy") \
    .option("tempdir", "s3://path/for/temp/data") \
    .option("aws_iam_role", "aws_iam_role_arn") \
    .mode("error") \
    .save()
```

SparkSQL

Use o PySpark para realizar a leitura e a gravação de dados de exemplo de e para um banco de dados do Amazon Redshift usando o Spark SQL.

```
import boto3
import json
import sys
import os
from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .enableHiveSupport() \
    .getOrCreate()

url = "jdbc:redshift:iam://redshifthost:5439/database"
aws_iam_role_arn = "arn:aws:iam::accountID:role/roleName"

bucket = "s3://path/for/temp/data"
tableName = "tableName" # Redshift table name

s = f"""CREATE TABLE IF NOT EXISTS {tableName} (country string, data string)
```

```
    USING io.github.spark_redshift_community.spark.redshift
    OPTIONS (dbtable '{tableName}', tempdir '{bucket}', url '{url}', aws_iam_role
 '{aws_iam_role_arn}' ); """

spark.sql(s)

columns = ["country" ,"data"]
data = [("test-country","test-data")]
df = spark.sparkContext.parallelize(data).toDF(columns)

# Insert data into table
df.write.insertInto(tableName, overwrite=False)
df = spark.sql(f"SELECT * FROM {tableName}")
df.show()
```

Considerações e limitações ao usar o conector do Spark

- Recomendamos que você ative o SSL para a conexão JDBC do Spark no Amazon EMR para o Amazon Redshift.
- Recomendamos que você gerencie as credenciais do cluster do Amazon Redshift no AWS Secrets Manager como uma prática recomendada. Consulte Using AWS Secrets Manager to retrieve credentials for connecting to Amazon Redshift para obter um exemplo.
- Recomendamos que você transmita um perfil do IAM com o parâmetro `aws_iam_role` para o parâmetro de autenticação do Amazon Redshift.
- No momento, o parâmetro `tempformat` não é compatível com o formato Parquet.
- O URI `tempdir` aponta para um local do Amazon S3. Esse diretório temporário não é limpo automaticamente e, portanto, pode incorrer em custos adicionais.
- Considere as seguintes recomendações para o Amazon Redshift:
  - Recomendamos bloquear o acesso público ao cluster do Amazon Redshift.
  - Recomendamos ativar o registro em log de auditoria do Amazon Redshift.
  - Recomendamos ativar a criptografia em repouso do Amazon Redshift.
- Considere as seguintes recomendações para o Amazon S3:
  - Recomendamos bloquear o acesso público aos buckets do Amazon S3.
  - Recomendamos usar a criptografia do lado do servidor do Amazon S3 para criptografar os buckets do S3 que você usa.

- Recomendamos usar as políticas de ciclo de vida do Amazon S3 para definir as regras de retenção para o bucket do S3.
- O Amazon EMR sempre verifica o código importado do código aberto para a imagem. Por motivos de segurança, não oferecemos suporte à codificação de chaves de acesso da AWS no URI `tempdir` como um método de autenticação do Spark para o Amazon S3.

Para obter mais informações sobre como usar o conector e os parâmetros compatíveis, consulte os seguintes recursos:

- Integração do Amazon Redshift para Apache Spark no Guia de gerenciamento do Amazon Redshift.
- O repositório da comunidade `spark-redshift` no GitHub.

Uso do Volcano como um programador personalizado para Apache Spark no Amazon EMR no EKS

Com o Amazon EMR no EKS, você pode usar o operador do Spark ou o spark-submit para executar trabalhos do Spark com programadores personalizados do Kubernetes. Este tutorial aborda como executar trabalhos do Spark com um programador do Volcano em uma fila personalizada.

Visão geral

O Volcano pode ajudar a gerenciar a programação do Spark com funções avançadas, como a programação de filas, a programação de compartilhamento equitativo e a reserva de recursos. Para obter mais informações sobre os benefícios do Volcano, consulte Why Spark chooses Volcano as built-in batch scheduler on Kubernetes? no blog CNCF da Linux Foundation.

Instalação e configuração do Volcano

1. Escolha um dos seguintes comandos kubectl para instalar o Volcano, com base em suas necessidades arquitetônicas:

```
# x86_64
kubectl apply -f https://raw.githubusercontent.com/volcano-sh/volcano/v1.5.1/
installer/volcano-development.yaml
# arm64:
```

```
kubectl apply -f https://raw.githubusercontent.com/volcano-sh/volcano/v1.5.1/
installer/volcano-development-arm64.yaml
```

2. Prepare uma fila de exemplo do Volcano. Uma fila corresponde a uma coleção de PodGroups. A fila adota o FIFO e é a base para a divisão de recursos.

```
cat << EOF > volcanoQ.yaml
apiVersion: scheduling.volcano.sh/v1beta1
kind: Queue
metadata:
  name: sparkqueue
spec:
  weight: 4
  reclaimable: false
  capability:
    cpu: 10
    memory: 20Gi
EOF

kubectl apply -f volcanoQ.yaml
```

3. Faça upload de um exemplo de manifesto do PodGroup para o Amazon S3. O PodGroup corresponde a um grupo de pods com uma forte associação. Normalmente, você usa um PodGroup para a programação em lote. Envie o PodGroup de exemplo, apresentado a seguir, para a fila definida na etapa anterior.

```
cat << EOF > podGroup.yaml
apiVersion: scheduling.volcano.sh/v1beta1
kind: PodGroup
spec:
  # Set minMember to 1 to make a driver pod
  minMember: 1
  # Specify minResources to support resource reservation.
  # Consider the driver pod resource and executors pod resource.
  # The available resources should meet the minimum requirements of the Spark job
  # to avoid a situation where drivers are scheduled, but they can't schedule
  # sufficient executors to progress.
  minResources:
    cpu: "1"
    memory: "1Gi"
  # Specify the queue. This defines the resource queue that the job should be
 submitted to.
  queue: sparkqueue
```

```
EOF

aws s3 mv podGroup.yaml s3://bucket-name
```

Execução de uma aplicação do Spark usando o programador do Volcano com o operador do Spark

1. Se ainda não o fez, conclua as etapas apresentadas nas seguintes seções para se preparar para usar:

   a. Instalação e configuração do Volcano

   b. Configuração do operador do Spark para o Amazon EMR no EKS

   c. Instalação do operador do Spark

   Inclua os seguintes argumentos ao executar o comando `helm install spark-operator-demo`:

   ```
   --set batchScheduler.enable=true
   --set webhook.enable=true
   ```

2. Crie um arquivo `SparkApplication` de definição `spark-pi.yaml` com `batchScheduler` configurado.

   ```
   apiVersion: "sparkoperator.k8s.io/v1beta2"
   kind: SparkApplication
   metadata:
     name: spark-pi
     namespace: spark-operator
   spec:
     type: Scala
     mode: cluster
     image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.10.0:latest"
     imagePullPolicy: Always
     mainClass: org.apache.spark.examples.SparkPi
     mainApplicationFile: "local:///usr/lib/spark/examples/jars/spark-examples.jar"
     sparkVersion: "3.3.1"
     batchScheduler: "volcano"   #Note: You must specify the batch scheduler name as
    'volcano'
     restartPolicy:
       type: Never
   ```

```
  volumes:
    - name: "test-volume"
      hostPath:
        path: "/tmp"
        type: Directory
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.3.1
    serviceAccount: emr-containers-sa-spark
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
  executor:
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      version: 3.3.1
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
```

3. Envie a aplicação do Spark com o comando apresentado a seguir. Isso também cria um objeto `SparkApplication` chamado `spark-pi`:

```
kubectl apply -f spark-pi.yaml
```

4. Verifique os eventos do objeto `SparkApplication` com o seguinte comando:

```
kubectl describe pods spark-pi-driver --namespace spark-operator
```

O primeiro evento do pod mostrará que o Volcano programou os pods:

```
Type    Reason     Age   From               Message
----    ------     ----  ----               -------
Normal  Scheduled  23s   volcano            Successfully assigned default/spark-
pi-driver to integration-worker2
```

Execução de uma aplicação do Spark usando o programador do Volcano com o `spark-submit`

1. Primeiro, conclua as etapas na seção Configuração do spark-submit para o Amazon EMR no EKS. Você deve desenvolver a distribuição do `spark-submit` com suporte do Volcano. Para obter mais informações, consulte a seção Build de Using Volcano as Customized Scheduler for Spark on Kubernetes na documentação do Apache Spark.
2. Defina os valores para as seguintes variáveis de ambiente:

```
export SPARK_HOME=spark-home
export MASTER_URL=k8s://Amazon-EKS-cluster-endpoint
```

3. Envie a aplicação do Spark com o seguinte comando:

```
$SPARK_HOME/bin/spark-submit \
 --class org.apache.spark.examples.SparkPi \
 --master $MASTER_URL \
 --conf spark.kubernetes.container.image=895885662937.dkr.ecr.us-
west-2.amazonaws.com/spark/emr-6.10.0:latest \
 --conf spark.kubernetes.authenticate.driver.serviceAccountName=spark \
 --deploy-mode cluster \
 --conf spark.kubernetes.namespace=spark-operator \
 --conf spark.kubernetes.scheduler.name=volcano \
 --conf spark.kubernetes.scheduler.volcano.podGroupTemplateFile=/path/to/podgroup-
template.yaml \
 --conf
 spark.kubernetes.driver.pod.featureSteps=org.apache.spark.deploy.k8s.features.VolcanoFeatu
 \
 --conf
 spark.kubernetes.executor.pod.featureSteps=org.apache.spark.deploy.k8s.features.VolcanoFea
 \
 local:///usr/lib/spark/examples/jars/spark-examples.jar 20
```

4. Verifique os eventos do objeto `SparkApplication` com o seguinte comando:

```
kubectl describe pod spark-pi --namespace spark-operator
```

O primeiro evento do pod mostrará que o Volcano programou os pods:

```
Type    Reason     Age   From               Message
```

```
----    ------     ----  ----                 -------
Normal  Scheduled  23s   volcano              Successfully assigned default/spark-
pi-driver to integration-worker2
```

Uso do YuniKorn como um programador personalizado para Apache Spark no Amazon EMR no EKS

Com o Amazon EMR no EKS, você pode usar o operador do Spark ou o spark-submit para executar trabalhos do Spark com programadores personalizados do Kubernetes. Este tutorial aborda como executar trabalhos do Spark com um programador do YuniKorn em uma fila personalizada e com agendamento coletivo.

Visão geral

O Apache YuniKorn pode ajudar a gerenciar a programação do Spark com programação ciente da aplicação para que você possa ter controle otimizado sobre as cotas e as prioridades de recursos. Com o agendamento coletivo, o YuniKorn programa uma aplicação somente quando a solicitação mínima de recursos para a aplicação puder ser atendida. Para obter mais informações, consulte What is gang scheduling no site de documentação do Apache YuniKorn.

Criação do cluster e preparação para usar o YuniKorn

Use as etapas a seguir para implantar um cluster do Amazon EKS. Você pode alterar a Região da AWS (`region`) e as zonas de disponibilidade (`availabilityZones`).

1. Defina o cluster do Amazon EKS:

```
cat <<EOF >eks-cluster.yaml
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: emr-eks-cluster
  region: eu-west-1

vpc:
  clusterEndpoints:
    publicAccess: true
```

```
    privateAccess: true

iam:
  withOIDC: true

nodeGroups:
  - name: spark-jobs
    labels: { app: spark }
    instanceType: m5.xlarge
    desiredCapacity: 2
    minSize: 2
    maxSize: 3
    availabilityZones: ["eu-west-1a"]
EOF
```

2. Crie o cluster:

```
eksctl create cluster -f eks-cluster.yaml
```

3. Crie o namespace `spark-job` em que você executará o trabalho do Spark:

```
kubectl create namespace spark-job
```

4. Em seguida, crie um perfil e uma associação de perfis do Kubernetes. Isso é obrigatório para a conta de serviço usada pela execução de trabalho do Spark.

   a. Defina a conta de serviço, o perfil e a associação de perfis para os trabalhos do Spark.

```
cat <<EOF >emr-job-execution-rbac.yaml
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: spark-sa
  namespace: spark-job
automountServiceAccountToken: false
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: spark-role
  namespace: spark-job
rules:
```

```
  - apiGroups: ["", "batch","extensions"]
    resources: ["configmaps","serviceaccounts","events","pods","pods/
exec","pods/log","pods/
portforward","secrets","services","persistentvolumeclaims"]
    verbs: ["create","delete","get","list","patch","update","watch"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: spark-sa-rb
  namespace: spark-job
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: spark-role
subjects:
  - kind: ServiceAccount
    name: spark-sa
    namespace: spark-job
EOF
```

b. Aplique a definição de perfil e de associação de perfis do Kubernetes com o seguinte comando:

```
kubectl apply -f emr-job-execution-rbac.yaml
```

Instalação e configuração do YuniKorn

1. Use o seguinte comando kubectl para criar um namespace `yunikorn` para implantar o programador do Yunikorn:

```
kubectl create namespace yunikorn
```

2. Para instalar o programador, execute os seguintes comandos do Helm:

```
helm repo add yunikorn https://apache.github.io/yunikorn-release
```

```
helm repo update
```

```
helm install yunikorn yunikorn/yunikorn --namespace yunikorn
```

Execução de uma aplicação do Spark usando o programador do YuniKorn com o operador do Spark

1. Se ainda não o fez, conclua as etapas apresentadas nas seguintes seções para se preparar para usar:

   a. Criação do cluster e preparação para usar o YuniKorn

   b. Instalação e configuração do YuniKorn

   c. Configuração do operador do Spark para o Amazon EMR no EKS

   d. Instalação do operador do Spark

   Inclua os seguintes argumentos ao executar o comando `helm install spark-operator-demo`:

   ```
   --set batchScheduler.enable=true
   --set webhook.enable=true
   ```

2. Crie um arquivo de definição `SparkApplication` `spark-pi.yaml`.

   Para usar o YuniKorn como um programador para os trabalhos, você deve adicionar determinados rótulos e anotações à definição da sua aplicação. As anotações e os rótulos especificam a fila do seu trabalho e a estratégia de programação que você deseja usar.

   No exemplo a seguir, a anotação `schedulingPolicyParameters` determina o agendamento coletivo para a aplicação. Em seguida, o exemplo cria grupos de tarefas, ou “coletivos” de tarefas, para especificar a capacidade mínima que deve estar disponível antes da programação dos pods para iniciar a execução do trabalho. E, por fim, é especificado na definição do grupo de tarefas o uso de grupos de nós com o rótulo `"app": "spark"`, conforme definido na seção Criação do cluster e preparação para usar o YuniKorn.

   ```
   apiVersion: "sparkoperator.k8s.io/v1beta2"
   kind: SparkApplication
   metadata:
     name: spark-pi
     namespace: spark-job
   ```

```
spec:
  type: Scala
  mode: cluster
  image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.10.0:latest"
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "local:///usr/lib/spark/examples/jars/spark-examples.jar"
  sparkVersion: "3.3.1"
  restartPolicy:
    type: Never
  volumes:
    - name: "test-volume"
      hostPath:
        path: "/tmp"
        type: Directory
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.3.1
    annotations:
      yunikorn.apache.org/schedulingPolicyParameters: "placeholderTimeoutSeconds=30
 gangSchedulingStyle=Hard"
      yunikorn.apache.org/task-group-name: "spark-driver"
      yunikorn.apache.org/task-groups: |-
        [{
            "name": "spark-driver",
            "minMember": 1,
            "minResource": {
              "cpu": "1200m",
              "memory": "1Gi"
            },
            "nodeSelector": {
              "app": "spark"
            }
          },
          {
            "name": "spark-executor",
            "minMember": 1,
            "minResource": {
              "cpu": "1200m",
              "memory": "1Gi"
            },
```

```
            "nodeSelector": {
              "app": "spark"
            }
        }]
    serviceAccount: spark-sa
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
  executor:
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      version: 3.3.1
    annotations:
      yunikorn.apache.org/task-group-name: "spark-executor"
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
```

3. Envie a aplicação do Spark com o comando apresentado a seguir. Isso também cria um objeto `SparkApplication` chamado `spark-pi`:

```
kubectl apply -f spark-pi.yaml
```

4. Verifique os eventos do objeto `SparkApplication` com o seguinte comando:

```
kubectl describe sparkapplication spark-pi --namespace spark-job
```

O primeiro evento do pod mostrará que o YuniKorn programou os pods:

```
Type    Reason            Age   From                          Message
----    ------            ----  ----                          -------
Normal Scheduling        3m12s yunikorn   spark-operator/org-apache-spark-examples-
sparkpi-2a777a88b98b8a95-driver is queued and waiting for allocation
Normal GangScheduling    3m12s yunikorn   Pod belongs to the taskGroup spark-
driver, it will be scheduled as a gang member
Normal Scheduled         3m10s yunikorn   Successfully assigned spark
Normal PodBindSuccessful 3m10s yunikorn   Pod spark-operator/
Normal TaskCompleted     2m3s  yunikorn   Task spark-operator/
Normal Pulling           3m10s kubelet    Pulling
```

Execução de uma aplicação do Spark usando o programador do YuniKorn com o **spark-submit**

1. Primeiro, conclua as etapas na seção Configuração do spark-submit para o Amazon EMR no EKS.
2. Defina os valores para as seguintes variáveis de ambiente:

```
export SPARK_HOME=spark-home
export MASTER_URL=k8s://Amazon-EKS-cluster-endpoint
```

3. Envie a aplicação do Spark com o seguinte comando:

   No exemplo a seguir, a anotação `schedulingPolicyParameters` determina o agendamento coletivo para a aplicação. Em seguida, o exemplo cria grupos de tarefas, ou “coletivos” de tarefas, para especificar a capacidade mínima que deve estar disponível antes da programação dos pods para iniciar a execução do trabalho. E, por fim, é especificado na definição do grupo de tarefas o uso de grupos de nós com o rótulo `"app": "spark"`, conforme definido na seção Criação do cluster e preparação para usar o YuniKorn.

```
$SPARK_HOME/bin/spark-submit \
 --class org.apache.spark.examples.SparkPi \
 --master $MASTER_URL \
 --conf spark.kubernetes.container.image=895885662937.dkr.ecr.us-
west-2.amazonaws.com/spark/emr-6.10.0:latest \
 --conf spark.kubernetes.authenticate.driver.serviceAccountName=spark-sa \
 --deploy-mode cluster \
 --conf spark.kubernetes.namespace=spark-job \
 --conf spark.kubernetes.scheduler.name=yunikorn \
 --conf spark.kubernetes.driver.annotation.yunikorn.apache.org/
schedulingPolicyParameters="placeholderTimeoutSeconds=30 gangSchedulingStyle=Hard"
 \
 --conf spark.kubernetes.driver.annotation.yunikorn.apache.org/task-group-
name="spark-driver" \
 --conf spark.kubernetes.executor.annotation.yunikorn.apache.org/task-group-
name="spark-executor" \
 --conf spark.kubernetes.driver.annotation.yunikorn.apache.org/task-groups='[{
            "name": "spark-driver",
            "minMember": 1,
            "minResource": {
              "cpu": "1200m",
              "memory": "1Gi"
```

```
            },
            "nodeSelector": {
              "app": "spark"
            }
          },
          {
            "name": "spark-executor",
            "minMember": 1,
            "minResource": {
              "cpu": "1200m",
              "memory": "1Gi"
            },
            "nodeSelector": {
              "app": "spark"
            }
        }]' \
 local:///usr/lib/spark/examples/jars/spark-examples.jar 20
```

4. Verifique os eventos do objeto `SparkApplication` com o seguinte comando:

```
kubectl describe pod spark-driver-pod --namespace spark-job
```

O primeiro evento do pod mostrará que o YuniKorn programou os pods:

```
Type    Reason           Age   From                          Message
----    ------           ----  ----                          -------
Normal Scheduling        3m12s yunikorn   spark-operator/org-apache-spark-examples-
sparkpi-2a777a88b98b8a95-driver is queued and waiting for allocation
Normal GangScheduling    3m12s yunikorn   Pod belongs to the taskGroup spark-
driver, it will be scheduled as a gang member
Normal Scheduled         3m10s yunikorn   Successfully assigned spark
Normal PodBindSuccessful 3m10s yunikorn   Pod spark-operator/
Normal TaskCompleted     2m3s  yunikorn   Task spark-operator/
Normal Pulling           3m10s kubelet    Pulling
```

Segurança no Amazon EMR no EKS

A segurança para com a nuvem na AWS é a nossa maior prioridade. Como cliente da AWS, você se beneficiará de datacenters e arquiteturas de rede criados para atender aos requisitos das empresas com as maiores exigências de segurança.

A segurança é uma responsabilidade compartilhada entre a AWS e você. O modelo de responsabilidade compartilhada descreve isso como segurança da nuvem e segurança na nuvem:

- Segurança da nuvem: a AWS é responsável pela proteção da infraestrutura que executa produtos da AWS na Nuvem AWS. A AWS também fornece serviços que podem ser usados com segurança. Auditores de terceiros testam e verificam regularmente a eficácia da nossa segurança como parte dos Programas de conformidade da AWS. Para saber mais sobre os programas de conformidade que se aplicam ao Amazon EMR, consulte Serviços da AWS no escopo por programa de conformidade.
- Segurança na nuvem: sua responsabilidade é determinada pelo serviço da AWS que você usa. Você também é responsável por outros fatores, incluindo a confidencialidade de seus dados, os requisitos da sua empresa e as leis e regulamentos aplicáveis.

Esta documentação ajuda você a entender como aplicar o modelo de responsabilidade compartilhada ao usar o Amazon EMR no EKS. Os tópicos a seguir demonstram como configurar o Amazon EMR no EKS para atender aos seus objetivos de segurança e de conformidade. Você também aprenderá a usar outros serviços da AWS que ajudam a monitorar e proteger os recursos do Amazon EMR no EKS.

Tópicos

- Práticas recomendadas de segurança para o Amazon EMR no EKS
- Proteção de dados
- Gerenciamento de identidade e acesso
- Registro e monitoramento
- Validação de conformidade para o Amazon EMR no EKS
- Resiliência no Amazon EMR no EKS
- Segurança da infraestrutura no Amazon EMR no EKS
- Análise de configuração e vulnerabilidade
- Conexão com o Amazon EMR no EKS usando um endpoint da VPC de interface

- Configuração do acesso entre contas para o Amazon EMR no EKS

Práticas recomendadas de segurança para o Amazon EMR no EKS

O Amazon EMR no EKS disponibiliza diversos recursos de segurança a serem considerados no desenvolvimento e na implementação das suas próprias políticas de segurança. As práticas recomendadas a seguir são diretrizes gerais e não representam uma solução completa de segurança. Como essas práticas recomendadas podem não ser adequadas ou suficientes no seu ambiente, trate-as como considerações úteis em vez de requisitos.

> **Note**
>
> Para obter mais práticas recomendadas de segurança, consulte Práticas recomendadas de segurança para o Amazon EMR no EKS.

Aplicação do princípio de privilégio mínimo

O Amazon EMR no EKS fornece uma política de acesso granular para aplicações que usam perfis do IAM, como perfis de execução. Esses perfis de execução são mapeados para as contas de serviço do Kubernetes por meio da política de confiança do perfil do IAM. O Amazon EMR no EKS cria pods em um namespace registrado do Amazon EKS que executa o código da aplicação fornecido pelo usuário. Os pods de trabalho que executam o código da aplicação assumem o perfil de execução ao se conectarem a outros serviços da AWS. Recomendamos que os perfis de execução recebam somente o conjunto de privilégios mínimos obrigatórios para o trabalho, como a cobertura da aplicação e o acesso ao destino do log. Também recomendamos auditar regularmente as permissões dos trabalhos e após qualquer alteração no código da aplicação.

Listagem de controle de acesso para endpoints

Os endpoints gerenciados podem ser criados somente para clusters do EKS que foram configurados para usar, no mínimo, uma sub-rede privada em sua VPC. Essa configuração restringe o acesso aos balanceadores de carga criados por endpoints gerenciados para que eles possam ser acessados somente usando a VPC. Para aumentar ainda mais a segurança, recomendamos configurar grupos de segurança com esses balanceadores de carga para que eles possam restringir o tráfego de entrada a um conjunto selecionado de endereços IP.

Obtenção das atualizações de segurança mais recentes para as imagens personalizadas

Para usar imagens personalizadas com o Amazon EMR no EKS, você pode instalar quaisquer binários e bibliotecas na imagem. Você é responsável pela aplicação de patches de segurança aos binários adicionados à imagem. As imagens do Amazon EMR no EKS têm aplicações regulares dos patches de segurança mais recentes. Para obter a imagem mais recente, você deve criar novamente as imagens personalizadas sempre que houver uma nova versão da imagem base da versão do Amazon EMR. Para obter mais informações, consulte Versões do Amazon EMR no EKS e Como selecionar um URI de imagem base.

Limitação do acesso à credencial do pod

O Kubernetes oferece suporte a diversos métodos de atribuição de credenciais para um pod. O provisionamento de múltiplos provedores de credenciais pode aumentar a complexidade do seu modelo de segurança. O Amazon EMR no EKS adotou o uso de perfis do IAM para contas de serviços (IRSA) como um provedor de credencial padrão em um namespace do EKS registrado. Outros métodos não têm suporte, incluindo kube2iam, kiam e o uso de um perfil de instância do EC2 da instância em execução no cluster.

Isolamento de código de uma aplicação não confiável

O Amazon EMR no EKS não inspeciona a integridade do código de uma aplicação enviada pelos usuários do sistema. Se você estiver executando um cluster virtual multilocatário configurado usando diversos perfis de execução, que podem ser usados para enviar trabalhos por locatários não confiáveis que executam códigos arbitrários, há o risco de uma aplicação mal-intencionada escalar os privilégios. Nesta situação, considere isolar perfis de execução com privilégios semelhantes em um cluster virtual diferente.

Permissões de controle de acesso por perfil (RBAC)

Os administradores devem controlar rigorosamente as permissões de controle de acesso por perfil (RBAC) para o Amazon EMR em namespaces gerenciados pelo EKS. No mínimo, as permissões apresentadas a seguir não devem ser concedidas aos emissores de trabalhos no Amazon EMR em namespaces gerenciados pelo EKS.

- Permissões de RBAC do Kubernetes para modificar o configmap: porque o Amazon EMR no EKS usa os configmaps do Kubernetes para gerar modelos de pod gerenciados que têm o nome da conta de serviço gerenciada. Este atributo não deve sofrer mutação.
- Permissões de RBAC do Kubernetes para executar em pods do Amazon EMR no EKS: para evitar conceder acesso a modelos de pod gerenciados que têm o nome da conta de serviço gerenciada. Este atributo não deve sofrer mutação. Essa permissão também pode conceder acesso ao token JWT montado no pod, que pode ser usado para recuperar as credenciais do perfil de execução.
- Permissões de RBAC do Kubernetes para criar pods: para evitar que os usuários criem pods usando uma ServiceAccount do Kubernetes que pode ser mapeada para um perfil do IAM com mais privilégios da AWS do que o usuário.
- Permissões de RBAC do Kubernetes para implantar webhooks mutantes: para evitar que os usuários usem o webhook mutante para alterar o nome da ServiceAccount do Kubernetes para pods criados pelo Amazon EMR no EKS.
- Permissões de RBAC do Kubernetes para ler segredos do Kubernetes: para evitar que os usuários façam a leitura de dados confidenciais armazenados nesses segredos.

Restrição do acesso às credenciais do perfil do IAM ou do perfil de instância do grupo de nós

- Recomendamos que você atribua permissões mínimas da AWS aos perfis do IAM do grupo de nós. Isso ajuda a evitar a escalação de privilégios por código que pode ser executado usando credenciais do perfil de instância de nós de processamento do EKS.
- Para bloquear completamente o acesso às credenciais do perfil de instância para todos os pods executados no Amazon EMR em namespaces gerenciados pelo EKS, recomendamos que você execute comandos `iptables` em nós do EKS. Para obter mais informações, consulte Restricting access to Amazon EC2 instance profile credentials. No entanto, é importante definir o escopo adequado para os perfis do IAM da sua conta de serviço para que os pods tenham todas as permissões necessárias. Por exemplo, o perfil do IAM do nó recebe permissões para extrair imagens de contêiner do Amazon ECR. Se essas permissões não forem atribuídas a um pod, o pod não poderá extrair imagens de contêiner do Amazon ECR. O plug-in CNI da VPC também precisa ser atualizado. Para obter mais informações, consulte Walkthrough: Updating the VPC CNI plugin to use IAM roles for service accounts.

Proteção de dados

O modelo de responsabilidade compartilhada da AWS se aplica à proteção de dados no Amazon EMR no EKS. Conforme descrito nesse modelo, a AWS é responsável por proteger a infraestrutura global que executa toda a Nuvem AWS. Você é responsável por manter o controle sobre seu conteúdo hospedado nessa infraestrutura. Esse conteúdo inclui as tarefas de configuração e gerenciamento de segurança dos serviços da AWS que você usa. Para obter mais informações sobre a privacidade de dados, consulte as Perguntas frequentes sobre privacidade de dados. Para obter informações sobre a proteção de dados na Europa, consulte a publicação The AWS Shared Responsibility Model and GDPR no blog de segurança da AWS.

Para finalidades de proteção de dados, recomendamos que você proteja as credenciais da conta da AWS e configure contas individuais com o AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com os recursos da AWS. Recomendamos TLS 1.2 ou posterior.
- Configure o registro em log das atividades da API e do usuário com o AWS CloudTrail.
- Use as soluções de criptografia da AWS, juntamente com todos os controles de segurança padrão nos serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados pessoais armazenados no Amazon S3.
- Use as opções de criptografia do Amazon EMR no EKS para criptografar dados em repouso e em trânsito.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-2 ao acessar a AWS por meio de uma interface de linha de comando ou uma API, use um endpoint do FIPS. Para obter mais informações sobre endpoints do FIPS, consulte Federal Information Processing Standard (FIPS) 140-2.

É altamente recomendável que você nunca coloque informações de identificação confidenciais, como números de conta dos seus clientes, em campos de formato livre, como um campo Name (Nome). Isso inclui quando você trabalha com o Amazon EMR no EKS ou com outros serviços da AWS usando o console, a API, a AWS CLI ou os AWS SDKs. Todos os dados inseridos no Amazon EMR no EKS ou em outros serviços podem ser coletados para a inclusão em logs de diagnóstico.

Ao fornecer um URL para um servidor externo, não inclua informações de credenciais no URL para validar a solicitação a esse servidor.

Criptografia em repouso

A criptografia de dados ajuda a impedir que usuários não autorizados leiam dados em um cluster e em sistemas de armazenamento físico de dados associados. Isso inclui dados salvos em mídias persistentes, conhecidos como dados em repouso, e dados que podem ser interceptados enquanto viajam pela rede, conhecidos como dados em trânsito.

A criptografia de dados requer chaves e certificados. É possível escolher entre diversas opções, incluindo chaves gerenciadas pelo AWS Key Management Service, chaves gerenciadas pelo Amazon S3 e chaves e certificados de provedores personalizados fornecidos por você. Ao usar o AWS KMS como seu provedor de chaves, cobranças são aplicáveis pelo armazenamento e uso de chaves de criptografia. Para obter mais informações, consulte Preços do AWS KMS.

Antes de especificar as opções de criptografia, decida quais sistemas de gerenciamento de chaves e de certificados você deseja usar. Em seguida, crie as chaves e os certificados para os provedores personalizados especificados como parte das configurações de criptografia.

Criptografia em repouso para dados do EMRFS no Amazon S3

A criptografia do Amazon S3 funciona com objetos do Sistema de Arquivos do EMR (EMRFS) lidos e gravados no Amazon S3. Você especifica a criptografia do lado do servidor (SSE) ou a criptografia do lado do cliente (CSE) do Amazon S3 como o modo de criptografia padrão ao habilitar a criptografia em repouso. Opcionalmente, você pode especificar diferentes métodos de criptografia para buckets individuais usando Per bucket encryption overrides (Substituições de criptografia por bucket). Independentemente de a criptografia do Amazon S3 estar habilitada, o Transport Layer Security (TLS) criptografa os objetos do EMRFS em trânsito entre os nós do cluster do EMR e o Amazon S3. Para obter informações detalhadas sobre a criptografia do Amazon S3, consulte Proteger dados com criptografia no Guia do desenvolvedor do Amazon Simple Storage Service.

> **Note**
>
> Quando você usa o AWS KMS, cobranças são aplicáveis ao armazenamento e ao uso de chaves de criptografia. Para obter mais informações, consulte Preços do AWS KMS.

Criptografia do lado do servidor do Amazon S3

Quando você configura a criptografia do lado do servidor do Amazon S3, o Amazon S3 criptografa os dados no nível do objeto à medida que os grava no disco e os descriptografa quando são acessados. Para obter mais informações sobre a SSE, consulte Proteger os dados usando criptografia do lado do servidor no Guia do desenvolvedor do Amazon Simple Storage Service.

Você pode escolher entre dois sistemas de gerenciamento de chaves diferentes ao especificar a SSE no Amazon EMR no EKS:

- SSE-S3: o Amazon S3 gerencia as chaves para você.
- SSE-KMS: você usa uma AWS KMS key para configurar políticas adequadas para o Amazon EMR no EKS.

A SSE com chaves fornecidas pelo cliente (SSE-C) não está disponível para o uso com o Amazon EMR no EKS.

Criptografia do lado do cliente do Amazon S3

Com a criptografia do lado do cliente do Amazon S3, a criptografia e a descriptografia do Amazon S3 ocorrem no cliente do EMRFS em seu cluster. Os objetos são criptografados antes de serem carregados no Amazon S3 e descriptografados após serem baixados. O provedor especificado por você fornece a chave de criptografia que o cliente usa. O cliente pode usar chaves fornecidas pelo AWS KMS (CSE-KMS) ou uma classe Java personalizada que fornece a chave raiz do lado do cliente (CSE-C). As especificações de criptografia são ligeiramente diferentes entre a CSE-KMS e a CSE-C, dependendo do provedor especificado e dos metadados do objeto que está sendo descriptografado ou criptografado. Para obter mais informações sobre essas diferenças, consulte Proteger dados usando a criptografia do lado do cliente no Guia do desenvolvedor do Amazon Simple Storage Service.

> **Note**
>
> A CSE do Amazon S3 garante somente que os dados do EMRFS trocados com o Amazon S3 sejam criptografados. Não são todos os dados nos volumes de instâncias do cluster que são criptografados. Além disso, como o Hue não usa o EMRFS, os objetos que o navegador de arquivos do S3 para Hue grava no Amazon S3 não são criptografados.

Criptografia de disco local

O Apache Spark oferece suporte à criptografia de dados temporários gravados em discos locais. Isso abrange arquivos embaralhados, vazamentos embaralhados e blocos de dados armazenados em disco para variáveis de armazenamento em cache e de transmissão. A criptografia de dados de saída gerados por aplicações com APIs como `saveAsHadoopFile` ou `saveAsTable` não é abrangida. Também é possível que não haja abrangência para arquivos temporários criados explicitamente pelo usuário. Para obter mais informações, consulte Local Storage Encryption na documentação do Spark. O Spark não oferece suporte a dados criptografados no disco local, como dados intermediários gravados em um disco local por um processo executor quando os dados não se encaixam na memória. Os dados que persistem no disco têm como escopo o runtime do trabalho, e a chave usada para criptografar os dados é gerada dinamicamente pelo Spark para cada execução de trabalho. Depois que o trabalho do Spark for encerrado, nenhum outro processo poderá descriptografar os dados.

Para pod de drivers e de executores, você criptografa dados em repouso que persistem no volume montado. Existem três opções diferentes de armazenamento nativo da AWS que você pode usar com o Kubernetes: EBS, EFS e FSx para Lustre. Todos os três oferecem criptografia em repouso usando uma chave gerenciada pelo serviço ou uma AWS KMS key. Para obter mais informações, consulte EKS Best Practices Guide. Com esta abordagem, todos os dados que persistem no volume montado são criptografados.

Gerenciamento de chaves

Você pode configurar o KMS para alternar automaticamente suas chaves do KMS. Isso alterna suas chaves uma vez por ano, enquanto salva as chaves antigas indefinidamente para que seus dados ainda possam ser descriptografados. Para obter informações adicionais, consulte Rotating AWS KMS keys.

Criptografia em trânsito

Vários mecanismos de criptografia estão habilitados com a criptografia em trânsito. Esses são recursos de código aberto, específicos da aplicação e podem variar de acordo com a versão do Amazon EMR no EKS. Os seguintes recursos de criptografia específicos da aplicação podem ser habilitados com o Amazon EMR no EKS:

- Spark
  - A comunicação de RPC interna entre os componentes do Spark, como o serviço de transferência de blocos e o serviço de shuffle externo, é criptografada usando a criptografia

AES-256 nas versões 5.9.0 e posteriores do Amazon EMR. Em versões anteriores, a comunicação RPC interna é criptografada usando SASL com DIGEST-MD5 como a criptografia.

- A comunicação do protocolo HTTP com interfaces de usuário, como o Spark History Server e servidores de arquivos habilitados para HTTPS é criptografada usando a configuração de SSL do Spark. Para obter mais informações, consulte Configuração do SSL na documentação do Spark.

Para obter mais informações, consulte as configurações de segurança do Spark.

- Você deve permitir somente conexões criptografadas por HTTPS (TLS) que usam a condição aws:SecureTransport nas políticas do IAM do bucket do Amazon S3.
- Os resultados de consultas que transmitem para clientes JDBC ou ODBC são criptografados usando TLS.

Gerenciamento de identidade e acesso

O AWS Identity and Access Management (IAM) é um serviço da AWS service (Serviço da AWS) que ajuda a controlar o acesso aos recursos da AWS de forma segura. Os administradores do IAM controlam quem pode ser autenticado (iniciar sessão) e autorizado (ter permissões) para usar os recursos do Amazon EMR no EKS. O IAM é um AWS service (Serviço da AWS) que pode ser usado sem custo adicional.

Tópicos

- Público
- Como autenticar com identidades
- Gerenciamento do acesso usando políticas
- Como o Amazon EMR no EKS funciona com o IAM
- Uso de perfis vinculados ao serviço para o Amazon EMR no EKS
- Políticas gerenciadas para o Amazon EMR no EKS
- Uso de perfis de execução de trabalho com o Amazon EMR no EKS
- Exemplos de políticas baseadas em identidade para o Amazon EMR no EKS
- Políticas para controle de acesso baseado em etiquetas
- Solução de problemas de identidade e acesso do Amazon EMR no EKS

Público

A forma de usar o AWS Identity and Access Management (IAM) varia, dependendo do trabalho realizado no Amazon EMR no EKS.

Usuário do serviço: se você usar o serviço Amazon EMR no EKS para realizar seu trabalho, o administrador fornecerá as credenciais e as permissões de que você precisa. À medida que você usa mais recursos do Amazon EMR no EKS para realizar o trabalho, talvez sejam necessárias permissões adicionais. Entender como o acesso é gerenciado pode ajudar você a solicitar as permissões corretas ao seu administrador. Se você não conseguir acessar um recurso no Amazon EMR no EKS, consulte Solução de problemas de identidade e acesso do Amazon EMR no EKS.

Administrador do serviço: se você for responsável pelos recursos do Amazon EMR no EKS em sua empresa, provavelmente terá acesso total ao Amazon EMR no EKS. É sua função determinar quais atributos e recursos do Amazon EMR no EKS seus usuários do serviço devem acessar. Assim, é necessário enviar solicitações ao administrador do IAM para alterar as permissões dos usuários de seu serviço. Revise as informações nesta página para entender os conceitos básicos do IAM. Para saber mais sobre como sua empresa pode usar o IAM com o Amazon EMR no EKS, consulte Como o Amazon EMR no EKS funciona com o IAM.

Administrador do IAM: se você for um administrador do IAM, talvez deseje saber detalhes sobre como escrever políticas para gerenciar o acesso ao Amazon EMR no EKS. Para visualizar exemplos de políticas baseadas em identidade do Amazon EMR no EKS que podem ser usadas no IAM, consulte Exemplos de políticas baseadas em identidade para o Amazon EMR no EKS.

Como autenticar com identidades

A autenticação é a forma como você faz login na AWS usando suas credenciais de identidade. É necessário ser autenticado (fazer login na AWS) como o usuário raiz da Usuário raiz da conta da AWS, como usuário do IAM ou assumindo um perfil do IAM.

Você pode fazer login na AWS como uma identidade federada usando credenciais fornecidas por uma fonte de identidades. AWS IAM Identity Center Os usuários do IAM Identity Center, a autenticação única da empresa e as suas credenciais do Google ou do Facebook são exemplos de identidades federadas. Quando você faz login como uma identidade federada, o administrador já configurou anteriormente a federação de identidades usando perfis do IAM. Quando você acessa a AWS usando a federação, está indiretamente assumindo um perfil.

É possível fazer login no AWS Management Console ou no portal de acesso da AWS dependendo do tipo de usuário que você é. Para obter mais informações sobre como fazer login na AWS,

consulte How to sign in to your Conta da AWS (Como fazer login na conta da) no Início de Sessão da AWS User Guide (Guia do usuário do).

Se você acessar a AWS programaticamente, a AWS fornecerá um kit de desenvolvimento de software (SDK) e uma interface da linha de comando (CLI) para você assinar criptograficamente as solicitações usando as suas credenciais. Se você não utilizar as ferramentas da AWS, deverá assinar as solicitações por conta própria. Para obter mais informações sobre como usar o método recomendado para assinar solicitações por conta própria, consulte Assinar solicitações de API da AWS no Guia do usuário do IAM.

Independentemente do método de autenticação usado, também pode ser exigido que você forneça informações adicionais de segurança. Por exemplo, a AWS recomenda o uso da autenticação multifator (MFA) para aumentar a segurança de sua conta. Para saber mais, consulte Autenticação multifator no GuiaAWS IAM Identity Center do usuário. Usar a autenticação multifator (MFA) naAWS no Guia do usuário do IAM.

Usuário root da Conta da AWS

Ao criar uma Conta da AWS, você começa com uma identidade de login que tem acesso completo a todos os recursos e Serviços da AWS na conta. Essa identidade, denominada usuário raiz da Conta da AWS, e é acessada por login com o endereço de email e a senha que você usou para criar a conta. É altamente recomendável não usar o usuário raiz para tarefas diárias. Proteja as credenciais do usuário raiz e use-as para executar as tarefas que somente ele pode executar. Para obter a lista completa das tarefas que exigem login como usuário raiz, consulte Tarefas que exigem credenciais de usuário raiz no Guia do usuário do IAM.

Identidade federada

Como prática recomendada, exija que os usuários, inclusive os que precisam de acesso de administrador, usem a federação com um provedor de identidades para acessar Serviços da AWS usando credenciais temporárias.

Identidade federada é um usuário de seu diretório de usuários corporativos, um provedor de identidades da web, o AWS Directory Service, o diretório do Centro de Identidade ou qualquer usuário que acesse os Serviços da AWS usando credenciais fornecidas por meio de uma fonte de identidade. Quando as identidades federadas acessam Contas da AWS, elas assumem funções que fornecem credenciais temporárias.

Para o gerenciamento de acesso centralizado, recomendamos usar o AWS IAM Identity Center. Você pode criar usuários e grupos no Centro de Identidade do IAM ou se conectar e sincronizar

com um conjunto de usuários e grupos em sua própria fonte de identidade para uso em todas as suas Contas da AWS e aplicações. Para obter mais informações sobre o Centro de Identidade do IAM, consulte "What is IAM Identity Center?" (O que é o Centro de Identidade do IAM?) no Guia do usuário do AWS IAM Identity Center.

Grupos e usuários do IAM

Um usuário do IAM é uma identidade dentro da Conta da AWS que tem permissões específicas para uma única pessoa ou aplicação. Sempre que possível, recomendamos depender de credenciais temporárias em vez de criar usuários do IAM com credenciais de longo prazo, como senhas e chaves de acesso. No entanto, se você tiver casos de uso específicos que exijam credenciais de longo prazo com usuários do IAM, recomendamos alternar as chaves de acesso. Para obter mais informações, consulte Alterne as chaves de acesso regularmente para casos de uso que exijam credenciais de longo prazo no Guia do usuário do IAM.

Um grupo do IAM é uma identidade que especifica uma coleção de usuários do IAM. Não é possível fazer login como um grupo. É possível usar grupos para especificar permissões para vários usuários de uma vez. Os grupos facilitam o gerenciamento de permissões para grandes conjuntos de usuários. Por exemplo, você pode ter um grupo chamado IAMAdmins e atribuir a esse grupo permissões para administrar recursos do IAM.

Usuários são diferentes de funções. Um usuário é exclusivamente associado a uma pessoa ou a uma aplicação, mas uma função pode ser assumida por qualquer pessoa que precisar dela. Os usuários têm credenciais permanentes de longo prazo, mas as funções fornecem credenciais temporárias. Para saber mais, consulte Quando criar um usuário do IAM (em vez de uma função) no Guia do usuário do IAM.

Funções do IAM

Uma função do IAM é uma identidade dentro da Conta da AWS que tem permissões específicas. Ela é semelhante a um usuário do IAM, mas não está associada a uma pessoa específica. É possível assumir temporariamente um perfil do IAM no AWS Management Console alternando perfis. É possível assumir um perfil chamando uma operação de API da AWS CLI ou da AWS, ou usando um URL personalizado. Para obter mais informações sobre métodos para o uso de perfis, consulte Usar perfis do IAM no Guia do usuário do IAM.

Os perfis do IAM com credenciais temporárias são úteis nas seguintes situações:

- Acesso de usuário federado: para atribuir permissões a identidades federadas, você pode criar um perfil e definir permissões para ele. Quando uma identidade federada é autenticada, ela é

associada ao perfil e recebe as permissões definidas pelo perfil. Para obter mais informações sobre perfis para federação, consulte Criar um perfil para um provedor de identidades de terceiros no Guia do usuário do IAM. Se você usar o IAM Identity Center, configure um conjunto de permissões. Para controlar o que suas identidades podem acessar após a autenticação, o IAM Identity Center correlaciona o conjunto de permissões a um perfil no IAM. Para obter informações sobre conjuntos de permissões, consulte Permission sets (Conjuntos de permissões) no Guia do usuário do AWS IAM Identity Center.

- Permissões temporárias para usuários do IAM: um usuário ou um perfil do IAM pode assumir um perfil do IAM para obter temporariamente permissões diferentes para uma tarefa específica.
- Acesso entre contas: é possível usar um perfil do IAM para permitir que alguém (um principal confiável) em outra conta acesse recursos em sua conta. As funções são a principal forma de conceder acesso entre contas. No entanto, alguns Serviços da AWS permitem que você anexe uma política diretamente a um recurso (em vez de usar uma função como proxy). Para saber a diferença entre funções e políticas baseadas em recurso para acesso entre contas, consulte Como as funções do IAM diferem das políticas baseadas em recurso no Guia do usuário do IAM.
- Acesso entre serviços: alguns Serviços da AWS usam recursos em outros Serviços da AWS. Por exemplo, quando você faz uma chamada em um serviço, é comum que esse serviço execute aplicações no Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso usando as permissões do principal de chamada, usando uma função de serviço ou uma função vinculada ao serviço.
  - Permissões de principal: ao usar um usuário ou uma função do IAM para executar ações na AWS, você é considerado um principal. As políticas concedem permissões a uma entidade principal. Quando você usa alguns serviços, pode executar uma ação que, em seguida, aciona outra ação em outro serviço. Nesse caso, você deve ter permissões para executar ambas as ações. Para visualizar se uma ação requer ações dependentes adicionais em uma política, consulte Ações, recursos e chaves de condição para o Amazon EMR no EKS na Referência de autorização do serviço.
  - Função de serviço: uma função de serviço é uma função do IAM que um serviço assume para realizar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para obter mais informações, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do usuário do IAM.
  - Função vinculada a serviço: uma função vinculada a serviço é um tipo de função de serviço vinculada a um AWS service (Serviço da AWS). O serviço pode assumir a função de executar uma ação em seu nome. Os perfis vinculados ao serviço aparecem em sua Conta da AWS e são

de propriedade do serviço. Um administrador do IAM pode visualizar, mas não pode editar as permissões para funções vinculadas ao serviço.

- Aplicações em execução no Amazon EC2: é possível usar uma função do IAM para gerenciar credenciais temporárias para aplicações em execução em uma instância do EC2 e fazer solicitações da AWS CLI ou da AWS API. É preferível fazer isso a armazenar chaves de acesso na instância do EC2. Para atribuir uma função da AWS a uma instância do EC2 e disponibilizá-la para todas as suas aplicações, crie um perfil de instância que esteja anexado a ela. Um perfil de instância contém a função e permite que os programas em execução na instância do EC2 obtenham credenciais temporárias. Para mais informações, consulte Usar uma função do IAM para conceder permissões a aplicações em execução nas instâncias do Amazon EC2 no Guia do usuário do IAM.

Para saber se deseja usar as funções do IAM, consulte Quando criar uma função do IAM (em vez de um usuário) no Guia do usuário do IAM.

Gerenciamento do acesso usando políticas

Você controla o acesso na AWS criando políticas e anexando-as a identidades ou recursos da AWS. Uma política é um objeto na AWS que, quando associado a uma identidade ou recurso, define suas permissões. A AWS avalia essas políticas quando uma entidade principal (usuário, usuário raiz ou sessão de função) faz uma solicitação. As permissões nas políticas determinam se a solicitação será permitida ou negada. A maioria das políticas são armazenadas na AWS como documentos JSON. Para obter mais informações sobre a estrutura e o conteúdo de documentos de políticas JSON, consulte Visão geral das políticas JSON no Guia do usuário do IAM.

Os administradores podem usar as políticas JSON da AWS para especificar quem tem acesso a quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

Por padrão, usuários e perfis não têm permissões. Para conceder aos usuários permissões para executar ações nos recursos de que eles precisam, um administrador do IAM pode criar políticas do IAM. O administrador pode então adicionar as políticas do IAM aos perfis, e os usuários podem assumir os perfis.

As políticas do IAM definem permissões para uma ação, independentemente do método usado para executar a operação. Por exemplo, suponha que você tenha uma política que permite a ação `iam:GetRole`. Um usuário com essa política pode obter informações de funções do AWS Management Console, da AWS CLI ou da API da AWS.

Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como usuário, grupo de usuários ou função do IAM. Essas políticas controlam quais ações os usuários e funções podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Criar políticas do IAM no Guia do usuário do IAM.

As políticas baseadas em identidade podem ser categorizadas ainda mais como políticas em linha ou políticas gerenciadas. As políticas em linha são anexadas diretamente a um único usuário, grupo ou função. As políticas gerenciadas são políticas independentes que podem ser anexadas a vários usuários, grupos e funções na Conta da AWS. As políticas gerenciadas incluem políticas gerenciadas pela AWS e políticas gerenciadas pelo cliente. Para saber como escolher entre uma política gerenciada ou uma política em linha, consulte Escolher entre políticas gerenciadas e políticas em linha no Guia do usuário do IAM.

Políticas baseadas em recursos

Políticas baseadas em recurso são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de função do IAM e as políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o recurso ao qual a política está anexada, a política define quais ações um principal especificado pode executar nesse recurso e em que condições. Você deve especificar um principal em uma política baseada em recursos. As entidades principais podem incluir contas, usuários, funções, usuários federados ou Serviços da AWS.

Políticas baseadas em recursos são políticas em linha que estão localizadas nesse serviço. Não é possível usar as políticas gerenciadas da AWS do IAM em uma política baseada em recursos.

Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais principais (membros, usuários ou funções da conta) têm permissões para acessar um recurso. As ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

Amazon S3, AWS WAF e Amazon VPC são exemplos de serviços que oferecem suporte a ACLs. Para saber mais sobre ACLs, consulte Visão geral da lista de controle de acesso (ACL) no Guia do desenvolvedor do Amazon Simple Storage Service.

Outros tipos de política

A AWS oferece suporte a tipos de política menos comuns. Esses tipos de política podem definir o máximo de permissões concedidas a você pelos tipos de política mais comuns.

- Limites de permissões: um limite de permissões é um recurso avançado no qual você define o máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM (usuário ou perfil do IAM). É possível definir um limite de permissões para uma entidade. As permissões resultantes são a interseção das políticas baseadas em identidade de uma entidade e dos seus limites de permissões. As políticas baseadas em recurso que especificam o usuário ou a função no campo `Principal` não são limitadas pelo limite de permissões. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações sobre limites de permissões, consulte Limites de permissões para identidades do IAM no Guia do usuário do IAM.
- Políticas de controle de serviço (SCPs): SCPs são políticas JSON que especificam as permissões máximas para uma organização ou unidade organizacional (UO) no AWS Organizations. O AWS Organizations é um serviço para agrupar e gerenciar centralmente várias Contas da AWS pertencentes à sua empresa. Se você habilitar todos os recursos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as contas. O SCP limita as permissões para entidades em contas-membro, incluindo cada Usuário raiz da conta da AWS. Para obter mais informações sobre o Organizations e SCPs, consulte Como os SCPs funcionam no Guia do usuário do AWS Organizations.
- Políticas de sessão: são políticas avançadas que você transmite como um parâmetro quando cria de forma programática uma sessão temporária para uma função ou um usuário federado. As permissões da sessão resultante são a interseção das políticas baseadas em identidade do usuário ou da função e das políticas de sessão. As permissões também podem ser provenientes de uma política baseada em recurso. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações, consulte Políticas de sessão no Guia do usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como a AWS determina se deve permitir uma solicitação quando há vários tipos de política envolvidos, consulte Lógica da avaliação de políticas no Guia do usuário do IAM.

Como o Amazon EMR no EKS funciona com o IAM

Antes de usar o IAM para gerenciar o acesso ao Amazon EMR no EKS, saiba quais recursos do IAM estão disponíveis para uso com o Amazon EMR no EKS.

Recursos do IAM que podem ser usados com o Amazon EMR no EKS

Recurso do IAM	Compatível com o Amazon EMR no EKS
Políticas baseadas em identidade	Sim
Políticas baseadas em recursos	Não
Ações de políticas	Sim
Recursos de políticas	Sim
Chaves de condição de políticas	Sim
ACLs	Não
ABAC (etiquetas em políticas)	Sim
Credenciais temporárias	Sim
Permissões de entidades principais	Sim
Funções de serviço	Não
Funções vinculadas ao serviço	Sim

Para obter uma visualização de alto nível de como o Amazon EMR no EKS e outros serviços da AWS funcionam com a maioria dos recursos do IAM, consulte Serviços da AWS que funcionam com o IAM no Guia do usuário do IAM.

Políticas baseadas em identidade para o Amazon EMR no EKS

É compatível com políticas baseadas em identidade	Sim

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como usuário, grupo de usuários ou perfil do IAM. Essas políticas controlam quais ações os usuários e funções podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Criar políticas do IAM no Guia do usuário do IAM.

Com as políticas baseadas em identidade do IAM, é possível especificar ações ou recursos permitidos ou negados, bem como as condições sob as quais as ações são permitidas ou negadas. Você não pode especificar a entidade principal em uma política baseada em identidade porque ela se aplica ao usuário ou função à qual ela está anexado. Para saber mais sobre todos os elementos que podem ser usados em uma política JSON, consulte Referência de elementos de política JSON do IAM no Guia do usuário do IAM.

Exemplos de políticas baseadas em identidade para o Amazon EMR no EKS

Para visualizar exemplos de políticas baseadas em identidade do Amazon EMR no EKS, consulte Exemplos de políticas baseadas em identidade para o Amazon EMR no EKS.

Políticas baseadas em recursos para o Amazon EMR no EKS

Oferece suporte a políticas baseadas em recursos	Não

Políticas baseadas em recurso são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de função do IAM e as políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o recurso ao qual a política está anexada, a política define quais ações um principal especificado pode executar nesse recurso e em que condições. Você deve especificar um principal em uma política baseada em recursos. As entidades principais podem incluir contas, usuários, perfis, usuários federados ou Serviços da AWS.

Para permitir o acesso entre contas, você pode especificar uma conta inteira ou as entidades do IAM em outra conta como a entidade principal em uma política baseada em recurso. Adicionar um principal entre contas à política baseada em recurso é apenas metade da tarefa de estabelecimento da relação de confiança. Quando a entidade principal e o recurso estão em diferentes Contas da

AWS, um administrador do IAM da conta confiável também deve conceder à entidade principal (usuário ou função) permissão para acessar o recurso. Eles concedem permissão ao anexar uma política baseada em identidade para a entidade. No entanto, se uma política baseada em recurso conceder acesso a um principal na mesma conta, nenhuma política baseada em identidade adicional será necessária. Para obter mais informações, consulte Como as funções do IAM diferem de políticas baseadas em recursos no Guia do usuário do IAM.

Ações de políticas para o Amazon EMR no EKS

Oferece suporte a ações de políticas	Sim

Os administradores podem usar AWS as políticas JSON da para especificar quem tem acesso a quê. Ou seja, qual principal pode executar ações em quais recursos, e em que condições.

O elemento `Action` de uma política JSON descreve as ações que você pode usar para permitir ou negar acesso em uma política. As ações de política geralmente têm o mesmo nome que a operação de API da AWS associada. Existem algumas exceções, como ações somente de permissão, que não têm uma operação de API correspondente. Há também algumas operações que exigem várias ações em uma política. Essas ações adicionais são chamadas de ações dependentes.

Inclua ações em uma política para conceder permissões para executar a operação associada.

Para visualizar uma lista de ações do Amazon EMR no EKS, consulte Ações, recursos e chaves de condição para o Amazon EMR no EKS na Referência de autorização do serviço.

As ações de política no Amazon EMR no EKS usam o seguinte prefixo antes da ação:

```
emr-containers
```

Para especificar várias ações em uma única declaração, separe-as com vírgulas.

```
"Action": [
      "emr-containers:action1",
      "emr-containers:action2"
         ]
```

Para visualizar exemplos de políticas baseadas em identidade do Amazon EMR no EKS, consulte Exemplos de políticas baseadas em identidade para o Amazon EMR no EKS.

Recursos de políticas para o Amazon EMR no EKS

Oferece suporte a recursos de políticas	Sim

Os administradores podem usar AWS as políticas JSON da para especificar quem tem acesso a quê. Ou seja, qual principal pode executar ações em quais recursos, e em que condições.

O elemento `Resource` de política JSON especifica o objeto ou os objetos aos quais a ação se aplica. As instruções devem incluir um elemento `Resource` ou um elemento `NotResource`. Como prática recomendada, especifique um recurso usando seu Nome do recurso da Amazon (ARN). Isso pode ser feito para ações que oferecem suporte a um tipo de recurso específico, conhecido como permissões em nível de recurso.

Para ações que não oferecem suporte a permissões em nível de recurso, como operações de listagem, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

Para visualizar uma lista de tipos de recursos do Amazon EMR no EKS e seus ARNs, consulte Recursos definidos pelo Amazon EMR no EKS na Referência de autorização do serviço. Para aprender com quais ações você pode especificar o ARN de cada recurso, consulte Ações, recursos e chaves de condição para o Amazon EMR no EKS.

Para visualizar exemplos de políticas baseadas em identidade do Amazon EMR no EKS, consulte Exemplos de políticas baseadas em identidade para o Amazon EMR no EKS.

Chaves de condição de políticas para o Amazon EMR no EKS

Compatível com chaves de condição de política específicas do serviço	Sim

Os administradores podem usar AWS as políticas JSON da para especificar quem tem acesso a quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Condition` (ou bloco de `Condition`) permite que você especifique condições nas quais uma instrução está em vigor. O elemento `Condition` é opcional. É possível criar expressões condicionais que usam agentes de condição, como “igual a” ou “menor que”, para fazer a condição da política corresponder aos valores na solicitação.

Se você especificar vários elementos `Condition` em uma instrução ou várias chaves em um único elemento `Condition`, a AWS os avaliará usando uma operação lógica `AND`. Se você especificar vários valores para uma única chave de condição, a AWS avaliará a condição usando uma operação lógica `OR`. Todas as condições devem ser atendidas para que as permissões da instrução sejam concedidas.

Você também pode usar variáveis de espaço reservado ao especificar as condições. Por exemplo, é possível conceder a um usuário do IAM permissão para acessar um recurso somente se ele estiver marcado com seu nome de usuário do IAM. Para obter mais informações, consulte Elementos de política do IAM: variáveis e tags no Guia do usuário do IAM.

A AWS oferece suporte a chaves de condição globais e chaves de condição específicas do serviço. Para ver todas as chaves de condição globais da AWS, consulte Chaves de contexto de condição globais da AWS no Guia do usuário do IAM.

Para visualizar uma lista de chaves de condição do Amazon EMR no EKS e aprender quais ações e recursos você pode usar uma chave de condição, consulte Ações, recursos e chaves de condição para o Amazon EMR no EKS na Referência de autorização do serviço.

Para visualizar exemplos de políticas baseadas em identidade do Amazon EMR no EKS, consulte Exemplos de políticas baseadas em identidade para o Amazon EMR no EKS.

Listas de controle de acesso (ACLs) no Amazon EMR no EKS

Oferece suporte a ACLs	Não

As listas de controle de acesso (ACLs) controlam quais entidades principais (membros, usuários ou funções da conta) têm permissões para acessar um recurso. As ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

Controle de acesso por atributo (ABAC) com o Amazon EMR no EKS

Oferece suporte a ABAC (tags em políticas)	Sim

O controle de acesso baseado em atributo (ABAC) é uma estratégia de autorização que define permissões com base em atributos. Na AWS, esses atributos são chamados de tags. É possível anexar tags a entidades do IAM (usuários ou funções) e a muitos recursos da AWS. A marcação de entidades e recursos é a primeira etapa do ABAC. Em seguida, você cria políticas de ABAC para permitir operações quando a tag da entidade principal corresponder à tag do recurso que ela está tentando acessar.

O ABAC é útil em ambientes que estão crescendo rapidamente e ajuda em situações em que o gerenciamento de políticas se torna um problema.

Para controlar o acesso baseado em tags, forneça informações sobre as tags no elemento de condição de uma política usando as `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou `aws:TagKeys` chaves de condição.

Se um serviço oferecer suporte às três chaves de condição para cada tipo de recurso, o valor será Yes (Sim) para o serviço. Se um serviço oferecer suporte às três chaves de condição somente para alguns tipos de recursos, o valor será Partial (Parcial).

Para obter mais informações sobre o ABAC, consulte O que é ABAC? no Guia do usuário do IAM. Para visualizar um tutorial com etapas para configurar o ABAC, consulte Usar controle de acesso baseado em atributos (ABAC) (Use attribute-based access control [ABAC]) no Guia do usuário do IAM.

Uso de credencial temporária com o Amazon EMR no EKS

Oferece suporte a credenciais temporárias	Sim

Alguns Serviços da AWS não funcionam quando você faz login usando credenciais temporárias. Para obter informações adicionais, incluindo quais Serviços da AWS funcionam com credenciais temporárias, consulte Serviços da AWS que funcionam com o IAM no Guia do usuário do IAM.

Você está usando credenciais temporárias se faz login no AWS Management Console usando qualquer método, exceto um nome de usuário e uma senha. Por exemplo, quando você acessa a AWS usando o link de autenticação única (SSO) da sua empresa, esse processo cria automaticamente credenciais temporárias. Você também cria automaticamente credenciais temporárias quando faz login no console como usuário e, em seguida, alterna funções. Para obter mais informações sobre como alternar funções, consulte Alternar para uma função (console) no Guia do usuário do IAM.

Você pode criar credenciais temporárias manualmente usando a AWS CLI ou a API da AWS. Em seguida, você pode usar essas credenciais temporárias para acessar a AWS. A AWS recomenda que você gere credenciais temporárias dinamicamente em vez de usar chaves de acesso de longo prazo. Para obter mais informações, consulte Credenciais de segurança temporárias no IAM.

Permissões de entidades principais entre serviços para o Amazon EMR no EKS

Oferece suporte a permissões de entidades	Sim

Quando você usa um usuário ou uma função do IAM para executar ações na AWS, você é considerado uma entidade principal. As políticas concedem permissões a uma entidade principal. Quando você usa alguns serviços, pode executar uma ação que, em seguida, aciona outra ação em outro serviço. Nesse caso, você deve ter permissões para executar ambas as ações. Para visualizar se uma ação requer ações dependentes adicionais em uma política, consulte Ações, recursos e chaves de condição para o Amazon EMR no EKS na Referência de autorização do serviço.

Perfis de serviço para o Amazon EMR no EKS

Oferece suporte a funções de serviço	Não

Perfis vinculados ao serviço para o Amazon EMR no EKS

Oferece suporte a funções vinculadas ao serviço	Sim

Para obter detalhes sobre como criar ou gerenciar funções vinculadas a serviços, consulte Serviços do AWS que funcionam com o IAM. Encontre um serviço na tabela que inclua um `Yes` na coluna Service-linked role (Função vinculada ao serviço). Escolha o link Sim para visualizar a documentação da função vinculada a serviço desse serviço.

Uso de perfis vinculados ao serviço para o Amazon EMR no EKS

O Amazon EMR no EKS usa perfis vinculados ao serviço do AWS Identity and Access Management (IAM). Um perfil vinculado ao serviço é um tipo exclusivo de perfil do IAM vinculado diretamente ao

Amazon EMR no EKS. Os perfis vinculados ao serviço são definidos previamente pelo Amazon EMR no EKS e incluem todas as permissões que o serviço requer para chamar outros serviços da AWS em seu nome.

Um perfil vinculado ao serviço facilita a configuração do Amazon EMR no EKS porque você não precisa adicionar manualmente as permissões necessárias. O Amazon EMR no EKS define as permissões dos perfis vinculados ao serviço e, a menos que definido de outra forma, somente o Amazon EMR no EKS pode assumir os perfis. As permissões definidas incluem a política de confiança e a política de permissões, e essa política não pode ser anexada a nenhuma outra entidade do IAM.

Uma função vinculada ao serviço poderá ser excluída somente após excluir seus recursos relacionados. Isso protege os recursos do Amazon EMR no EKS, pois você não pode remover inadvertidamente a permissão de acesso aos recursos.

Para obter informações sobre outros serviços compatíveis com funções vinculadas a serviços, consulte Serviços da AWS compatíveis com o IAM e procure os serviços que contenham Yes (Sim) na coluna Service-Linked Role (Função vinculada a serviço). Escolha um Sim com um link para visualizar a documentação da função vinculada a esse serviço.

Permissões de perfis vinculados ao serviço para o Amazon EMR no EKS

O Amazon EMR no EKS usa o perfil vinculado ao serviço chamado AWSServiceRoleForAmazonEMRContainers.

A função vinculada ao serviço `AWSServiceRoleForAmazonEMRContainers` confia nos seguintes serviços para assumir a função:

- `emr-containers.amazonaws.com`

A política de permissões de perfil `AmazonEMRContainersServiceRolePolicy` permite que o Amazon EMR no EKS conclua um conjunto de ações nos recursos especificados, como demonstra a instrução de política apresentada a seguir.

> **Note**
>
> Como o conteúdo da política gerenciada muda, a política mostrada aqui pode estar desatualizada. Confira a política AmazonEMRContainersServiceRolePolicy mais atualizada no AWS Management Console.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster",
                "eks:ListNodeGroups",
                "eks:DescribeNodeGroup",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "elasticloadbalancing:DescribeInstanceHealth",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeTargetHealth"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "acm:ImportCertificate",
                "acm:AddTagsToCertificate"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/emr-container:endpoint:managed-certificate": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "acm:DeleteCertificate"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/emr-container:endpoint:managed-certificate":
 "true"
                }
```

```
            }
        }
    ]
}
```

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua uma função vinculada ao serviço. Para obter mais informações, consulte Permissões de função vinculada ao serviço no Guia do usuário do IAM.

Criação de um perfil vinculado ao serviço para o Amazon EMR no EKS

Não é necessário criar manualmente uma função vinculada ao serviço. Ao criar um cluster virtual, o Amazon EMR no EKS cria o perfil vinculado ao serviço para você.

Se excluir essa função vinculada ao serviço e precisar criá-la novamente, você poderá usar esse mesmo processo para recriar a função em sua conta. Ao criar um cluster virtual, o Amazon EMR no EKS cria o perfil vinculado ao serviço para você novamente.

Você também pode usar o console do IAM para criar um perfil vinculado ao serviço com o caso de uso do Amazon EMR no EKS. Na AWS CLI ou na API do AWS, crie uma função vinculada ao serviço com o nome de serviço `emr-containers.amazonaws.com`. Para obter mais informações, consulte Criar uma função vinculada ao serviço no Guia do usuário do IAM. Se você excluir essa função vinculada ao serviço, será possível usar esse mesmo processo para criar a função novamente.

Edição de um perfil vinculado ao serviço para o Amazon EMR no EKS

O Amazon EMR no EKS não permite editar o perfil vinculado ao serviço `AWSServiceRoleForAmazonEMRContainers`. Depois que criar uma função vinculada ao serviço, você não poderá alterar o nome da função, pois várias entidades podem fazer referência a ela. No entanto, será possível editar a descrição da função usando o IAM. Para obter mais informações, consulte Editar uma função vinculada ao serviço no Guia do usuário do IAM.

Exclusão de um perfil vinculado ao serviço do Amazon EMR no EKS

Se você não precisar mais usar um recurso ou serviço que requer uma função vinculada a serviço, é recomendável excluí-la. Dessa forma, você não tem uma entidade não utilizada que não seja monitorada ativamente ou mantida. No entanto, você deve limpar os recursos de sua função vinculada ao serviço antes de excluí-la manualmente.

> **Note**
>
> Se o serviço Amazon EMR no EKS estiver usando o perfil quando você tentar excluir os recursos, a exclusão poderá falhar. Se isso acontecer, espere alguns minutos e tente a operação novamente.

Excluir recursos do Amazon EMR no EKS usados pelos **`AWSServiceRoleForAmazonEMRContainers`**

1. Abra o console do Amazon EMR.
2. Escolha um cluster virtual.
3. Na página `Virtual Cluster`, escolha Excluir.
4. Repita este procedimento para quaisquer outros clusters virtuais em sua conta.

Como excluir manualmente a função vinculada ao serviço usando o IAM

Use o console do IAM, a AWS CLI ou a API da AWS para excluir a função vinculada ao serviço `AWSServiceRoleForAmazonEMRContainers`. Para obter mais informações, consulte Excluir uma função vinculada ao serviço no Guia do usuário do IAM.

Regiões com suporte para perfis vinculados ao serviço do Amazon EMR no EKS

O Amazon EMR no EKS oferece suporte ao uso de perfis vinculados ao serviço em todas as regiões nas quais o serviço está disponível. Para obter mais informações, consulte Cotas e endpoints de serviço do Amazon EMR no EKS.

Políticas gerenciadas para o Amazon EMR no EKS

Visualize os detalhes sobre as atualizações nas políticas gerenciadas pela AWS para o Amazon EMR no EKS desde 1.º de março de 2021.

Alteração	Descrição	Data
`AmazonEMRContainersServiceRolePolicy` : permissões adicionad	As seguintes permissões são adicionadas à política: `eks:ListNodeGroups` , `eks:DescribeNodeGroup` ,	13 de março de 2023

Alteração	Descrição	Data
as para descrever e listar grupos de nós do Amazon EKS, descrever grupos de destino do balanceador de carga e descrever a integridade do destino do balanceador de carga.	`elasticloadbalancing:DescribeTargetGroups` e `elasticloadbalancing:DescribeTargetHealth` .	
`AmazonEMRContainersServiceRolePolicy` : permissões adicionadas para importar e excluir certificados no AWS Certificate Manager.	As seguintes permissões são adicionadas à política: `acm:ImportCertificate` , `acm:AddTagsToCertificate` e `acm:DeleteCertificate` .	3 de dezembro de 2021
O Amazon EMR no EKS começou a rastrear alterações	O Amazon EMR no EKS começou a rastrear alterações em suas políticas gerenciadas pela AWS.	1º de março de 2021

Uso de perfis de execução de trabalho com o Amazon EMR no EKS

Para usar o comando `StartJobRun` para enviar uma execução de trabalho em um cluster do EKS, primeiro é necessário integrar um perfil de execução de trabalho para ser usado com um cluster virtual. Para obter mais informações, consulte Criação de um perfil de execução de trabalho em Configuração do Amazon EMR no EKS. Você também pode seguir as instruções na seção Create IAM Role for job execution do Amazon EMR on EKS Workshop.

As permissões apresentadas a seguir devem ser inclusas na política de confiança para o perfil de execução de trabalho.

```
{
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Principal": {
```

```
        "Federated": "arn:aws:iam::AWS_ACCOUNT_ID:oidc-provider/OIDC_PROVIDER"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringLike": {
          "OIDC_PROVIDER:sub": "system:serviceaccount:NAMESPACE:emr-containers-sa-*-
*-AWS_ACCOUNT_ID-BASE36_ENCODED_ROLE_NAME"
        }
      }
    }
  ]
}
```

A política de confiança no exemplo anterior concede permissões somente para uma conta de serviço do Kubernetes gerenciada pelo Amazon EMR com um nome que corresponda ao padrão `emr-containers-sa-*-*-`*`AWS_ACCOUNT_ID`*`-`*`BASE36_ENCODED_ROLE_NAME`*. As contas de serviço com esse padrão serão criadas automaticamente no envio do trabalho e terão como escopo o namespace no qual você envia o trabalho. Esta política de confiança permite que estas contas de serviço assumam o perfil de execução e obtenham as credenciais temporárias para o perfil de execução. As contas de serviço de um cluster diferente do Amazon EKS ou de um namespace diferente no mesmo cluster do EKS estão impedidas de assumir o perfil de execução.

Você pode executar o comando apresentado a seguir para atualizar automaticamente a política de confiança no formato fornecido acima.

```
aws emr-containers update-role-trust-policy \
       --cluster-name cluster \
       --namespace namespace \
       --role-name iam_role_name_for_job_execution
```

Controle do acesso ao perfil de execução

Um administrador do cluster do Amazon EKS pode criar um cluster virtual multilocatário do Amazon EMR no EKS ao qual um administrador do IAM pode adicionar diversos perfis de execução. Como os locatários não confiáveis podem usar esses perfis de execução para enviar trabalhos que executam códigos arbitrários, você pode desejar restringir esses locatários para que eles não possam executar códigos que concedem as permissões atribuídas a um ou mais desses perfis de execução. Para restringir a política do IAM anexada a uma identidade do IAM, o administrador do IAM pode usar a chave de condição opcional `emr-containers:ExecutionRoleArn` do nome do recurso da

Amazon (ARN). Essa condição aceita uma lista de ARNs de perfis de execução que têm permissões para o cluster virtual, conforme demonstra a política de permissões a seguir.

```
  {
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "emr-containers:StartJobRun",
      "Resource": "arn:aws:emr-containers:REGION:AWS_ACCOUNT_ID:/
virtualclusters/VIRTUAL_CLUSTER_ID",
      "Condition": {
        "ArnEquals": {
          "emr-containers:ExecutionRoleArn": [
            "execution_role_arn_1",
            "execution_role_arn_2",
            ...
          ]
        }
      }
    }
  ]
}
```

Se desejar permitir todos os perfis de execução que começam com um prefixo específico, como `MyRole`, você poderá substituir o operador de condição `ArnEquals` pelo operador `ArnLike`, e poderá substituir o valor de `execution_role_arn` na condição por um caractere curinga `*`. Por exemplo, `arn:aws:iam::AWS_ACCOUNT_ID:role/MyRole*`. Todas as outras chaves de condição de ARN também são compatíveis.

> **Note**
>
> Com o Amazon EMR no EKS, não é possível conceder permissões para perfis de execução com base em etiquetas ou atributos. O Amazon EMR no EKS não oferece suporte ao controle de acesso por etiqueta (TBAC) ou ao controle de acesso por atributo (ABAC) para perfis de execução.

Exemplos de políticas baseadas em identidade para o Amazon EMR no EKS

Por padrão, os usuários e os perfis não têm permissão para criar ou modificar os recursos do Amazon EMR no EKS. Eles também não podem executar tarefas usando o AWS Management Console, a AWS Command Line Interface (AWS CLI) ou a AWS API. Para conceder aos usuários permissão para executar ações nos recursos de que eles precisam, um administrador do IAM pode criar políticas do IAM. O administrador pode então adicionar as políticas do IAM a perfis, e os usuários podem assumir os perfis.

Para saber como criar uma política baseada em identidade do IAM usando esses exemplos de documento de política JSON, consulte Criação de políticas do IAM no Guia do usuário do IAM.

Para obter detalhes sobre ações e tipos de recursos definidos pelo Amazon EMR no EKS, incluindo o formato dos ARNs para cada um dos tipos de recursos, consulte Ações, recursos e chaves de condição para o Amazon EMR no EKS na Referência de autorização do serviço.

Tópicos

- Práticas recomendadas de políticas
- Uso do console do Amazon EMR no EKS
- Permitir que os usuários visualizem suas próprias permissões

Práticas recomendadas de políticas

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir recursos do Amazon EMR no EKS em sua conta. Essas ações podem incorrer em custos para a Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com as políticas gerenciadas pela AWS e avance para as permissões de privilégio mínimo: para começar a conceder permissões a seus usuários e workloads, use as políticas gerenciadas pela AWS que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis na sua Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo cliente da AWS específicas para seus casos de uso. Para obter mais informações, consulte Políticas gerenciadas pela AWS ou Políticas gerenciadas pela AWS para funções de trabalho no Guia do usuário do IAM.
- Aplique permissões de privilégio mínimo: ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações

que podem ser executadas em recursos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para obter mais informações sobre como usar o IAM para aplicar permissões, consulte Políticas e permissões no IAM no Guia do usuário do IAM.

- Use condições nas políticas do IAM para restringir ainda mais o acesso: você pode adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, você pode escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usando SSL. Você também pode usar condições para conceder acesso a ações de serviço, se elas forem usadas por meio de um AWS service (Serviço da AWS) específico, como o AWS CloudFormation. Para obter mais informações, consulte Elementos de política JSON do IAM: Condition no Manual do usuário do IAM.
- Use o IAM Access Analyzer para validar suas políticas do IAM a fim de garantir permissões seguras e funcionais: o IAM Access Analyzer valida as políticas novas e existentes para que elas sigam a linguagem de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access Analyzer oferece mais de 100 verificações de política e recomendações acionáveis para ajudar você a criar políticas seguras e funcionais. Para obter mais informações, consulte Validação de políticas do IAM Access Analyzer no Guia do usuário do IAM.
- Require multi-factor authentication (MFA) (Exigir autenticação multifator (MFA)): se houver um cenário que exija usuários do IAM ou um usuário raiz em sua Conta da AWS, ative a MFA para obter segurança adicional. Para exigir a MFA quando as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para obter mais informações, consulte Configuração de acesso à API protegido por MFA no Guia do usuário do IAM.

Para obter mais informações sobre as práticas recomendadas do IAM, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.

Uso do console do Amazon EMR no EKS

Para acessar o console do Amazon EMR no EKS, você deve ter um conjunto mínimo de permissões. Essas permissões devem permitir que você liste e visualize detalhes sobre recursos do Amazon EMR no EKS em sua Conta da AWS. Se você criar uma política baseada em identidade que seja mais restritiva que as permissões mínimas necessárias, o console não funcionará como pretendido para entidades (usuários ou perfis) com essa política.

Não é necessário conceder permissões mínimas do console para usuários que fazem chamadas somente à AWS CLI ou à API do AWS. Em vez disso, permita o acesso somente às ações que correspondem à operação da API que estão tentando executar.

Para garantir que os usuários e os perfis ainda possam usar o console do Amazon EMR no EKS, anexe também as políticas gerenciadas pela AWS `ConsoleAccess` ou `ReadOnly` para o Amazon EMR no EKS às entidades. Para obter mais informações, consulte Adicionar permissões a um usuário no Guia do usuário do IAM.

Permitir que os usuários visualizem suas próprias permissões

Este exemplo mostra como você pode criar uma política que permite que os usuários do IAM visualizem as políticas gerenciadas e em linha anexadas a sua identidade de usuário. Essa política inclui permissões para concluir essa ação no console ou de forma programática usando a AWS CLI ou a API da AWS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

Políticas para controle de acesso baseado em etiquetas

É possível aplicar condições em sua política baseada em identidade para controlar o acesso a clusters virtuais e execuções de trabalhos com base em etiquetas. Para obter mais informações sobre marcação, consulte Marcação de recursos do Amazon EMR no EKS.

Os exemplos a seguir demonstram diferentes cenários e maneiras de usar operadores de condição com chaves de condição do Amazon EMR no EKS. Estas instruções de política do IAM são destinadas somente para fins de demonstração e não devem ser usadas em ambientes de produção. Há várias maneiras de combinar declarações de políticas para conceder e negar permissões de acordo com seus requisitos. Para obter mais informações sobre como planejar e testar políticas do IAM, consulte o Guia do usuário do IAM.

> **Important**
>
> Recusar, explicitamente, permissões para ações de uso de tags é uma consideração importante. Isso evita que os usuários façam a marcação de um recurso e, assim, concedam a si mesmos permissões que você não pretendia conceder. Se as ações de marcação de um recurso não forem negadas, um usuário poderá modificar as etiquetas e driblar a intenção das políticas baseadas em etiquetas. Para obter um exemplo de política que nega ações de marcação, consulte Negação de acesso para adição ou remoção de etiquetas.

Os exemplos abaixo demonstram políticas de permissões baseadas em identidade usadas para controlar as ações que são permitidas com clusters virtuais do Amazon EMR no EKS.

Ações permitidas somente em recursos com valores de etiquetas específicos

No exemplo de política apresentado a seguir, o operador de condição StringEquals tenta corresponder “dev” ao valor da etiqueta “Department”. Se a etiqueta “Department” não tiver sido adicionada ao cluster virtual ou não contiver o valor “dev”, a política não se aplicará e as ações não serão permitidas por esta política. Se nenhuma outra instrução de política permitir as ações, o usuário poderá trabalhar somente com clusters virtuais que tenham essa etiqueta com este valor.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:DescribeVirtualCluster"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/department": "dev"
        }
      }
    }
  ]
}
```

Você também pode especificar vários valores de tag usando um operador de condição. Por exemplo, para permitir ações em clusters virtuais nos quais a etiqueta `department` contém o valor `dev` ou `test`, você pode substituir o bloco condicional no exemplo anterior pelo apresentado a seguir.

```
"Condition": {
        "StringEquals": {
          "aws:ResourceTag/department": ["dev", "test"]
        }
      }
```

Marcação obrigatória na criação de um recurso

No exemplo abaixo, a etiqueta precisa ser aplicada ao criar o cluster virtual.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:CreateVirtualCluster"
      ],
      "Resource": "*",
      "Condition": {
```

```
        "StringEquals": {
          "aws:RequestTag/department": "dev"
        }
      }
    }
  ]
}
```

A instrução de política apresentada a seguir permite que um usuário crie um cluster virtual somente se o cluster tiver uma etiqueta `department`, que pode conter qualquer valor.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:CreateVirtualCluster"
      ],
      "Resource": "*",
      "Condition": {
        "Null": {
          "aws:RequestTag/department": "false"
        }
      }
    }
  ]
}
```

Negação de acesso para adição ou remoção de etiquetas

O efeito desta política é negar a um usuário a permissão para adicionar ou remover quaisquer etiquetas em clusters virtuais que estejam marcados com uma etiqueta `department` que contenha o valor `dev`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "emr-containers:TagResource",
```

```
        "emr-containers:UntagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceTag/department": "dev"
        }
      }
    }
  ]
}
```

Solução de problemas de identidade e acesso do Amazon EMR no EKS

Use as informações apresentadas a seguir para ajudar a diagnosticar e corrigir problemas comuns que você possa encontrar ao trabalhar com o Amazon EMR no EKS e com o IAM.

Tópicos

- Não tenho autorização para executar uma ação no Amazon EMR no EKS
- Não estou autorizado a executar iam:PassRole
- Desejo permitir que pessoas externas a minha conta da AWS acessem meus recursos do Amazon EMR no EKS

Não tenho autorização para executar uma ação no Amazon EMR no EKS

Se o AWS Management Console informar que você não está autorizado a executar uma ação, você deverá entrar em contato com o administrador para obter assistência. O administrador é a pessoa que forneceu a você o seu nome de usuário e senha.

O erro do exemplo a seguir ocorre quando o usuário `mateojackson` tenta usar o console para visualizar detalhes sobre um recurso do *`my-example-widget`* fictício, mas não tem as permissões fictícias do `emr-containers:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: emr-
containers:GetWidget on resource: my-example-widget
```

Neste caso, Mateo pede ao administrador para atualizar suas políticas para permitir a ele o acesso ao recurso *`my-example-widget`* usando a ação `emr-containers:`*`GetWidget`*.

Não estou autorizado a executar iam:PassRole

Se você receber uma mensagem de erro informando que não tem autorização para executar a ação `iam:PassRole`, suas políticas devem ser atualizadas para permitir a transmissão de um perfil ao Amazon EMR no EKS.

Alguns Serviços da AWS permitem que você transmita um perfil existente para o serviço, em vez de criar um perfil de serviço ou um perfil vinculado ao serviço. Para fazer isso, um usuário deve ter permissões para passar o perfil para o serviço.

O erro exemplificado a seguir ocorre quando uma usuária do IAM chamada `marymajor` tenta usar o console para executar uma ação no Amazon EMR no EKS. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar a função para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se você precisar de ajuda, entre em contato com seu administrador da AWS. Seu administrador é a pessoa que forneceu a você suas credenciais de login.

Desejo permitir que pessoas externas a minha conta da AWS acessem meus recursos do Amazon EMR no EKS

Você pode criar uma função que os usuários de outras contas ou pessoas fora da sua organização podem usar para acessar seus recursos. Você pode especificar quem é confiável para assumir a função. Para serviços que oferecem suporte a políticas baseadas em recursos ou listas de controle de acesso (ACLs), você pode usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte o seguinte:

- Para saber se o Amazon EMR no EKS oferece suporte a esses recursos, consulte Como o Amazon EMR no EKS funciona com o IAM.
- Para saber como conceder acesso a seus recursos em todas as Contas da AWS pertencentes a você, consulte Fornecimento de acesso a um usuário do IAM em outra Conta da AWS pertencente a você no Guia de usuário do IAM.

- Para saber como conceder acesso a seus recursos para terceiros Contas da AWS, consulte Fornecimento de acesso a Contas da AWS pertencentes a terceiros no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte Conceder acesso a usuários autenticados externamente (federação de identidades) no Guia do usuário do IAM.
- Para saber a diferença entre usar perfis e políticas baseadas em recursos para acesso entre contas, consulte Como os perfis do IAM diferem de políticas baseadas em recursos no Guia do usuário do IAM.

Registro e monitoramento

Para detectar incidentes, receber alertas quando incidentes ocorrerem e responder a eles, use estas opções com o Amazon EMR no EKS:

- Monitorar o Amazon EMR no EKS com o AWS CloudTrail: o AWS CloudTrail fornece um registro das ações realizadas por um usuário, um perfil ou um serviço da AWS no Amazon EMR no EKS. Ele captura as chamadas do console do Amazon EMR e as chamadas de código para as operações de API do Amazon EMR no EKS como eventos. Dessa forma, você pode determinar a solicitação feita ao Amazon EMR no EKS, o endereço IP do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita e detalhes adicionais. Para obter mais informações, consulte Registro em log de chamadas de API do Amazon EMR no EKS usando o AWS CloudTrail.
- Usar o CloudWatch Events com o Amazon EMR no EKS: o CloudWatch Events disponibiliza um fluxo quase em tempo real de eventos do sistema que descrevem alterações nos recursos da AWS. O CloudWatch Events reconhece essas alterações operacionais logo que acontecem, respondem a elas e executa a ação corretiva conforme necessário, enviando mensagens para responder ao ambiente, ativando funções, fazendo alterações e capturando informações de estado. Para usar o CloudWatch Events com o Amazon EMR no EKS, crie uma regra que seja acionada em uma chamada de API do Amazon EMR no EKS usando o CloudTrail. Para obter mais informações, consulte Monitoramento de trabalhos com o Amazon CloudWatch Events.

Registro em log de chamadas de API do Amazon EMR no EKS usando o AWS CloudTrail

O Amazon EMR no EKS está integrado ao AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, um perfil ou um serviço da AWS no Amazon EMR no EKS.

O CloudTrail captura todas as chamadas de API para o Amazon EMR no EKS como eventos. As chamadas capturadas incluem as chamadas do console do Amazon EMR no EKS e as chamadas de código para as operações de API do Amazon EMR no EKS. Se você criar uma trilha, poderá habilitar a entrega contínua de eventos do CloudTrail para um bucket do Amazon S3, incluindo eventos para o Amazon EMR no EKS. Se você não configurar uma trilha, ainda poderá visualizar os eventos mais recentes no console do CloudTrail em Event history (Histórico de eventos). Com as informações coletadas pelo CloudTrail, é possível determinar a solicitação feita ao Amazon EMR no EKS, o endereço IP do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita e detalhes adicionais.

Para saber mais sobre o CloudTrail, consulte o Guia do usuário do AWS CloudTrail.

Informações do Amazon EMR no EKS no CloudTrail

O CloudTrail é habilitado em sua conta da AWS quando ela é criada. Quando ocorre uma atividade no Amazon EMR no EKS, essa atividade é registrada em um evento do CloudTrail junto com outros eventos de serviços da AWS em Histórico de eventos. Você pode visualizar, pesquisar e baixar eventos recentes em sua conta da AWS. Para obter mais informações, consulte Viewing events with CloudTrail Event history (Como visualizar eventos com o histórico de eventos do CloudTrail).

Para obter um registro contínuo de eventos na sua conta da AWS, incluindo eventos para o Amazon EMR no EKS, crie uma trilha. Uma trilha permite que o CloudTrail entregue arquivos de log a um bucket do Amazon S3. Por padrão, quando você cria uma trilha no console, ela é aplicada a todas as regiões da AWS. A trilha registra em log eventos de todas as regiões na partição da AWS e entrega os arquivos de log para o bucket do Amazon S3 especificado por você. Além disso, é possível configurar outros serviços da AWS para analisar mais ainda mais e agir com base nos dados de eventos coletados nos logs do CloudTrail. Para obter mais informações, consulte:

- Visão geral da criação de uma trilha
- Serviços e integrações compatíveis com o CloudTrail
- Configurar notificações do Amazon SNS para o CloudTrail
- Receber arquivos de log do CloudTrail de várias regiões e Receber arquivos de log do CloudTrail de várias contas

Todas as ações do Amazon EMR no EKS são registradas em log pelo CloudTrail e documentadas na documentação de API do Amazon EMR no EKS. Por exemplo, as chamadas para as ações `CreateVirtualCluster`, `StartJobRun` e `ListJobRuns` geram entradas nos arquivos de log do CloudTrail.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar:

- Se a solicitação foi feita com credenciais de usuário raiz ou do AWS Identity and Access Management (IAM).
- Se a solicitação foi feita com credenciais de segurança temporárias de uma função ou de um usuário federado.
- Se a solicitação foi feita por outro serviço da AWS.

Para obter mais informações, consulte CloudTrail userIdentity element.

Noções básicas sobre entradas de arquivos de log do Amazon EMR no EKS

Uma trilha é uma configuração que permite a entrega de eventos como arquivos de log a um bucket do Amazon S3 especificado. Os arquivos de log do CloudTrail contêm uma ou mais entradas de log. Um evento representa uma única solicitação de qualquer fonte e inclui informações sobre a ação solicitada, a data e a hora da ação, os parâmetros de solicitação e assim por diante. Os arquivos de log do CloudTrail não são um rastreamento de pilha ordenada de chamadas de API pública. Dessa forma, eles não são exibidos em uma ordem específica.

O exemplo a seguir mostra uma entrada de log do CloudTrail que demonstra a ação `ListJobRuns`.


```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE:admin",
    "arn": "arn:aws:sts::012345678910:assumed-role/Admin/admin",
    "accountId": "012345678910",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::012345678910:role/Admin",
        "accountId": "012345678910",
        "userName": "Admin"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
```

```
        "creationDate": "2020-11-04T21:49:36Z"
      }
    }
  },
  "eventTime": "2020-11-04T21:52:58Z",
  "eventSource": "emr-containers.amazonaws.com",
  "eventName": "ListJobRuns",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "203.0.113.1",
  "userAgent": "aws-cli/1.11.167 Python/2.7.10 Darwin/16.7.0 botocore/1.7.25",
  "requestParameters": {
    "virtualClusterId": "1K48XXXXXXHCB"
  },
  "responseElements": null,
  "requestID": "890b8639-e51f-11e7-b038-EXAMPLE",
  "eventID": "874f89fa-70fc-4798-bc00-EXAMPLE",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "recipientAccountId": "012345678910"
}
```


Validação de conformidade para o Amazon EMR no EKS

Auditores terceirizados avaliam a segurança e a conformidade do Amazon EMR no EKS como parte de diversos programas de conformidade da AWS. Isso inclui SOC, PCI, FedRAMP, HIPAA e outros.

Resiliência no Amazon EMR no EKS

A infraestrutura global da AWS é criada com base em regiões da AWS e zonas de disponibilidade. As regiões da AWS As regiões fornecem várias zonas de disponibilidade separadas e isoladas fisicamente, que são conectadas com baixa latência, throughput elevada e redes altamente redundantes. Com as zonas de disponibilidade, é possível projetar e operar aplicações e bancos de dados que automaticamente executam o failover entre as zonas sem interrupção. As zonas de disponibilidade são mais altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data center tradicionais.

Para obter mais informações sobre regiões e zonas de disponibilidade da AWS, consulte Infraestrutura global da AWS.

Além da infraestrutura global da AWS, o Amazon EMR no EKS oferece integração com o Amazon S3 por meio do EMRFS para ajudar a apoiar suas necessidades de backup e de resiliência de dados.

Segurança da infraestrutura no Amazon EMR no EKS

Como um serviço gerenciado, o Amazon EMR no EKS é protegido pelos procedimentos de segurança de rede global da AWS descritos no whitepaper Amazon Web Services: Overview of Security Processes.

Você usa chamadas de API publicadas pela AWS para acessar o Amazon EMR no EKS usando a rede. Os clientes devem oferecer suporte a Transport Layer Security (TLS) 1.0 ou posterior. Recomendamos TLS 1.2 ou posterior. Os clientes também devem ter suporte a conjuntos de criptografia com perfect forward secrecy (PFS) como Ephemeral Diffie-Hellman (DHE) ou Ephemeral Elliptic Curve Diffie-Hellman (ECDHE). A maioria dos sistemas modernos como Java 7 e versões posteriores oferece suporte a esses modos.

Além disso, as solicitações devem ser assinadas usando um ID da chave de acesso e uma chave de acesso secreta associada a uma entidade principal do IAM. Ou você pode usar o AWS Security Token Service (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

Análise de configuração e vulnerabilidade

A AWS se encarrega das tarefas básicas de segurança, como aplicação de patches a bancos de dados e sistemas operacionais (SOs) convidados, configuração de firewalls e recuperação de desastres. Esses procedimentos foram revisados e certificados por terceiros certificados. Para obter mais detalhes, consulte os seguintes recursos da :

- Validação de conformidade para o Amazon EMR no EKS
- Modelo de responsabilidade compartilhada
- Amazon Web Services: visão geral do processo de segurança (whitepaper)

Conexão com o Amazon EMR no EKS usando um endpoint da VPC de interface

É possível se conectar diretamente ao Amazon EMR no EKS usando endpoints da VPC de interface (AWS PrivateLink) em sua nuvem privada virtual (VPC), em vez de se conectar usando a Internet. Quando você usa um endpoint da VPC de interface, a comunicação entre a VPC e o Amazon EMR no EKS é realizada inteiramente dentro da rede da AWS. Cada endpoint da VPC é representado por uma ou mais interfaces de rede elástica (ENIs) com endereços IP privados nas sub-redes da VPC.

O endpoint da VPC de interface conecta a VPC diretamente ao Amazon EMR no EKS sem a necessidade de um gateway da Internet, um dispositivo NAT, uma conexão VPN ou uma conexão do AWS Direct Connect. As instâncias em sua VPC não precisam de endereços IP públicos para se comunicarem com a API do Amazon EMR no EKS.

É possível criar um endpoint da VPC de interface para se conectar ao Amazon EMR no EKS usando o AWS Management Console ou os comandos da AWS Command Line Interface (AWS CLI). Para obter mais informações, consulte Criação de um endpoint de interface.

Após criar um endpoint da VPC de interface, se você habilitar nomes de host DNS privados para o endpoint, o endpoint padrão do Amazon EMR no EKS será resolvido para seu endpoint da VPC. O endpoint de nome de serviço padrão para o Amazon EMR no EKS estará no formato a seguir.

```
emr-containers.Region.amazonaws.com
```

Se você não habilitar nomes de host DNS privados, a Amazon VPC fornecerá um nome de endpoint DNS que poderá ser usado no formato a seguir.

```
VPC_Endpoint_ID.emr-containers.Region.vpce.amazonaws.com
```

Para obter mais informações, consulte Interface VPC Endpoints (AWS PrivateLink) no Guia do usuário da Amazon VPC. O Amazon EMR no EKS oferece suporte a chamadas para todas as ações de API dentro da sua VPC.

Você pode anexar políticas de endpoint da VPC a um endpoint da VPC para controlar o acesso de entidades principais do IAM. Também é possível associar grupos de segurança a um VPC endpoint para controlar o acesso de entrada e saída com base na origem e no destino do tráfego de rede, como um intervalo de endereços IP. Para obter mais informações, consulte Controlar o acesso a serviços com endpoints da VPC.

Criação de uma política de endpoint da VPC para o Amazon EMR no EKS

É possível criar uma política para endpoints da Amazon VPC para o Amazon EMR no EKS com a finalidade de especificar o seguinte:

- O principal que pode ou não executar ações
- As ações que podem ser executadas
- Os recursos nos quais as ações podem ser executadas

Para obter mais informações, consulte Controlling Access to Services with VPC Endpoints no Guia do usuário da Amazon VPC.

Example Política de endpoint da VPC para negar todo o acesso de uma conta da AWS especificada

A política de VPC endpoint a seguir nega à conta da AWS *123456789012* todos os acessos aos recursos que usam o endpoint.

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "*",
            "Effect": "Deny",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
            }
        }
    ]
}
```

Example Política de endpoint da VPC para permitir o acesso à VPC somente a uma entidade principal do IAM (usuário) especificada

A política de endpoint da VPC a seguir permite o acesso total somente ao usuário do IAM *lijuan* na conta da AWS *123456789012*. Todos os outros principais IAM têm acesso negado usando o endpoint.

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
```

```
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123456789012:user/lijuan"
                ]
            }
        }
    ]
}
```

Example Política de endpoint da VPC para permitir operações somente leitura do Amazon EMR no EKS

A política de endpoint da VPC a seguir permite que somente a conta da AWS *123456789012* execute as ações especificadas do Amazon EMR no EKS.

As ações especificadas fornecem o equivalente ao acesso somente leitura para o Amazon EMR no EKS. Todas as outras ações na VPC serão negadas para a conta especificada. Todas as outras contas terão acesso negado. Para obter uma lista de ações do Amazon EMR no EKS, consulte Ações, recursos e chaves de condição para o Amazon EMR no EKS.

```
{
    "Statement": [
        {
            "Action": [
                "emr-containers:DescribeJobRun",
                "emr-containers:DescribeVirtualCluster",
                "emr-containers:ListJobRuns",
                "emr-containers:ListTagsForResource",
                "emr-containers:ListVirtualClusters"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
            }
        }
    ]
}
```

Example Política de endpoint da VPC com negação de acesso a um cluster virtual especificado

A política de endpoint da VPC a seguir permite acesso total para todas as contas e entidades principais, mas nega qualquer acesso para a conta da AWS *123456789012* a ações executadas no cluster virtual com ID de cluster *A1B2CD34EF5G*. Outras ações do Amazon EMR no EKS que não oferecem suporte a permissões em nível de recurso para clusters virtuais ainda são permitidas. Para obter uma lista de ações do Amazon EMR no EKS e seus tipos de recursos correspondentes, consulte Ações, recursos e chaves de condição para o Amazon EMR no EKS no Guia do usuário do AWS Identity and Access Management.

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "*",
            "Effect": "Deny",
            "Resource": "arn:aws:emr-containers:us-west-2:123456789012:/
virtualclusters/A1B2CD34EF5G",
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
            }
        }
    ]
}
```

Configuração do acesso entre contas para o Amazon EMR no EKS

Você pode configurar o acesso entre contas para o Amazon EMR no EKS. O acesso entre contas possibilita que os usuários de uma conta da AWS executem trabalhos do Amazon EMR no EKS e acessem os dados subjacentes que pertencem a outra conta da AWS.

Pré-requisitos

Para configurar o acesso entre contas para o Amazon EMR no EKS, você concluirá tarefas enquanto estiver conectado às seguintes contas da AWS:

- `AccountA`: uma conta da AWS na qual você criou um cluster virtual do Amazon EMR no EKS ao registrar o Amazon EMR com um namespace em um cluster do EKS.
- `AccountB`: uma conta da AWS que contém um bucket do Amazon S3 ou uma tabela do DynamoDB que você deseja que os trabalhos do Amazon EMR no EKS acessem.

Você deve ter o seguinte em suas contas da AWS antes de configurar o acesso entre contas:

- Um cluster virtual do Amazon EMR no EKS na `AccountA` em que deseja executar trabalhos.
- Um perfil de execução de trabalho na `AccountA` que tem as permissões obrigatórias para executar trabalhos no cluster virtual. Para obter mais informações, consulte Criação de um perfil de execução de trabalho e Uso de perfis de execução de trabalho com o Amazon EMR no EKS.

Como acessar um bucket do Amazon S3 ou uma tabela do DynamoDB entre contas

Para configurar o acesso entre contas do Amazon EMR no EKS, conclua as etapas apresentadas a seguir.

1. Crie um bucket do Amazon S3, `cross-account-bucket`, na `AccountB`. Para mais informações, consulte Criar um bucket. Se desejar ter acesso entre contas para o DynamoDB, você também pode criar uma tabela do DynamoDB na `AccountB`. Para obter mais informações, consulte Creating a DynamoDB table.
2. Crie um perfil do IAM `Cross-Account-Role-B` na `AccountB` que possa acessar o `cross-account-bucket`.
    1. Faça login no console do IAM.
    2. Escolha Perfis e crie um novo perfil: `Cross-Account-Role-B`. Para obter mais informações sobre como criar perfis do IAM, consulte Criação de perfis do IAM no Guia do usuário do IAM.
    3. Crie uma política do IAM que especifique as permissões para `Cross-Account-Role-B` acessar o bucket `cross-account-bucket` do S3, como demonstra a instrução de política a

seguir. Em seguida, anexe a política do IAM ao `Cross-Account-Role-B`. Para obter mais informações, consulte Creating a New Policy no Guia do usuário do IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::cross-account-bucket",
                "arn:aws:s3:::cross-account-bucket/*"
            ]
        }
    ]
}
```

Se o acesso ao DynamoDB for necessário, crie uma política do IAM que especifique as permissões de acesso à tabela do DynamoDB entre contas. Em seguida, anexe a política do IAM ao `Cross-Account-Role-B`. Para obter mais informações, consulte Criar uma tabela do DynamoDB no Guia do usuário do IAM.

A seguir, é apresentada uma política de acesso a uma tabela do DynamoDB, `CrossAccountTable`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "dynamodb:*",
            "Resource": "arn:aws:dynamodb:MyRegion:AccountB:table/
CrossAccountTable"
        }
    ]
}
```

3. Edite a relação de confiança para o perfil `Cross-Account-Role-B`.

    1. Para configurar a relação de confiança para o perfil, escolha a guia Relações de confiança no console do IAM para o perfil criado na Etapa 2: `Cross-Account-Role-B`.

2. Selecione Editar relação de confiança.
3. Adicione o documento de política a seguir, que permite que `Job-Execution-Role-A` na `AccountA` assuma esse perfil `Cross-Account-Role-B`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::AccountA:role/Job-Execution-Role-A"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

4. Conceda que o `Job-Execution-Role-A` na `AccountA` tenha a permissão sts assume-role para assumir `Cross-Account-Role-B`.
   1. No console do IAM para a conta `AccountA` da AWS, selecione `Job-Execution-Role-A`.
   2. Adicione a instrução de política a seguir ao `Job-Execution-Role-A` para permitir a ação `AssumeRole` no perfil `Cross-Account-Role-B`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::AccountB:role/Cross-Account-Role-B"
        }
    ]
}
```

5. Para obter acesso ao Amazon S3, defina os parâmetros `spark-submit` (`spark conf`) apresentados a seguir ao enviar o trabalho para o Amazon EMR no EKS.

> **Note**
>
> Por padrão, o EMRFS usa o perfil de execução do trabalho para acessar o bucket do S3 usando o trabalho. Entretanto, quando `customAWSCredentialsProvider` é definido como `AssumeRoleAWSCredentialsProvider`, o EMRFS usa o perfil correspondente que você especifica com `ASSUME_ROLE_CREDENTIALS_ROLE_ARN` em vez do `Job-Execution-Role-A` para obter acesso ao Amazon S3.

- `--conf spark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRol`
- `--conf spark.kubernetes.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN=arn:aws:iam::Ac Cross-Account-Role-B \`
- `--conf spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN=arn:aws:iam::AccountB:ro Cross-Account-Role-B \`

> **Note**
>
> Você deve definir `ASSUME_ROLE_CREDENTIALS_ROLE_ARN` para o `env` do executor e do driver na configuração de trabalho do Spark.

Para obter acesso entre contas do DynamoDB, você deve definir `--conf spark.dynamodb.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRoleAWSCr`

6. Execute o trabalho do Amazon EMR no EKS com acesso entre contas, como demonstrado pelo exemplo a seguir.

```
aws emr-containers start-job-run \
--virtual-cluster-id 123456 \
--name myjob \
--execution-role-arn execution-role-arn \
--release-label emr-6.2.0-latest \
--job-driver '{"sparkSubmitJobDriver": {"entryPoint": "entryPoint_location",
 "entryPointArguments": ["arguments_list"], "sparkSubmitParameters": "--class
```

```
 <main_class> --conf spark.executor.instances=2 --conf spark.executor.memory=2G
 --conf spark.executor.cores=2 --conf spark.driver.cores=1 --conf
 spark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRoleAWSCredentials
 --conf
 spark.kubernetes.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN=arn:aws:iam::AccountB:role/
Cross-Account-Role-B --conf
 spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN=arn:aws:iam::AccountB:role/
Cross-Account-Role-B"}} ' \
--configuration-overrides '{"applicationConfiguration": [{"classification":
 "spark-defaults", "properties": {"spark.driver.memory": "2G"}}],
 "monitoringConfiguration": {"cloudWatchMonitoringConfiguration":
 {"logGroupName": "log_group_name", "logStreamNamePrefix": "log_stream_prefix"},
 "persistentAppUI":"ENABLED",  "s3MonitoringConfiguration": {"logUri": "s3://
my_s3_log_location" }}}'
```

Marcação de recursos do Amazon EMR no EKS

Para ajudar no gerenciamento de recursos do Amazon EMR no EKS, você pode atribuir seus próprios metadados a cada recurso usando etiquetas. Este tópico fornece uma visão geral da função das etiquetas e mostra como você pode criá-las.

Tópicos

- Conceitos básicos de tags
- Marcar com tag os recursos do
- Restrições de tags
- Trabalho com etiquetas usando a AWS CLI e a API do Amazon EMR no EKS

Conceitos básicos de tags

Uma etiqueta é um rótulo atribuído a um recurso da AWS. Cada tag consiste em uma chave e um valor opcional, ambos definidos por você.

As etiquetas possibilitam categorizar os recursos da AWS por atributos como finalidade, proprietário ou ambiente. Quando você tem muitos recursos do mesmo tipo; é possível identificar rapidamente um recurso específico com base nas tags que você atribuiu a ele. Por exemplo, é possível definir um conjunto de etiquetas para os clusters do Amazon EMR no EKS para ajudar a rastrear o proprietário e o nível da pilha de cada cluster. Recomendamos planejar um conjunto consistente de chaves de etiquetas para cada tipo de recurso. Você pode pesquisar e filtrar os recursos de acordo com as tags que adicionar.

Além disso, as tags não são automaticamente atribuídas aos recursos. Depois de adicionar uma tag, você pode editar as chaves e os valores das tags ou remover tags de um recurso a qualquer momento. Se você excluir um recurso, todas as tags do recurso também serão excluídas.

As etiquetas não têm significado semântico para o Amazon EMR no EKS e são interpretadas estritamente como uma string de caracteres.

Um valor de tag pode ser uma string vazia, mas não nula. Uma chave de etiqueta não pode ser uma string vazia. Se você adicionar uma etiqueta que tenha a mesma chave de uma etiqueta existente nesse recurso, o novo valor substituirá o antigo.

Se você estiver usando o AWS Identity and Access Management (IAM), poderá controlar quais usuários na sua conta da AWS têm permissão para gerenciar etiquetas.

Para obter exemplos de políticas de controle de acesso por etiquetas, consulte Políticas para controle de acesso baseado em etiquetas.

Marcar com tag os recursos do

Você pode etiquetar clusters virtuais e execuções de trabalhos novos ou existentes que estão em estados ativos. Os estados ativos para execuções de trabalhos incluem: `PENDING`, `SUBMITTED`, `RUNNING` e `CANCEL_PENDING`. Os estados ativos para clusters virtuais incluem: `RUNNING`, `TERMINATING` e `ARRESTED`. Para obter mais informações, consulte Estados de execução de trabalho e Estados de um cluster virtual.

Quando um cluster virtual é encerrado, as etiquetas são limpas e não ficam mais acessíveis.

Se você estiver usando a API do Amazon EMR no EKS, a AWS CLI ou um AWS SDK, poderá aplicar etiquetas a novos recursos usando o parâmetro de etiquetas na ação da API relevante. Você também pode aplicar etiquetas aos recursos usando a ação da API `TagResource`.

Você pode usar algumas ações de criação de recursos para especificar etiquetas para um recurso quando ele for criado. Nesse caso, se as etiquetas não puderem ser aplicadas enquanto o recurso estiver sendo criado, ele não será criado. Esse mecanismo garante que os recursos que você pretende etiquetar na criação sejam criados com as etiquetas especificadas ou não sejam criados. Se você etiquetar os recursos no momento da criação, não precisará executar os scripts de marcação personalizados após a criação do recurso.

A tabela a seguir descreve os recursos do Amazon EMR no EKS que podem ser etiquetados.

Recurso	Compatível com tags	Oferece suporte à propagação de tags	Oferece suporte à marcação na criação (API do Amazon EMR no EKS, AWS CLI e AWS SDK)	API para criação (etiquetas podem ser adicionadas durante a criação)
Cluster virtual	Sim	Não. As etiquetas	Sim	CreateVirtualCluster

Recurso	Compatível com tags	Oferece suporte à propagação de tags	Oferece suporte à marcação na criação (API do Amazon EMR no EKS, AWS CLI e AWS SDK)	API para criação (etiquetas podem ser adicionadas durante a criação)
		associadas a um cluster virtual não se propagam para execuções de trabalhos enviadas a esse cluster virtual.		
Execuções de trabalhos	Sim	Não	Sim	StartJobRun

Restrições de tags

As restrições básicas a seguir se aplicam às tags:

- Número máximo de tags por recurso: 50
- Em todos os recursos, cada chave de etiqueta deve ser exclusiva e pode ter apenas um valor.
- Comprimento máximo da chave: 128 caracteres Unicode em UTF-8
- Comprimento máximo do valor: 256 caracteres Unicode em UTF-8
- Se seu esquema de marcação for usado em vários serviços e recursos da AWS, lembre-se de que outros serviços podem ter restrições nos caracteres permitidos. Em geral, os caracteres permitidos são: letras, números, espaços representáveis em UTF-8 e os seguintes caracteres: + - = . _ : / @.
- As chaves e os valores de tags diferenciam maiúsculas de minúsculas.
- Um valor de tag pode ser uma string vazia, mas não nula. Uma chave de etiqueta não pode ser uma string vazia.

- Não use `aws:`, `AWS:` ou qualquer combinação de letras maiúsculas e minúsculas como prefixo para chaves ou valores. Esses são reservados para uso pela AWS.

Trabalho com etiquetas usando a AWS CLI e a API do Amazon EMR no EKS

Use os comandos apresentados a seguir da AWS CLI ou as operações da API do Amazon EMR no EKS para adicionar, atualizar, listar e excluir as etiquetas dos seus recursos.

Tarefa	AWS CLI	Ação de API
Adicione ou substitua uma ou mais tags	tag-resource	TagResource
Listar as tags de um recurso	list-tags-for-resource	ListTagsForResource
Exclua uma ou mais tags	untag-resource	UntagResource

Os exemplos a seguir mostram como marcar ou desmarcar recursos usando a AWS CLI.

Exemplo 1: etiquetar um cluster virtual existente

O comando a seguir realiza a marcação de um cluster virtual existente.

```
aws emr-containers tag-resource --resource-arn resource_ARN --tags team=devs
```

Exemplo 2: remover a etiqueta de um cluster virtual existente

O comando a seguir exclui uma etiqueta de um cluster virtual existente.

```
aws emr-containers untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

Exemplo 3: listar etiquetas para um recurso

O comando a seguir lista as tags associadas a um recurso existente.

```
aws emr-containers list-tags-for-resource --resource-arn resource_ARN
```

Solução de problemas para o Amazon EMR no EKS

Esta seção descreve como solucionar problemas com o Amazon EMR no EKS. Para obter informações sobre como solucionar problemas gerais com o Amazon EMR, consulte Troubleshoot a cluster no Guia de gerenciamento do Amazon EMR.

Tópicos

- Solução de problemas de trabalhos que usam PersistentVolumeClaims (PVC)
- Solução de problemas de escalabilidade automática vertical do Amazon EMR no EKS
- Solução de problemas do operador do Spark do Amazon EMR no EKS

Solução de problemas de trabalhos que usam PersistentVolumeClaims (PVC)

Se você precisar criar, listar ou excluir PersistentVolumeClaims (PVC) para um trabalho, mas não adicionar permissões de PVC ao perfil padrão do Kubernetes emr-containers, o trabalho falhará quando você o enviar. Sem essas permissões, o perfil emr-containers não pode criar os perfis necessários para o driver do Spark ou para o cliente do Spark. Não é suficiente adicionar permissões ao driver do Spark ou aos perfis de clientes, conforme sugerido pelas mensagens de erro. O perfil primário emr-containers também deve incluir as permissões necessárias. Esta seção explica como adicionar as permissões necessárias ao perfil primário emr-containers.

Verificação

Para verificar se o perfil emr-containers tem ou não as permissões necessárias, defina a variável NAMESPACE com seu próprio valor e execute o seguinte comando:

```
export NAMESPACE=YOUR_VALUE
kubectl describe role emr-containers -n ${NAMESPACE}
```

Além disso, para verificar se os perfis do Spark e dos clientes têm as permissões necessárias, execute o seguinte comando:

```
kubectl describe role emr-containers-role-spark-driver -n ${NAMESPACE}
kubectl describe role emr-containers-role-spark-client -n ${NAMESPACE}
```

Se as permissões não existirem, prossiga com a aplicação de patch, como apresentado a seguir.

Patch

1. Se os trabalhos estiverem em execução no momento sem as permissões, interrompa-os.
2. Crie um arquivo chamado RBAC_Patch.py da seguinte forma:

```
import os
import subprocess as sp
import tempfile as temp
import json
import argparse
import uuid

def delete_if_exists(dictionary: dict, key: str):
    if dictionary.get(key, None) is not None:
        del dictionary[key]

def doTerminalCmd(cmd):
    with temp.TemporaryFile() as f:
        process = sp.Popen(cmd, stdout=f, stderr=f)
        process.wait()
        f.seek(0)
        msg = f.read().decode()
    return msg

def patchRole(roleName, namespace, extraRules, skipConfirmation=False):
    cmd = f"kubectl get role {roleName} -n {namespace} --output json".split(" ")
    msg = doTerminalCmd(cmd)
    if "(NotFound)" in msg and "Error" in msg:
        print(msg)
        return False
    role = json.loads(msg)
    rules = role["rules"]
    rulesToAssign = extraRules[::]
    passedRules = []
    for rule in rules:
        apiGroups = set(rule["apiGroups"])
        resources = set(rule["resources"])
        verbs = set(rule["verbs"])
        for extraRule in extraRules:
            passes = 0
            apiGroupsExtra = set(extraRule["apiGroups"])
```

```
            resourcesExtra = set(extraRule["resources"])
            verbsExtra = set(extraRule["verbs"])
            passes += len(apiGroupsExtra.intersection(apiGroups)) >=
len(apiGroupsExtra)
            passes += len(resourcesExtra.intersection(resources)) >=
len(resourcesExtra)
            passes += len(verbsExtra.intersection(verbs)) >= len(verbsExtra)
            if passes >= 3:
                if extraRule not in passedRules:
                    passedRules.append(extraRule)
                    if extraRule in rulesToAssign:
                        rulesToAssign.remove(extraRule)
                break
    prompt_text = "Apply Changes?"
    if len(rulesToAssign) == 0:
        print(f"The role {roleName} seems to already have the necessary
permissions!")
        prompt_text = "Proceed anyways?"
    for ruleToAssign in rulesToAssign:
        role["rules"].append(ruleToAssign)
    delete_if_exists(role, "creationTimestamp")
    delete_if_exists(role, "resourceVersion")
    delete_if_exists(role, "uid")
    new_role = json.dumps(role, indent=3)
    uid = uuid.uuid4()
    filename = f"Role-{roleName}-New_Permissions-{uid}-TemporaryFile.json"
    try:
        with open(filename, "w+") as f:
            f.write(new_role)
            f.flush()
        prompt = "y"
        if not skipConfirmation:
            prompt = input(
                doTerminalCmd(f"kubectl diff -f {filename}".split(" ")) +
f"\n{prompt_text} y/n: "
            ).lower().strip()
            while prompt != "y" and prompt != "n":
                prompt = input("Please make a valid selection. y/n:
").lower().strip()
        if prompt == "y":
            print(doTerminalCmd(f"kubectl apply -f {filename}".split(" ")))
    except Exception as e:
        print(e)
    os.remove(f"./{filename}")
```

```
if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument("-n", "--namespace",
                        help="Namespace of the Role. By default its the
 VirtualCluster's namespace",
                        required=True,
                        dest="namespace"
                        )

    parser.add_argument("-p", "--no-prompt",
                        help="Applies the patches without asking first",
                        dest="no_prompt",
                        default=False,
                        action="store_true"
                        )
    args = parser.parse_args()

    emrRoleRules = [
        {
            "apiGroups": [""],
            "resources": ["persistentvolumeclaims"],
            "verbs": ["list", "create", "delete"]
         }

    ]

    driverRoleRules = [
        {
            "apiGroups": [""],
            "resources": ["persistentvolumeclaims"],
            "verbs": ["list", "create", "delete"]
        },
        {
            "apiGroups": [""],
            "resources": ["services"],
            "verbs": ["get", "list", "describe", "create", "delete", "watch"]
        }
    ]

    clientRoleRules = [
        {
            "apiGroups": [""],
            "resources": ["persistentvolumeclaims"],
```

```
            "verbs": ["list", "create", "delete"]
        }
    ]

    patchRole("emr-containers", args.namespace, emrRoleRules, args.no_prompt)
    patchRole("emr-containers-role-spark-driver", args.namespace, driverRoleRules,
args.no_prompt)
    patchRole("emr-containers-role-spark-client", args.namespace, clientRoleRules,
args.no_prompt)
```

3. Execute o script do Python:

```
python3 RBAC_Patch.py -n ${NAMESPACE}
```

4. O comando “kubectl diff” apresentará as diferenças entre as permissões novas e as antigas. Pressione Y para aplicar patches ao perfil.
5. Verifique os três perfis com permissões adicionais, como o apresentado:

```
kubectl describe role -n ${NAMESPACE}
```

6. Execute o script do Python:

```
python3 RBAC_Patch.py -n ${NAMESPACE}
```

7. Após executar o comando, o comando “kubectl diff” apresentará as diferenças entre as permissões novas e as antigas. Pressione Y para aplicar patches ao perfil.
8. Verifique os três perfis com permissões adicionais:

```
kubectl describe role -n ${NAMESPACE}
```

9. Envie o trabalho novamente.

Aplicação manual de patches

Se a permissão requerida pela sua aplicação se aplicar a algo diferente das regras de PVC, você poderá adicionar manualmente as permissões do Kubernetes ao cluster virtual do Amazon EMR, conforme necessário.

> **Note**
>
> O perfil emr-containers é um perfil primário. Isso significa que ele deve fornecer todas as permissões necessárias antes que você possa alterar os perfis subjacentes do driver ou dos clientes.

1. Faça o download das permissões atuais em arquivos YAML ao executar os comandos abaixo:

```
kubectl get role -n ${NAMESPACE} emr-containers -o yaml >> emr-containers-role-patch.yaml
kubectl get role -n ${NAMESPACE} emr-containers-role-spark-driver -o yaml >> driver-role-patch.yaml
kubectl get role -n ${NAMESPACE} emr-containers-role-spark-client -o yaml >> client-role-patch.yaml
```

2. Com base na permissão requerida pela aplicação, edite cada arquivo e adicione regras adicionais, como as seguintes:

   - emr-containers-role-patch.yaml

```
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - list
  - create
  - delete
```

   - driver-role-patch.yaml

```
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - list
  - create
  - delete
- apiGroups:
  - ""
```

```
  resources:
  - services
  verbs:
  - get
  - list
  - describe
  - create
  - delete
  - watch
```

- client-role-patch.yaml

```
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - list
  - create
  - delete
```

3. Remova os atributos a seguir em conjunto com os valores deles. Isso é necessário para aplicar a atualização.
   - creationTimestamp
   - resourceVersion
   - uid
4. Por fim, execute a aplicação de patches:

```
kubectl apply -f emr-containers-role-patch.yaml
kubectl apply -f driver-role-patch.yaml
kubectl apply -f client-role-patch.yaml
```

Solução de problemas de escalabilidade automática vertical do Amazon EMR no EKS

Consulte as seções a seguir se você encontrar problemas ao configurar o operador de escalabilidade automática vertical do Amazon EMR no EKS em um cluster do Amazon EKS com o Operator

Lifecycle Manager. Para obter mais informações, incluindo as etapas para concluir a instalação, consulte Uso da escalabilidade automática vertical com trabalhos do Spark no Amazon EMR .

Erro 403 Forbidden

Se você seguiu as etapas em Instalação do Operator Lifecycle Manager (OLM) no cluster do Amazon EKS, executou o comando `olm status` e ele retornou um erro `403 Forbidden`, como o mostrado abaixo, pode ser que você não tenha obtido os tokens de autenticação para o repositório do Amazon ECR para o operador.

Para resolver esse problema, repita a etapa descrita em Instalação do operador de escalabilidade automática vertical do Amazon EMR no EKS para obter os tokens. Em seguida, tente instalar novamente.

```
Error: FATA[0002] Failed to run bundle: pull bundle image: error pulling image IMAGE.
error resolving name : unexpected status code [manifests latest]: 403 Forbidden
```

Namespace do Kubernetes não encontrado

Ao configurar o operador de escalabilidade automática vertical do Amazon EMR no EKS em um cluster do Amazon EKS, você poderá receber um erro `namespaces not found`, como o mostrado aqui:

```
FATA[0020] Failed to run bundle: create catalog: error creating catalog source:
 namespaces "NAME" not found.
```

Se o namespace especificado não existir, o OLM não instalará o operador de escalabilidade automática vertical. Para resolver esse problema, use o comando apresentado a seguir para criar o namespace. Em seguida, tente instalar novamente.

```
kubectl create namespace NAME
```

Erro ao salvar as credenciais do Docker

Para configurar a escalabilidade automática vertical, você deve autenticar e buscar as imagens do Docker relacionadas à escalabilidade automática vertical do Amazon EMR no EKS. Ao fazer isso, você poderá receber um erro como o seguinte se o Docker não estiver em execução:

```
aws ecr get-login-password \
```

```
 --region $REGION | docker login \
 --username AWS \
 --password-stdin $ACCOUNT_ID.dkr.ecr.$REGION.amazonaws.com

Error saving credentials: error storing credentials - err: exit status 1
out: 'Post "http://ipc/registry/credstore-updated": dial unix backend.sock: connect: no
 such file or directory'
```

Para resolver esse problema, confirme se o Docker está em execução ou abra o Docker Desktop. Em seguida, tente salvar as credenciais novamente.

Solução de problemas do operador do Spark do Amazon EMR no EKS

Consulte as seções a seguir se você encontrar problemas com o operador do Spark do Amazon EMR no EKS. Para obter mais informações, incluindo as etapas para concluir a instalação, consulte Execução de trabalhos do Spark com o operador do Spark.

Erro na instalação do chart do Helm

Se você seguiu as etapas em Instalação do operador do Spark e ele retornou um erro `INSTALLATION FAILED`, como o mostrado abaixo, ao tentar instalar ou verificar o chart do Helm, pode ser que você não tenha obtido os tokens de autenticação para o repositório do Amazon ECR para o operador.

Para resolver esse problema, repita a descrita etapa em Instalação do operador do Spark para autenticar seu cliente Helm para o registro do Amazon ECR. Em seguida, tente realizar a etapa de instalação novamente.

```
Error: INSTALLATION FAILED: Kubernetes cluster unreachable: the server has asked for
 the client to provide credentials
```

UnsupportedFileSystemException: No FileSystem for scheme "s3"

Você pode encontrar a seguinte exceção no thread “principal”:

```
org.apache.hadoop.fs.UnsupportedFileSystemException: No FileSystem for scheme "s3"
```

Se isso ocorrer, adicione as seguintes exceções à especificação `SparkApplication`:

```
 hadoopConf:
   # EMRFS filesystem
   fs.s3.customAWSCredentialsProvider:
 com.amazonaws.auth.WebIdentityTokenCredentialsProvider
   fs.s3.impl: com.amazon.ws.emr.hadoop.fs.EmrFileSystem
   fs.AbstractFileSystem.s3.impl: org.apache.hadoop.fs.s3.EMRFSDelegate
   fs.s3.buffer.dir: /mnt/s3
   fs.s3.getObject.initialSocketTimeoutMilliseconds: "2000"
   mapreduce.fileoutputcommitter.algorithm.version.emr_internal_use_only.EmrFileSystem:
 "2"
   mapreduce.fileoutputcommitter.cleanup-
failures.ignored.emr_internal_use_only.EmrFileSystem: "true"
 sparkConf:
   # Required for EMR Runtime
   spark.driver.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/hadoop-
aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/share/aws/
emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/security/conf:/
usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-glue-datacatalog-
spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-serde.jar:/usr/share/aws/
sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/hadoop/extrajars/*
   spark.driver.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/lib/
native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/native
   spark.executor.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/hadoop-
aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/share/aws/
emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/security/conf:/
usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-glue-datacatalog-
spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-serde.jar:/usr/share/aws/
sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/hadoop/extrajars/*
   spark.executor.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/lib/
native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/native
```

Cotas e endpoints de serviço do Amazon EMR no EKS

A seguir são apresentados os endpoints de serviço e as cotas de serviço para o Amazon EMR no EKS. Para se conectar a um serviço da AWS de forma programática, use um endpoint. Além dos endpoints padrão da AWS, alguns serviços da AWS oferecem endpoints FIPS em regiões selecionadas. Para mais informações, consulte Endpoints de serviço da AWS. As cotas de serviço, também chamadas de limites, correspondem ao número máximo de recursos ou operações de serviço para sua conta da AWS. Para obter mais informações, consulte as cotas de serviço do AWS.

Service endpoints (Endpoints de serviço)

Região da AWS name	Código	Endpoint	Protocolo
Leste dos EUA (N. da Virgínia)	us-east-1	`emr-containers.us-east-1.amazonaws.com`	HTTPS
Leste dos EUA (Ohio)	us-east-2	`emr-containers.us-east-2.amazonaws.com`	HTTPS
Oeste dos EUA (N. da Califórnia)	us-west-1	`emr-containers.us-west-1.amazonaws.com`	HTTPS
Oeste dos EUA (Oregon)	us-west-2	`emr-containers.us-west-2.amazonaws.com`	HTTPS
Ásia-Pacífico (Tóquio)	ap-northeast-1	`emr-containers.ap-northeast-1.amazonaws.com`	HTTPS
Ásia-Pacífico (Seul)	ap-northeast-2	`emr-containers.ap-northeast-2.amazonaws.com`	HTTPS
Ásia-Pacífico (Mumbai)	ap-south-1	`emr-containers.ap-south-1.amazonaws.com`	HTTPS
Ásia-Pacífico (Singapura)	ap-southeast-1	`emr-containers.ap-southeast-1.amazonaws.com`	HTTPS

Região da AWS name	Código	Endpoint	Protocolo
Ásia-Pacífico (Sydney)	ap-southeast-2	`emr-containers.ap-southeast-2.amazonaws.com`	HTTPS
Ásia-Pacífico (Hong Kong)	ap-east-1	`emr-containers.ap-east-1.amazonaws.com`	HTTPS
Canadá (Central)	ca-central-1	`emr-containers.ca-central-1.amazonaws.com`	HTTPS
Europa (Frankfurt)	eu-central-1	`emr-containers.eu-central-1.amazonaws.com`	HTTPS
Europa (Irlanda)	eu-west-1	`emr-containers.eu-west-1.amazonaws.com`	HTTPS
Europe (London)	eu-west-2	`emr-containers.eu-west-2.amazonaws.com`	HTTPS
Europe (Stockholm)	eu-north-1	`emr-containers.eu-north-1.amazonaws.com`	HTTPS
América do Sul (São Paulo)	sa-east-1	`emr-containers.sa-east-1.amazonaws.com`	HTTPS
Oriente Médio (Bahrein)	me-south-1	`emr-containers.me-south-1.amazonaws.com`	HTTPS
AWS GovCloud (Leste dos EUA)	us-gov-east-1	`emr-containers.us-gov-east-1.amazonaws.com`	HTTPS
AWS GovCloud (Oeste dos EUA)	us-gov-west-1	`emr-containers.us-gov-west-1.amazonaws.com`	HTTPS

Service Quotas

O Amazon EMR no EKS limita as solicitações de API apresentadas a seguir para cada conta da AWS de acordo com a região. Para obter mais informações sobre como o controle de utilização é

aplicado, consulte API Request Throttling na Referência da API do Amazon EC2. É possível solicitar um aumento das cotas de limitação de API para a conta da AWS.

Ação de API	Capacidade máxima do bucket	Taxa de reabastecimento do bucket (por segundo)
CancelJobRun	25	1
CreateManagedEndpoint	25	1
CreateVirtualCluster	25	1
DeleteManagedEndpoint	25	1
DeleteVirtualCluster	25	1
DescribeJobRun	25	1
DescribeVirtualCluster	25	1
ListJobRun	25	1
ListManagedEndpoint	25	1
ListVirtualCluster	25	1
StartJobRun	25	1
At the AWS account level, the bucket maximum capacity and refill rate for the sum of all API actions listed in this table	50	7

Versões do Amazon EMR no EKS

Uma versão do Amazon EMR corresponde a um conjunto de aplicações de código aberto do ecossistema de big data. Cada versão inclui diferentes aplicações, componentes e recursos de big data que você seleciona para que o Amazon EMR no EKS implante e configure ao executar seu trabalho.

A partir das versões 5.32.0 e 6.2.0 do Amazon EMR, é possível implantar o Amazon EMR no EKS. Essa opção de implantação não está disponível em versões de liberação anteriores do Amazon EMR. Você deve especificar uma versão de liberação com suporte ao enviar seu trabalho.

O Amazon EMR no EKS usa o seguinte formato de rótulo de versão: `emr-x.x.x-latest` ou `emr-x.x.x-yyyymmdd` com uma data de versão específica. Por exemplo, o `emr-6.14.0-latest` ou o `emr-6.14.0-20210129`. Ao usar o sufixo `-latest`, você garante que sua versão do Amazon EMR sempre inclua as atualizações de segurança mais recentes.

> **Note**
>
> Para obter uma comparação entre o Amazon EMR no EKS e o Amazon EMR em execução no EC2, consulte as perguntas frequentes sobre o Amazon EMR no site da AWS.

Tópicos

- Versões 6.14.0 do Amazon EMR no EKS
- Versões 6.13.0 do Amazon EMR no EKS
- Versões 6.12.0 do Amazon EMR no EKS
- Versões 6.11.0 do Amazon EMR no EKS
- Versões 6.10.0 do Amazon EMR no EKS
- Versões 6.9.0 do Amazon EMR no EKS
- Versões 6.8.0 do Amazon EMR no EKS
- Versões 6.7.0 do Amazon EMR no EKS
- Versões 6.6.0 do Amazon EMR no EKS
- Versões 6.5.0 do Amazon EMR no EKS
- Versões 6.4.0 do Amazon EMR no EKS
- Versões 6.3.0 do Amazon EMR no EKS

- Versões 6.2.0 do Amazon EMR no EKS
- Versões 5.36.0 do Amazon EMR no EKS
- Versões 5.35.0 do Amazon EMR no EKS
- Versões 5.34.0 do Amazon EMR no EKS
- Versões 5.33.0 do Amazon EMR no EKS
- Versões 5.32.0 do Amazon EMR no EKS

Versões 6.14.0 do Amazon EMR no EKS

Esta página descreve a funcionalidade nova e atualizada do Amazon EMR que é específica para a implantação do Amazon EMR no EKS. Para obter detalhes sobre o Amazon EMR em execução no Amazon EC2 e sobre a versão 6.14.0 do Amazon EMR em geral, consulte Amazon EMR 6.14.0 no Guia de versão do Amazon EMR.

Versões 6.14 do Amazon EMR no EKS

As versões 6.14.0 do Amazon EMR apresentadas a seguir estão disponíveis para o Amazon EMR no EKS. Selecione uma versão específica do emr-6.14.0-XXXX para visualizar mais detalhes, como a etiqueta de imagem do contêiner relacionada.

- emr-6.14.0-latest
- emr-6.14.0-20231005
- emr-6.14.0-spark-rapids-latest
- emr-6.14.0-spark-rapids-20231005
- emr-6.14.0-java11-latest
- emr-6.14.0-java11-20231005
- emr-6.14.0-java17-latest
- emr-6.14.0-java17-20231005
- emr-6.14.0-java17-al2023-latest
- emr-6.14.0-java17-al2023-20231005
- emr-6.14.0-spark-rapids-java17-latest
- emr-6.14.0-spark-rapids-java17-20231005
- emr-6.14.0-spark-rapids-java17-al2023-latest

- emr-6.14.0-spark-rapids-java17-al2023-20231005
- notebook-spark/emr-6.14.0-latest
- notebook-spark/emr-6.14.0-20231005
- notebook-spark/emr-6.14.0-spark-rapids-latest
- notebook-spark/emr-6.14.0-spark-rapids-20231005
- notebook-spark/emr-6.14.0-java11-latest
- notebook-spark/emr-6.14.0-java11-20231005
- notebook-spark/emr-6.14.0-java17-latest
- notebook-spark/emr-6.14.0-java17-20231005
- notebook-spark/emr-6.14.0-java17-al2023-latest
- notebook-spark/emr-6.14.0-java17-al2023-20231005
- notebook-python/emr-6.14.0-latest
- notebook-python/emr-6.14.0-20231005
- notebook-python/emr-6.14.0-spark-rapids-latest
- notebook-python/emr-6.14.0-spark-rapids-20231005
- notebook-python/emr-6.14.0-java11-latest
- notebook-python/emr-6.14.0-java11-20231005
- notebook-python/emr-6.14.0-java17-latest
- notebook-python/emr-6.14.0-java17-20231005
- notebook-python/emr-6.14.0-java17-al2023-latest
- notebook-python/emr-6.14.0-java17-al2023-20231005

Notas de lançamento

Notas da versão 6.14.0 do Amazon EMR no EKS

- Aplicações com suporte: AWS SDK for Java 1.12.543, Apache Spark 3.4.1-amzn-1, Apache Hudi 0.13.1-amzn-2, Apache Iceberg 1.3.0-amzn-0, Delta 2.4.0, Apache Spark RAPIDS 23.06.0-amzn-2, Jupyter Enterprise Gateway 2.7.0
- Componentes com suporte: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg` e `spark-kubernetes`.
- Classificações de configuração com suporte

Para uso com as APIs StartJobRun e CreateManagedEndpoint:

Classificações	Descrições
`core-site`	Altera os valores no arquivo `core-site.xml` do Hadoop.
`emrfs-site`	Alterar as configurações do EMRFS.
`spark-metrics`	Altera os valores no arquivo `metrics.properties` do Spark.
`spark-defaults`	Altera os valores no arquivo `spark-defaults.conf` do Spark.
`spark-env`	Alterar os valores no ambiente do Spark.
`spark-hive-site`	Altera os valores no arquivo `hive-site.xml` do Spark.
`spark-log4j2`	Altera os valores no arquivo `log4j2.properties` do Spark.
`emr-job-submitter`	Configuração para o pod de envio de trabalho.

Para uso específico com as APIs CreateManagedEndpoint:

Classificações	Descrições
`jeg-config`	Altera os valores no arquivo `jupyter_enterprise_gateway_config.py` do Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Altera o valor da imagem do kernel no arquivo de um kernel do Jupyter especificado.

As classificações de configuração permitem que você personalize aplicações. Elas geralmente correspondem a um arquivo XML de configuração da aplicação, como `spark-hive-site.xml`. Para obter mais informações, consulte Configure Applications.

Recursos notáveis

Os atributos apresentados a seguir estão inclusos na versão 6.14 do Amazon EMR no EKS.

- Suporte ao Apache Livy: o Amazon EMR no EKS agora oferece suporte ao Apache Livy com `spark-submit`.

emr-6.14.0-latest

Notas de versão: a versão `emr-6.14.0-latest` direciona para `emr-6.14.0-20231005`, no momento.

Regiões: `emr-6.14.0-latest` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.14.0:latest`.

emr-6.14.0-20231005

Notas de versão: `6.14.0-20231005` foi liberada em 17 de outubro de 2023. Esta é a versão inicial do Amazon EMR 6.14.0.

Regiões: `emr-6.14.0-20231005` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.14.0:20231005`.

Versões 6.13.0 do Amazon EMR no EKS

Esta página descreve a funcionalidade nova e atualizada do Amazon EMR que é específica para a implantação do Amazon EMR no EKS. Para obter detalhes sobre o Amazon EMR em execução no Amazon EC2 e sobre a versão 6.13.0 do Amazon EMR em geral, consulte Amazon EMR 6.13.0 no Guia de versão do Amazon EMR.

Versões 6.13 do Amazon EMR no EKS

As versões 6.13.0 do Amazon EMR apresentadas a seguir estão disponíveis para o Amazon EMR no EKS. Selecione uma versão específica do emr-6.13.0-XXXX para visualizar mais detalhes, como a etiqueta de imagem do contêiner relacionada.

- emr-6.13.0-latest
- emr-6.13.0-20230814
- emr-6.13.0-spark-rapids-latest
- emr-6.13.0-spark-rapids-20230814
- emr-6.13.0-java11-latest
- emr-6.13.0-java11-20230814
- emr-6.13.0-java17-latest
- emr-6.13.0-java17-20230814
- emr-6.13.0-java17-al2023-latest
- emr-6.13.0-java17-al2023-20230814
- emr-6.13.0-spark-rapids-java17-latest
- emr-6.13.0-spark-rapids-java17-20230814
- emr-6.13.0-spark-rapids-java17-al2023-latest
- emr-6.13.0-spark-rapids-java17-al2023-20230814
- notebook-spark/emr-6.13.0-latest
- notebook-spark/emr-6.13.0-20230814
- notebook-spark/emr-6.13.0-spark-rapids-latest
- notebook-spark/emr-6.13.0-spark-rapids-20230814
- notebook-spark/emr-6.13.0-java11-latest
- notebook-spark/emr-6.13.0-java11-20230814
- notebook-spark/emr-6.13.0-java17-latest
- notebook-spark/emr-6.13.0-java17-20230814
- notebook-spark/emr-6.13.0-java17-al2023-latest
- notebook-spark/emr-6.13.0-java17-al2023-20230814
- notebook-python/emr-6.13.0-latest
- notebook-python/emr-6.13.0-20230814

- notebook-python/emr-6.13.0-spark-rapids-latest
- notebook-python/emr-6.13.0-spark-rapids-20230814
- notebook-python/emr-6.13.0-java11-latest
- notebook-python/emr-6.13.0-java11-20230814
- notebook-python/emr-6.13.0-java17-latest
- notebook-python/emr-6.13.0-java17-20230814
- notebook-python/emr-6.13.0-java17-al2023-latest
- notebook-python/emr-6.13.0-java17-al2023-20230814

Notas de lançamento

Notas da versão 6.13.0 do Amazon EMR no EKS

- Aplicações com suporte: AWS SDK for Java 1.12.513, Apache Spark 3.4.1-amzn-0, Apache Hudi 0.13.1-amzn-0, Apache Iceberg 1.3.0-amzn-0, Delta 2.4.0, Apache Spark RAPIDS 23.06.0-amzn-1 e Jupyter Enterprise Gateway 2.6.0.amzn
- Componentes com suporte: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg` e `spark-kubernetes`.
- Classificações de configuração com suporte

  Para uso com as APIs StartJobRun e CreateManagedEndpoint:

Classificações	Descrições
`core-site`	Altera os valores no arquivo `core-site.xml` do Hadoop.
`emrfs-site`	Alterar as configurações do EMRFS.
`spark-metrics`	Altera os valores no arquivo `metrics.properties` do Spark.
`spark-defaults`	Altera os valores no arquivo `spark-defaults.conf` do Spark.
`spark-env`	Alterar os valores no ambiente do Spark.

Classificações	Descrições
`spark-hive-site`	Altera os valores no arquivo `hive-site.xml` do Spark.
`spark-log4j2`	Altera os valores no arquivo `log4j2.properties` do Spark.
`emr-job-submitter`	Configuração para o pod de envio de trabalho.

Para uso específico com as APIs CreateManagedEndpoint:

Classificações	Descrições
`jeg-config`	Altera os valores no arquivo `jupyter_enterprise_gateway_config.py` do Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Altera o valor da imagem do kernel no arquivo de um kernel do Jupyter especificado.

As classificações de configuração permitem que você personalize aplicações. Elas geralmente correspondem a um arquivo XML de configuração da aplicação, como `spark-hive-site.xml`. Para obter mais informações, consulte Configure Applications.

Recursos notáveis

Os recursos apresentados a seguir estão inclusos na versão 6.13 do Amazon EMR no EKS.

- Amazon Linux 2023: com a versão 6.13 do Amazon EMR no EKS e versões superiores, é possível iniciar o Spark com o AL2023 como um sistema operacional em conjunto com o runtime do Java 17. Para fazer isso, use o rótulo de versão `al2023` em seu nome. Por exemplo: `emr-6.13.0-java17-al2023-latest`. Recomendamos que você valide e execute testes de performance antes de mover as workloads de produção para o AL2023 e para o Java 17.
- Amazon EMR no EKS com o Apache Flink (pré-visualização pública): as versões 6.13 do Amazon EMR no EKS e versões superiores oferecem suporte ao Apache Flink, que está disponível em

pré-visualização pública. Com este lançamento, você pode executar sua aplicação baseada no Apache Flink em conjunto com outros tipos de aplicações no mesmo cluster do Amazon EKS. Isso ajuda a melhorar a utilização de recursos e a simplificar o gerenciamento da infraestrutura. Se você já executa estruturas de big data no Amazon EKS, agora é possível permitir que o Amazon EMR automatize o provisionamento e o gerenciamento.

emr-6.13.0-latest

Notas de versão: a versão `emr-6.13.0-latest` direciona para `emr-6.13.0-20230814`, no momento.

Regiões: `emr-6.13.0-latest` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.13.0:latest`.

emr-6.13.0-20230814

Notas de versão: `6.13.0-20230814` foi liberada em 7 de setembro de 2023. Esta é a versão inicial do Amazon EMR 6.13.0.

Regiões: `emr-6.13.0-20230814` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.13.0:20230814`.

Versões 6.12.0 do Amazon EMR no EKS

Esta página descreve a funcionalidade nova e atualizada do Amazon EMR que é específica para a implantação do Amazon EMR no EKS. Para obter detalhes sobre o Amazon EMR em execução no Amazon EC2 e sobre a versão 6.12.0 do Amazon EMR em geral, consulte Amazon EMR 6.12.0 no Guia de versão do Amazon EMR.

Versões 6.12 do Amazon EMR no EKS

As versões 6.12.0 do Amazon EMR apresentadas a seguir estão disponíveis para o Amazon EMR no EKS. Selecione uma versão específica do emr-6.12.0-XXXX para visualizar mais detalhes, como a etiqueta de imagem do contêiner relacionada.

- emr-6.12.0-latest
- emr-6.12.0-20230701
- emr-6.12.0-spark-rapids-latest
- emr-6.12.0-spark-rapids-20230701
- emr-6.12.0-java11-latest
- emr-6.12.0-java11-20230701
- emr-6.12.0-java17-latest
- emr-6.12.0-java17-20230701
- emr-6.12.0-spark-rapids-java17-latest
- emr-6.12.0-spark-rapids-java17-20230701
- notebook-spark/emr-6.12.0-latest
- notebook-spark/emr-6.12.0-20230701
- notebook-spark/emr-6.12.0-spark-rapids-latest
- notebook-spark/emr-6.12.0-spark-rapids-20230701
- notebook-python/emr-6.12.0-latest
- notebook-python/emr-6.12.0-20230701
- notebook-python/emr-6.12.0-spark-rapids-latest
- notebook-python/emr-6.12.0-spark-rapids-20230701

Notas de lançamento

Notas da versão 6.12.0 do Amazon EMR no EKS

- Aplicações com suporte: AWS SDK for Java 1.12.490, Apache Spark 3.4.0-amzn-0, Apache Hudi 0.13.1-amzn-0, Apache Iceberg 1.3.0-amzn-0, Delta 2.4.0, Apache Spark RAPIDS 23.06.0-amzn-0 e Jupyter Enterprise Gateway 2.6.0
- Componentes com suporte: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg` e `spark-kubernetes`.
- Classificações de configuração com suporte

  Para uso com as APIs StartJobRun e CreateManagedEndpoint:

Classificações	Descrições
`core-site`	Altera os valores no arquivo `core-site.xml` do Hadoop.
`emrfs-site`	Alterar as configurações do EMRFS.
`spark-metrics`	Altera os valores no arquivo `metrics.properties` do Spark.
`spark-defaults`	Altera os valores no arquivo `spark-defaults.conf` do Spark.
`spark-env`	Alterar os valores no ambiente do Spark.
`spark-hive-site`	Altera os valores no arquivo `hive-site.xml` do Spark.
`spark-log4j2`	Altera os valores no arquivo `log4j2.properties` do Spark.
`emr-job-submitter`	Configuração para o pod de envio de trabalho.

Para uso específico com as APIs CreateManagedEndpoint:

Classificações	Descrições
`jeg-config`	Altera os valores no arquivo `jupyter_enterprise_gateway_config.py` do Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Altera o valor da imagem do kernel no arquivo de um kernel do Jupyter especificado.

As classificações de configuração permitem que você personalize aplicações. Elas geralmente correspondem a um arquivo XML de configuração da aplicação, como `spark-hive-site.xml`. Para obter mais informações, consulte Configure Applications.

Recursos notáveis

Os recursos apresentados a seguir estão inclusos na versão 6.12 do Amazon EMR no EKS.

- Java 17: com a versão 6.12 do Amazon EMR no EKS e versões superiores, é possível iniciar o Spark com o runtime do Java 17. Para fazer isso, transmita `emr-6.12.0-java17-latest` como uma etiqueta de versão. Recomendamos que você valide e execute testes de performance antes de mover as workloads de produção de versões anteriores da imagem do Java para a imagem do Java 17.

emr-6.12.0-latest

Notas de versão: a versão `emr-6.12.0-latest` direciona para `emr-6.12.0-20230701`, no momento.

Regiões: `emr-6.12.0-latest` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.12.0:latest`.

emr-6.12.0-20230701

Notas de versão: `6.12.0-20230701` foi liberada em 1.º de julho de 2023. Esta é a versão inicial do Amazon EMR 6.12.0.

Regiões: `emr-6.12.0-20230701` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.12.0:20230701`.

Versões 6.11.0 do Amazon EMR no EKS

Esta página descreve a funcionalidade nova e atualizada do Amazon EMR que é específica para a implantação do Amazon EMR no EKS. Para obter detalhes sobre o Amazon EMR em execução no

Amazon EC2 e sobre a versão 6.11.0 do Amazon EMR em geral, consulte Amazon EMR 6.11.0 no Guia de versão do Amazon EMR.

Versões 6.11 do Amazon EMR no EKS

As versões 6.11.0 do Amazon EMR apresentadas a seguir estão disponíveis para o Amazon EMR no EKS. Selecione uma versão específica do emr-6.11.0-XXXX para visualizar mais detalhes, como a etiqueta de imagem do contêiner relacionada.

- emr-6.11.0-latest
- emr-6.11.0-20230509

- emr-6.11.0-spark-rapids-latest
- emr-6.11.0-spark-rapids-20230509
- emr-6.11.0-java11-latest
- emr-6.11.0-java11-20230509
- notebook-spark/emr-6.11.0-latest
- notebook-spark/emr-6.11.0-20230509
- notebook-python/emr-6.11.0-latest
- notebook-python/emr-6.11.0-20230509

Notas de lançamento

Notas da versão 6.11.0 do Amazon EMR no EKS

- Aplicações com suporte: AWS SDK for Java 1.12.446, Apache Spark 3.3.2-amzn-0, Apache Hudi 0.13.0-amzn-0, Apache Iceberg 1.2.0-amzn-0, Delta 2.2.0, Apache Spark RAPIDS 23.02.0-amzn-0 e Jupyter Enterprise Gateway 2.6.0
- Componentes com suporte: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg` e `spark-kubernetes`.
- Classificações de configuração com suporte

  Para uso com as APIs StartJobRun e CreateManagedEndpoint:

Classificações	Descrições
`core-site`	Altera os valores no arquivo `core-site.xml` do Hadoop.
`emrfs-site`	Alterar as configurações do EMRFS.
`spark-metrics`	Altera os valores no arquivo `metrics.properties` do Spark.
`spark-defaults`	Altera os valores no arquivo `spark-defaults.conf` do Spark.
`spark-env`	Alterar os valores no ambiente do Spark.
`spark-hive-site`	Altera os valores no arquivo `hive-site.xml` do Spark.
`spark-log4j`	Altera os valores no arquivo `log4j.properties` do Spark.

Para uso específico com as APIs CreateManagedEndpoint:

Classificações	Descrições
`jeg-config`	Altera os valores no arquivo `jupyter_enterprise_gateway_config.py` do Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Altera o valor da imagem do kernel no arquivo de um kernel do Jupyter especificado.

As classificações de configuração permitem que você personalize aplicações. Elas geralmente correspondem a um arquivo XML de configuração da aplicação, como `spark-hive-site.xml`. Para obter mais informações, consulte Configure Applications.

Recursos notáveis

Os recursos apresentados a seguir estão inclusos na versão 6.11 do Amazon EMR no EKS.

- Imagem base do Amazon EMR no EKS na galeria pública do Amazon ECR: se você usar a funcionalidade imagem personalizada, nossa imagem base fornecerá os arquivos em JARs, as configurações e as bibliotecas essenciais para a interação com o Amazon EMR no EKS. É possível encontrar a imagem base na Galeria pública do Amazon ECR.
- Alternância de log do contêiner do Spark: a verão 6.11 do Amazon EMR no EKS oferece suporte para a alternância de log do contêiner do Spark. É possível habilitar a funcionalidade com `containerLogRotationConfiguration` na operação `MonitoringConfiguration` da API `StartJobRun`. Você pode configurar `rotationSize` e `maxFilestoKeep` para especificar o número e o tamanho dos arquivos de log que você deseja que o Amazon EMR no EKS mantenha nos pods do driver e do executor do Spark. Para obter mais informações, consulte Uso da alternância de log do contêiner do Spark.
- Suporte ao Volcano no operador do Spark e no spark-submit: a versão 6.11 do Amazon EMR no EKS oferece suporte à execução de trabalhos do Spark com o Volcano como programador personalizado do Kubernetes no operador do Spark e no spark-submit. É possível usar recursos como o agendamento coletivo, o gerenciamento de filas, a preempção e a programação de compartilhamento equitativo para obter alto throughput de programação e capacidade otimizada. Para obter mais informações, consulte Uso do Volcano como um programador personalizado para Apache Spark no Amazon EMR no EKS.

emr-6.11.0-latest

Notas de versão: a versão `emr-6.11.0-latest` direciona para `emr-20230509`, no momento.

Regiões: `emr-6.11.0-latest` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.11.0:latest`.

emr-6.11.0-20230509

Notas de versão: `6.11.0-20230509` foi liberada em 9 de maio de 2023. Esta é a versão inicial do Amazon EMR 6.11.0.

Regiões: `emr-6.11.0-20230509` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.11.0:20230509`.

Versões 6.10.0 do Amazon EMR no EKS

As versões 6.10.0 do Amazon EMR apresentadas a seguir estão disponíveis para o Amazon EMR no EKS. Selecione uma versão específica do emr-6.10.0-XXXX para visualizar mais detalhes, como a etiqueta de imagem do contêiner relacionada.

- emr-6.10.0-latest
- emr-6.10.0-20230624
- emr-6.10.0-20230421
- emr-6.10.0-20230403
- emr-6.10.0-20230220
- emr-6.10.0-spark-rapids-latest
- emr-6.10.0-spark-rapids-20230624
- emr-6.10.0-spark-rapids-20230220
- emr-6.10.0-java11-latest
- emr-6.10.0-java11-20230624
- emr-6.10.0-java11-20230220
- notebook-spark/emr-6.10.0-latest
- notebook-spark/emr-6.10.0-20230624
- notebook-spark/emr-6.10.0-20230220
- notebook-python/emr-6.10.0-latest
- notebook-python/emr-6.10.0-20230624
- notebook-python/emr-6.10.0-20230220

Notas da versão 6.10.0 do Amazon EMR

- Aplicações com suporte: AWS SDK for Java 1.12.397, Spark 3.3.1-amzn-0, Hudi 0.12.2-amzn-0, Iceberg 1.1.0-amzn-0 e Delta 2.2.0.

- Componentes com suporte: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg` e `spark-kubernetes`.
- Classificações de configuração com suporte:

  Para uso com as APIs StartJobRun e CreateManagedEndpoint:

Classificações	Descrições
`core-site`	Altera os valores no arquivo `core-site.xml` do Hadoop.
`emrfs-site`	Alterar as configurações do EMRFS.
`spark-metrics`	Altera os valores no arquivo `metrics.properties` do Spark.
`spark-defaults`	Altera os valores no arquivo `spark-defaults.conf` do Spark.
`spark-env`	Alterar os valores no ambiente do Spark.
`spark-hive-site`	Altera os valores no arquivo `hive-site.xml` do Spark.
`spark-log4j`	Altera os valores no arquivo `log4j.properties` do Spark.

  Para uso específico com as APIs CreateManagedEndpoint:

Classificações	Descrições
`jeg-config`	Altera os valores no arquivo `jupyter_enterprise_gateway_config.py` do Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Altera o valor da imagem do kernel no arquivo de um kernel do Jupyter especificado.

As classificações de configuração permitem que você personalize aplicações. Elas geralmente correspondem a um arquivo XML de configuração da aplicação, como `spark-hive-site.xml`. Para obter mais informações, consulte Configure Applications.

Recursos notáveis

- Operador do Spark: com a versão 6.10.0 do Amazon EMR no EKS e versões posteriores, é possível usar o operador do Kubernetes para Apache Spark ou o operador do Spark para implantar e gerenciar aplicações do Spark com o runtime de versão do Amazon EMR em seus próprios clusters do Amazon EKS. Para obter mais informações, consulte Execução de trabalhos do Spark com o operador do Spark.
- Java 11: com a versão 6.10 do Amazon EMR no EKS e versões superiores, é possível iniciar o Spark com o runtime do Java 11. Para fazer isso, transmita `emr-6.10.0-java11-latest` como uma etiqueta de versão. Recomendamos que você valide e execute testes de performance antes de mover as workloads de produção da imagem do Java 8 para a imagem do Java 11.
- Para a integração do Amazon Redshift para Apache Spark, a versão 6.10.0 do Amazon EMR no EKS remove a dependência de `minimal-json.jar` e adiciona automaticamente os arquivos em JARs relacionados ao `spark-redshift` obrigatórios ao caminho de classe do executor para o Spark: `spark-redshift.jar`, `spark-avro.jar` e `RedshiftJDBC.jar`.

Alterações

- O confirmador otimizado para EMRFS S3 está habilitado por padrão para Parquet, ORC e formatos baseados em texto (incluindo CSV e JSON).

emr-6.10.0-latest

Notas de versão: a versão `emr-6.10.0-latest` direciona para `emr-6.10.0-20230624`, no momento.

Regiões: `emr-6.10.0-latest` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.10.0:latest`.

emr-6.10.0-20230624

Notas de versão: `6.10.0-20230624` foi liberada em 7 de julho de 2023. Em comparação com a versão anterior, esta versão foi atualizada com pacotes do Amazon Linux atualizados recentemente e correções críticas.

Regiões: `emr-6.10.0-20230624` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.10.0:20230624`.

emr-6.10.0-20230421

Notas de versão: `6.10.0-20230421` foi liberada em 28 de abril de 2023. Em comparação com a versão anterior, esta versão foi atualizada com pacotes do Amazon Linux atualizados recentemente e correções críticas.

Regiões: `emr-6.10.0-20230421` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.10.0:20230421`.

emr-6.10.0-20230403

Notas de versão: `6.10.0-20230403` foi liberada em 12 de abril de 2023. Em comparação com a versão anterior, esta versão foi atualizada com pacotes do Amazon Linux atualizados recentemente e correções críticas.

Regiões: `emr-6.10.0-20230403` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.10.0:20230403`.

emr-6.10.0-20230220

Notas de versão: `emr-6.10.0-20230220` foi liberada em 20 de fevereiro de 2023. Esta é a versão inicial do Amazon EMR 6.10.0.

Regiões: `emr-6.10.0-20230220` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.10.0:20230220`.

Versões 6.9.0 do Amazon EMR no EKS

As versões 6.9.0 do Amazon EMR apresentadas a seguir estão disponíveis para o Amazon EMR no EKS. Selecione uma versão específica do emr-6.9.0-XXXX para visualizar mais detalhes, como a etiqueta de imagem do contêiner relacionada.

- emr-6.9.0-latest
- emr-6.9.0-20221108
- emr-6.9.0-20221108
- emr-6.9.0-spark-rapids-latest
- emr-6.9.0-spark-rapids-20230624
- emr-6.9.0-spark-rapids-20221108
- notebook-spark/emr-6.9.0-latest
- notebook-spark/emr-6.9.0-20230624
- notebook-spark/emr-6.9.0-20221108
- notebook-python/emr-6.9.0-latest
- notebook-python/emr-6.9.0-20230624
- notebook-python/emr-6.9.0-20221108

Notas da versão 6.9.0 do Amazon EMR

- Aplicações com suporte: AWS SDK for Java 1.12.331, Spark 3.3.0-amzn-1, Hudi 0.12.1-amzn-0, Iceberg 0.14.1-amzn-0 e Delta 2.1.0.
- Componentes com suporte: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg` e `spark-kubernetes`.
- Classificações de configuração com suporte:

  Para uso com as APIs StartJobRun e CreateManagedEndpoint:

Classificações	Descrições
`core-site`	Altera os valores no arquivo core-site.xml do Hadoop.

Classificações	Descrições
`emrfs-site`	Alterar as configurações do EMRFS.
`spark-metrics`	Alterar os valores no arquivo metrics.properties do Spark.
`spark-defaults`	Alterar os valores no arquivo spark-defaults.conf do Spark.
`spark-env`	Alterar os valores no ambiente do Spark.
`spark-hive-site`	Altera os valores no arquivo hive-site.xml do Spark.
`spark-log4j`	Alterar os valores no arquivo log4j.properties do Spark.

Para uso específico com as APIs CreateManagedEndpoint:

Classificações	Descrições
`jeg-config`	Altera os valores no arquivo `jupyter_enterprise_gateway_config.py` do Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Altera o valor da imagem do kernel no arquivo de um kernel do Jupyter especificado.

As classificações de configuração permitem que você personalize aplicações. Elas geralmente correspondem a um arquivo XML de configuração da aplicação, como `spark-hive-site.xml`. Para obter mais informações, consulte Configure Applications.

Recursos notáveis

- Acelerador RAPIDS da Nvidia para Apache Spark: Amazon EMR no EKS para acelerar o Spark usando tipos de instância de unidade de processamento gráfico (GPU) do EC2. Para usar a

imagem do Spark com o acelerador RAPIDS, especifique o rótulo de versão como emr-6.9.0-spark-rapids-latest. Acesse a página de documentação para saber mais.

- Conector Spark-Redshift: a integração do Amazon Redshift para Apache Spark está inclusa nas versões 6.9.0 e posteriores do Amazon EMR. Anteriormente uma ferramenta de código aberto, a integração nativa é um conector do Spark que você pode usar para criar aplicações do Apache Spark que realizam a leitura e a gravação de dados no Amazon Redshift e no Amazon Redshift sem servidor. Para obter mais informações, consulte Uso da integração do Amazon Redshift para Apache Spark no Amazon EMR no EKS.
- Delta Lake: o Delta Lake é um formato de armazenamento de código aberto que possibilita o desenvolvimento de data lakes com consistência transacional, definição consistente de conjuntos de dados, alterações de evolução de esquema e suporte a mutações de dados. Acesse Uso do Delta Lake para saber mais.
- Modificação de parâmetros PySpark: os endpoints interativos passaram a oferecer suporte para a modificação de parâmetros Spark associados a sessões PySpark em cadernos Jupyter no EMR Studio. Acesse Modificação de parâmetros de sessões do PySpark para saber mais.

Problemas resolvidos

- Ao usar o conector DynamoDB com o Spark nas versões 6.6.0, 6.7.0 e 6.8.0 do Amazon EMR, todas as leituras da tabela retornam um resultado vazio, mesmo que a divisão de entrada faça referência a dados que não estão vazios. A versão 6.9.0 do Amazon EMR corrige esse problema.
- A versão 6.8.0 do Amazon EMR no EKS preenche incorretamente o hash de compilação nos metadados dos arquivos em Parquet gerados usando o Apache Spark. Esse problema pode fazer com que as ferramentas que analisam a string de versão de metadados dos arquivos em Parquet gerados pela versão 6.8.0 do Amazon EMR no EKS apresentem falhas.

Problema conhecido

- Se você usar a integração do Amazon Redshift para Apache Spark e tiver um time, timetz, timestamp ou timestamptz com precisão de microssegundos no formato Parquet, o conector arredondará os valores de tempo para o valor de milissegundo mais próximo. Como solução alternativa, use o parâmetro `unload_s3_format` do formato de descarregamento de texto.

emr-6.9.0-latest

Notas de versão: a versão `emr-6.9.0-latest` direciona para `emr-6.9.0-20230624`, no momento.

Regiões: `emr-6.9.0-latest` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.9.0:latest`.

emr-6.9.0-20230624

Notas de versão: `emr-6.9.0-20230624` foi liberada em 7 de julho de 2023. Esta é a versão inicial do Amazon EMR 6.9.0.

Regiões: `emr-6.9.0-20230624` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.9.0:20230624`.

emr-6.9.0-20221108

Notas de versão: `emr-6.9.0-20221108` foi liberada em 8 de dezembro de 2022. Esta é a versão inicial do Amazon EMR 6.9.0.

Regiões: `emr-6.9.0-20221108` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.9.0:20221108`.

Versões 6.8.0 do Amazon EMR no EKS

As versões 6.8.0 do Amazon EMR apresentadas a seguir estão disponíveis para o Amazon EMR no EKS. Selecione uma versão específica do emr-6.8.0-XXXX para visualizar mais detalhes, como a etiqueta de imagem do contêiner relacionada.

- emr-6.8.0-latest
- emr-6.8.0-20230624

- emr-6.8.0-20221219
- emr-6.8.0-20220802

Notas da versão 6.8.0 do Amazon EMR

- Aplicações com suporte: AWS SDK for Java 1.12.170, Spark 3.3.0-amzn-0, Hudi 0.11.1-amzn-0 e Iceberg 0.14.0-amzn-0.
- Componentes com suporte: `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg` e `spark-kubernetes`.
- Classificações de configuração com suporte:

Classificações	Descrições
`core-site`	Altera os valores no arquivo core-site.xml do Hadoop.
`emrfs-site`	Alterar as configurações do EMRFS.
`spark-metrics`	Alterar os valores no arquivo metrics.properties do Spark.
`spark-defaults`	Alterar os valores no arquivo spark-defaults.conf do Spark.
`spark-env`	Alterar os valores no ambiente do Spark.
`spark-hive-site`	Altera os valores no arquivo hive-site.xml do Spark.
`spark-log4j`	Alterar os valores no arquivo log4j.properties do Spark.

As classificações de configuração permitem que você personalize aplicações. Elas geralmente correspondem a um arquivo XML de configuração da aplicação, como `spark-hive-site.xml`. Para obter mais informações, consulte Configure Applications.

Recursos notáveis

- Spark3.3.0: a versão 6.8 Amazon EMR no EKS inclui o Spark 3.3.0, que oferece suporte ao uso de rótulos seletores de nós separados para pods de executores e de drivers do Spark. Esses novos rótulos possibilitam que você defina os tipos de nós para os pods de drivers e de executores separadamente na API StartJobRun, sem a necessidade de usar modelos de pod.
  - Propriedade do seletor do nó do driver: spark.kubernetes.driver.node.selector.[labelKey]
  - Propriedade do seletor do nó do executor: spark.kubernetes.executor.node.selector.[labelKey]
- Mensagem aprimorada de falha de trabalho: esta versão apresenta as configurações `spark.stage.extraDetailsOnFetchFailures.enabled` e `spark.stage.extraDetailsOnFetchFailures.maxFailuresToInclude` para rastrear falhas de tarefas devido ao código do usuário. Esses detalhes serão usados para aprimorar a mensagem de falha exibida no log do driver quando um estágio é interrompido devido a uma falha na busca aleatória.

Nome da propriedade	Valor padrão	Significado	Desde a versão
`spark.stage.extraDetailsOnFetchFailures.enabled`	false	Se definida como `true`, esta propriedade será usada para aprimorar a mensagem de falha do trabalho exibida no log do driver quando um estágio é interrompido devido a uma falha de busca aleatória. Por padrão, as últimas cinco falhas de tarefas causadas pelo código do usuário são rastreadas e a mensagem de erro de falha é anexada aos logs do driver.	emr-6.8

Nome da propriedade	Valor padrão	Significado	Desde a versão
		Para aumentar o número de falhas de tarefas com exceções de usuário a serem rastreadas, consulte a configuração `spark.stage.extraDetailsOnFetchFailures.maxFailuresToInclude` .	

Nome da propriedade	Valor padrão	Significado	Desde a versão
`spark.stage.extraDetailsOnFetchFailures.maxFailuresToInclude`	5	Número de falhas de tarefas a serem rastreadas por estágio e por tentativa. Esta propriedade é usada para aprimorar a mensagem de falha do trabalho com exceções de usuário exibidas no log do driver quando um estágio é interrompido devido a uma falha de busca aleatória. Esta propriedade funciona somente se a configuração spark.stage.extraDetailsOnFetchFailures.enabled estiver definida como verdadeira.	emr-6.8

Para obter mais informações, consulte a documentação de configuração do Apache Spark.

Problema conhecido

- A versão 6.8.0 do Amazon EMR no EKS preenche incorretamente o hash de compilação nos metadados dos arquivos em Parquet gerados usando o Apache Spark. Esse problema pode fazer com que as ferramentas que analisam a string de versão de metadados dos arquivos em Parquet gerados pela versão 6.8.0 do Amazon EMR no EKS apresentem falhas. Os clientes que analisam

a string de versão dos metadados do Parquet e dependem do hash de compilação devem realizar a alteração para uma versão diferente do Amazon EMR e reescrever o arquivo.

Problema resolvido

- Funcionalidade de interrupção do kernel para kernels do PySpark: as workloads interativas em andamento que são acionadas pela execução de células em um caderno podem ser interrompidas usando a funcionalidade `Interrupt Kernel`. Uma correção foi introduzida para que esta funcionalidade funcione para kernels do PySpark. Isso também está disponível em código aberto em Changes for handling interrupts for PySpark Kubernetes Kernel #1115.

emr-6.8.0-latest

Notas de versão: a versão `emr-6.8.0-latest` direciona para `emr-6.8.0-20230624`, no momento.

Regiões: `emr-6.8.0-latest` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.8.0:latest`.

emr-6.8.0-20230624

Notas de versão: `emr-6.8.0-20230624` foi liberada em 7 de julho de 2023. Em comparação com a versão anterior, esta versão foi atualizada com os pacotes do Amazon Linux atualizados recentemente e com as correções críticas.

Regiões: `emr-6.8.0-20230624` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.8.0:20230624`.

emr-6.8.0-20221219

Notas de versão: `emr-6.8.0-20221219` foi liberada em 19 de janeiro de 2023. Em comparação com a versão anterior, esta versão foi atualizada com os pacotes do Amazon Linux atualizados recentemente e com as correções críticas.

Regiões: `emr-6.8.0-20221219` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.8.0:20221219`.

emr-6.8.0-20220802

Notas de versão: `emr-6.8.0-20220802` foi liberada em 27 de setembro de 2022. Esta é a versão inicial do Amazon EMR 6.8.0.

Regiões: `emr-6.8.0-20220802` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.8.0:20220802`.

Versões 6.7.0 do Amazon EMR no EKS

As versões 6.7.0 do Amazon EMR apresentadas a seguir estão disponíveis para o Amazon EMR no EKS. Selecione uma versão específica do emr-6.7.0-XXXX para visualizar mais detalhes, como a etiqueta de imagem do contêiner relacionada.

- emr-6.7.0-latest
- emr-6.7.0-20230624
- emr-6.7.0-20221219
- emr-6.7.0-20220630

Notas da versão 6.7.0 do Amazon EMR

- Aplicações com suporte: Spark 3.2.1-amzn-0, Jupyter Enterprise Gateway 2.6, Hudi 0.11-amzn-0 e Iceberg 0.13.1.
- Componentes com suporte: `aws-hm-client` (conector do Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb` e `hudi-spark`.
- Com a atualização para o JEG 2.6, o gerenciamento do kernel passou a ser assíncrono, o que significa que o JEG não bloqueia transações quando a inicialização do kernel está em andamento. Isso melhora muito a experiência do usuário ao fornecer o seguinte:
  - capacidade de executar comandos em cadernos que estão em execução no momento quando outras inicializações de kernel estão em andamento;
  - capacidade de iniciar vários kernels simultaneamente sem afetar os kernels que já estão em execução.

- Classificações de configuração com suporte:

Classificações	Descrições
`core-site`	Altera os valores no arquivo `core-site.xml` do Hadoop.
`emrfs-site`	Alterar as configurações do EMRFS.
`spark-metrics`	Altera os valores no arquivo `metrics.properties` do Spark.
`spark-defaults`	Altera os valores no arquivo `spark-defaults.conf` do Spark.
`spark-env`	Alterar os valores no ambiente do Spark.
`spark-hive-site`	Altera os valores no arquivo `hive-site.xml` do Spark.
`spark-log4j`	Altera os valores no arquivo `log4j.properties` do Spark.

  As classificações de configuração permitem que você personalize aplicações. Elas geralmente correspondem a um arquivo XML de configuração da aplicação, como `spark-hive-site.xml`. Para obter mais informações, consulte Configuração de aplicações.

Problemas resolvidos

- A versão 6.7 do Amazon EMR no EKS corrige um problema na versão 6.6 relacionado ao uso da funcionalidade de modelos de pod do Apache Spark com endpoints interativos. O problema estava presente nas versões 6.4, 6.5 e 6.6 do Amazon EMR no EKS. No momento, você pode usar modelos de pod para definir como os pods de driver e de executores do Spark são iniciados ao usar endpoints interativos para executar análises interativas.
- Nas versões anteriores do Amazon EMR no EKS, o Jupyter Enterprise Gateway bloqueava transações quando a inicialização do kernel estava em andamento, e isso impedia a execução de sessões de caderno que estavam em execução no momento. Agora, é possível executar

comandos em cadernos em execução quando outras inicializações de kernel estiverem em andamento. Você também pode iniciar vários kernels simultaneamente sem o risco de perder a conectividade com os kernels que já estão em execução.

emr-6.7.0-latest

Notas de versão: a versão `emr-6.7.0-latest` direciona para `emr-6.7.0-20230624`, no momento.

Regiões: `emr-6.7.0-latest` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.7.0:latest`.

emr-6.7.0-20230624

Notas de versão: `emr-6.7.0-20230624` foi liberada em 7 de julho de 2023. Em comparação com a versão anterior, esta versão foi atualizada com os pacotes do Amazon Linux atualizados recentemente e com as correções críticas.

Regiões: `emr-6.7.0-20230624` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.7.0:20230624`.

emr-6.7.0-20221219

Notas de versão: `emr-6.7.0-20221219` foi liberada em 19 de janeiro de 2023. Em comparação com a versão anterior, esta versão foi atualizada com os pacotes do Amazon Linux atualizados recentemente e com as correções críticas.

Regiões: `emr-6.7.0-20221219` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.7.0:20221219`.

emr-6.7.0-20220630

Notas de versão: `emr-6.7.0-20220630` foi liberada em 12 de julho de 2022. Esta é a versão inicial do Amazon EMR 6.7.0.

Regiões: `emr-6.7.0-20220630` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.7.0:20220630`.

Versões 6.6.0 do Amazon EMR no EKS

As versões 6.6.0 do Amazon EMR apresentadas a seguir estão disponíveis para o Amazon EMR no EKS. Selecione uma versão específica do emr-6.6.0-XXXX para visualizar mais detalhes, como a etiqueta de imagem do contêiner relacionada.

- emr-6.6.0-latest
- emr-6.6.0-20230624
- emr-6.6.0-20221219
- emr-6.6.0-20220411

Notas da versão 6.6.0 do Amazon EMR

- Aplicações com suporte: Spark 3.2.0-amzn-0, Jupyter Enterprise Gateway (endpoints e pré-visualização pública), Hudi 0.10.1-amzn-0 e Iceberg 0.13.1.
- Componentes com suporte: `aws-hm-client` (conector do Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb` e `hudi-spark`.
- Classificações de configuração com suporte:

Classificações	Descrições
`core-site`	Altera os valores no arquivo core-site.xml do Hadoop.
`emrfs-site`	Alterar as configurações do EMRFS.
`spark-metrics`	Alterar os valores no arquivo metrics.properties do Spark.
`spark-defaults`	Alterar os valores no arquivo spark-defaults.conf do Spark.

Classificações	Descrições
`spark-env`	Alterar os valores no ambiente do Spark.
`spark-hive-site`	Altera os valores no arquivo hive-site.xml do Spark.
`spark-log4j`	Alterar os valores no arquivo log4j.properties do Spark.

As classificações de configuração permitem que você personalize aplicações. Elas geralmente correspondem a um arquivo XML de configuração da aplicação, como spark-hive-site.xml. Para obter mais informações, consulte Configuração de aplicações.

Problema conhecido

- A funcionalidade do modelo de pod do Spark com endpoints interativos não funciona nas versões 6.4, 6.5 e 6.6 do Amazon EMR no EKS.

Problema resolvido

- Os logs de endpoint interativos são carregados no CloudWatch e no S3.

emr-6.6.0-latest

Notas de versão: a versão `emr-6.6.0-latest` direciona para `emr-6.6.0-20230624`, no momento.

Regiões: `emr-6.6.0-latest` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.6.0:latest`.

emr-6.6.0-20230624

Notas de versão: `emr-6.6.0-20230624` foi liberada em 27 de janeiro de 2023. Em comparação com a versão anterior, esta versão foi atualizada com os pacotes do Amazon Linux atualizados recentemente e com as correções críticas.

Regiões: `emr-6.6.0-20230624` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.6.0:20230624`.

emr-6.6.0-20221219

Notas de versão: `emr-6.6.0-20221219` foi liberada em 27 de janeiro de 2023. Em comparação com a versão anterior, esta versão foi atualizada com os pacotes do Amazon Linux atualizados recentemente e com as correções críticas.

Regiões: `emr-6.6.0-20221219` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.6.0:20221219`.

emr-6.6.0-20220411

Notas de versão: `emr-6.6.0-20220411` foi liberada em 20 de maio de 2022. Esta é a versão inicial do Amazon EMR 6.6.0.

Regiões: `emr-6.6.0-20220411` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.6.0:20220411`.

Versões 6.5.0 do Amazon EMR no EKS

As versões 6.5.0 do Amazon EMR apresentadas a seguir estão disponíveis para o Amazon EMR no EKS. Selecione uma versão específica do emr-6.5.0-XXXX para visualizar mais detalhes, como a etiqueta de imagem do contêiner relacionada.

- emr-6.5.0-latest
- emr-6.5.0-20221219
- emr-6.5.0-20220802
- emr-6.5.0-20211119

Notas da versão 6.5.0 do Amazon EMR

- Aplicações com suporte: Spark 3.1.2-amzn-1 e Jupyter Enterprise Gateway (endpoints e pré-visualização pública).
- Componentes com suporte: `aws-hm-client` (conector do Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb` e `hudi-spark`.
- Classificações de configuração com suporte:

Classificações	Descrições
`core-site`	Altera os valores no arquivo core-site.xml do Hadoop.
`emrfs-site`	Alterar as configurações do EMRFS.
`spark-metrics`	Alterar os valores no arquivo metrics.properties do Spark.
`spark-defaults`	Alterar os valores no arquivo spark-defaults.conf do Spark.
`spark-env`	Alterar os valores no ambiente do Spark.
`spark-hive-site`	Altera os valores no arquivo hive-site.xml do Spark.
`spark-log4j`	Alterar os valores no arquivo log4j.properties do Spark.

As classificações de configuração permitem que você personalize aplicações. Elas geralmente correspondem a um arquivo XML de configuração da aplicação, como spark-hive-site.xml. Para obter mais informações, consulte Configuração de aplicações.

Problema conhecido

- A funcionalidade do modelo de pod do Spark com endpoints interativos não funciona nas versões 6.4 e 6.5 do Amazon EMR no EKS.

emr-6.5.0-latest

Notas de versão: a versão `emr-6.5.0-latest` direciona para `emr-6.5.0-20221219`, no momento.

Regiões: `emr-6.5.0-latest` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.5.0:latest`.

emr-6.5.0-20221219

Notas de versão: `emr-6.5.0-20221219` foi liberada em 19 de janeiro de 2023. Em comparação com a versão anterior, esta versão foi atualizada com os pacotes do Amazon Linux atualizados recentemente e com as correções críticas.

Regiões: `emr-6.5.0-20221219` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.5.0:20221219`.

emr-6.5.0-20220802

Notas de versão: `emr-6.5.0-20220802` foi liberada em 24 de agosto de 2022. Em comparação com a versão anterior, esta versão foi atualizada com os pacotes do Amazon Linux atualizados recentemente.

Regiões: `emr-6.5.0-20220802` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.5.0:20220802`.

emr-6.5.0-20211119

Notas de versão: `emr-6.5.0-20211119` foi liberada em 20 de janeiro de 2022. Esta é a versão inicial do Amazon EMR 6.5.0.

Regiões: `emr-6.5.0-20211119` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.5.0:20211119`.

Versões 6.4.0 do Amazon EMR no EKS

As versões 6.4.0 do Amazon EMR apresentadas a seguir estão disponíveis para o Amazon EMR no EKS. Selecione uma versão específica do emr-6.4.0-XXXX para visualizar mais detalhes, como a etiqueta de imagem do contêiner relacionada.

- emr-6.4.0-latest
- emr-6.4.0-20221219
- emr-6.4.0-20210830

Notas da versão 6.4.0 do Amazon EMR

- Aplicações com suporte: Spark 3.1.2-amzn-0 e Jupyter Enterprise Gateway (endpoints e pré-visualização pública).
- Componentes com suporte: `aws-hm-client` (conector do Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb` e `hudi-spark`.
- Classificações de configuração com suporte:

Classificações	Descrições
`core-site`	Altera os valores no arquivo core-site.xml do Hadoop.
`emrfs-site`	Alterar as configurações do EMRFS.
`spark-metrics`	Alterar os valores no arquivo metrics.properties do Spark.
`spark-defaults`	Alterar os valores no arquivo spark-defaults.conf do Spark.
`spark-env`	Alterar os valores no ambiente do Spark.
`spark-hive-site`	Altera os valores no arquivo hive-site.xml do Spark.

Classificações	Descrições
`spark-log4j`	Alterar os valores no arquivo log4j.properties do Spark.

As classificações de configuração permitem que você personalize aplicações. Elas geralmente correspondem a um arquivo XML de configuração da aplicação, como spark-hive-site.xml. Para obter mais informações, consulte Configuração de aplicações.

Problema conhecido

- A funcionalidade do modelo de pod do Spark com endpoints interativos não funciona na versão 6.4 do Amazon EMR no EKS.

emr-6.4.0-latest

Notas de versão: a versão `emr-6.4.0-latest` direciona para `emr-6.4.0-20221219`, no momento.

Regiões: `emr-6.4.0-latest` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.4.0:latest`.

emr-6.4.0-20221219

Notas de versão: `emr-6.4.0-20221219` foi liberada em 27 de janeiro de 2023. Em comparação com a versão anterior, esta versão foi atualizada com os pacotes do Amazon Linux adicionados recentemente.

Regiões: `emr-6.4.0-20221219` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.4.0:20221219`.

emr-6.4.0-20210830

Notas de versão: `emr-6.4.0-20210830` foi liberada em 9 de dezembro de 2021. Esta é a versão inicial do Amazon EMR 6.4.0.

Regiões: `emr-6.4.0-20210830` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.4.0:20210830`.

Versões 6.3.0 do Amazon EMR no EKS

As versões 6.3.0 do Amazon EMR apresentadas a seguir estão disponíveis para o Amazon EMR no EKS. Selecione uma versão específica do emr-6.3.0-XXXX para visualizar mais detalhes, como a etiqueta de imagem do contêiner relacionada.

- emr-6.3.0-latest
- emr-6.3.0-20220802
- emr-6.3.0-20211008
- emr-6.3.0-20210802
- emr-6.3.0-20210429

Notas da versão 6.3.0 do Amazon EMR

- Novos recursos: começando com o Amazon EMR 6.3.0 na série de versões 6.x, o Amazon EMR no EKS oferece suporte ao recurso de modelo de pod do Spark. Também é possível ativar o recurso de alternância de log de eventos do Spark para o Amazon EMR no EKS. Para obter mais informações, consulte Uso de modelos de pod e Uso da alternância de log de eventos do Spark.
- Aplicações com suporte: Spark 3.1.1-amzn-0 e Jupyter Enterprise Gateway (endpoints e pré-visualização pública).
- Componentes com suporte: `aws-hm-client` (conector do Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb` e `hudi-spark`.
- Classificações de configuração com suporte:

Classificações	Descrições
`core-site`	Altera os valores no arquivo core-site.xml do Hadoop.
`emrfs-site`	Alterar as configurações do EMRFS.

Classificações	Descrições
`spark-metrics`	Alterar os valores no arquivo metrics.properties do Spark.
`spark-defaults`	Alterar os valores no arquivo spark-defaults.conf do Spark.
`spark-env`	Alterar os valores no ambiente do Spark.
`spark-hive-site`	Altera os valores no arquivo hive-site.xml do Spark.
`spark-log4j`	Alterar os valores no arquivo log4j.properties do Spark.

As classificações de configuração permitem que você personalize aplicações. Elas geralmente correspondem a um arquivo XML de configuração da aplicação, como spark-hive-site.xml. Para obter mais informações, consulte Configuração de aplicações.

emr-6.3.0-latest

Notas de versão: a versão `emr-6.3.0-latest` direciona para `emr-6.3.0-20220802`, no momento.

Regiões: `emr-6.3.0-latest` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.3.0:latest`.

emr-6.3.0-20220802

Notas de versão: `emr-6.3.0-20220802` foi liberada em 27 de setembro de 2022. Em comparação com a versão anterior, esta versão foi atualizada com os pacotes do Amazon Linux atualizados recentemente.

Regiões: `emr-6.3.0-20220802` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.3.0:20220802`.

emr-6.3.0-20211008

Notas de versão: `emr-6.3.0-20211008` foi liberada em 9 de dezembro de 2021. Em comparação com a versão anterior, esta versão contém correções de problemas e atualizações de segurança.

Regiões: `emr-6.3.0-20211008` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.3.0:20211008`.

emr-6.3.0-20210802

Notas de versão: `emr-6.3.0-20210802` foi liberada em 2 de agosto de 2021. Em comparação com a versão anterior, esta versão contém correções de problemas e atualizações de segurança.

Regiões: `emr-6.3.0-20210802` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.3.0:20210802`.

emr-6.3.0-20210429

Notas de versão: `emr-6.3.0-20210429` foi liberada em 29 de abril de 2021. Esta é a versão inicial do Amazon EMR 6.3.0.

Regiões: `emr-6.3.0-20210429` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.3.0:20210429`.

Versões 6.2.0 do Amazon EMR no EKS

As versões 6.2.0 do Amazon EMR apresentadas a seguir estão disponíveis para o Amazon EMR no EKS. Selecione uma versão específica do emr-6.2.0-XXXX para visualizar mais detalhes, como a etiqueta de imagem do contêiner relacionada.

- emr-6.2.0-latest

- emr-6.2.0-20220802
- emr-6.2.0-20211008
- emr-6.2.0-20210802
- emr-6.2.0-20210615
- emr-6.2.0-20210129
- emr-6.2.0-20201218
- emr-6.2.0-20201201

Notas da versão 6.2.0 do Amazon EMR

- Aplicações com suporte: Spark 3.0.1-amzn-0 e Jupyter Enterprise Gateway (endpoints e pré-visualização pública).
- Componentes com suporte: `aws-hm-client` (conector do Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb` e `hudi-spark`.
- Classificações de configuração com suporte:

Classificações	Descrições
`core-site`	Altera os valores no arquivo core-site.xml do Hadoop.
`emrfs-site`	Alterar as configurações do EMRFS.
`spark-metrics`	Alterar os valores no arquivo metrics.properties do Spark.
`spark-defaults`	Alterar os valores no arquivo spark-defaults.conf do Spark.
`spark-env`	Alterar os valores no ambiente do Spark.
`spark-hive-site`	Altera os valores no arquivo hive-site.xml do Spark.
`spark-log4j`	Alterar os valores no arquivo log4j.properties do Spark.

As classificações de configuração permitem que você personalize aplicações. Elas geralmente correspondem a um arquivo XML de configuração da aplicação, como spark-hive-site.xml. Para obter mais informações, consulte Configuração de aplicações.

emr-6.2.0-latest

Notas de versão: a versão `emr-6.2.0-latest` direciona para `emr-6.2.0-20220802`, no momento.

Regiões: `emr-6.2.0-latest` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.2.0:20220802`.

emr-6.2.0-20220802

Notas de versão: `emr-6.2.0-20220802` foi liberada em 27 de setembro de 2022. Em comparação com a versão anterior, esta versão foi atualizada com os pacotes do Amazon Linux atualizados recentemente.

Regiões: `emr-6.2.0-20220802` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-6.2.0:20220802`.

emr-6.2.0-20211008

Notas de versão: `emr-6.2.0-20211008` foi liberada em 9 de dezembro de 2021. Em comparação com a versão anterior, esta versão contém correções de problemas e atualizações de segurança.

Regiões: `emr-6.2.0-20211008` está disponível nas seguintes regiões: Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Oregon), Ásia-Pacífico (Tóquio), Europa (Irlanda) e América do Sul (São Paulo).

Etiqueta da imagem do contêiner: `emr-6.2.0:20211008`.

emr-6.2.0-20210802

Notas de versão: `emr-6.2.0-20210802` foi liberada em 2 de agosto de 2021. Em comparação com a versão anterior, esta versão contém correções de problemas e atualizações de segurança.

Regiões: `emr-6.2.0-20210802` está disponível nas seguintes regiões: Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Oregon), Ásia-Pacífico (Tóquio), Europa (Irlanda) e América do Sul (São Paulo).

Etiqueta da imagem do contêiner: `emr-6.2.0:20210802`.

emr-6.2.0-20210615

Notas de versão: `emr-6.2.0-20210615` foi liberada em 15 de junho de 2021. Em comparação com a versão anterior, esta versão contém correções de problemas e atualizações de segurança.

Regiões: `emr-6.2.0-20210615` está disponível nas seguintes regiões: Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Oregon), Ásia-Pacífico (Tóquio), Europa (Irlanda) e América do Sul (São Paulo).

Etiqueta da imagem do contêiner: `emr-6.2.0:20210615`.

emr-6.2.0-20210129

Notas de versão: `emr-6.2.0-20210129` foi liberada em 29 de janeiro de 2021. Em comparação com `emr-6.2.0-20201218`, esta versão contém correções de problemas e atualizações de segurança.

Regiões: `emr-6.2.0-20210129` está disponível nas seguintes regiões: Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Oregon), Ásia-Pacífico (Tóquio), Europa (Irlanda) e América do Sul (São Paulo).

Etiqueta da imagem do contêiner: `emr-6.2.0-20210129`.

emr-6.2.0-20201218

Notas de versão: `emr-6.2.0-20201218` foi liberada em 18 de dezembro de 2020. Em comparação com `emr-6.2.0-20201201`, esta versão contém correções de problemas e atualizações de segurança.

Regiões: `emr-6.2.0-20201218` está disponível nas seguintes regiões: Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Oregon), Ásia-Pacífico (Tóquio), Europa (Irlanda) e América do Sul (São Paulo).

Etiqueta da imagem do contêiner: `emr-6.2.0-20201218`.

emr-6.2.0-20201201

Notas de versão: `emr-6.2.0-20201201` foi liberada em 1.º de dezembro de 2020. Esta é a versão inicial do Amazon EMR 6.2.0.

Regiões: `emr-6.2.0-20201201` está disponível nas seguintes regiões: Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Oregon), Ásia-Pacífico (Tóquio), Europa (Irlanda) e América do Sul (São Paulo).

Etiqueta da imagem do contêiner: `emr-6.2.0-20201201`.

Versões 5.36.0 do Amazon EMR no EKS

As versões 5.36.0 do Amazon EMR apresentadas a seguir estão disponíveis para o Amazon EMR no EKS. Selecione uma versão específica do emr-5.36.0-XXXX para visualizar mais detalhes, como a etiqueta de imagem do contêiner relacionada.

- emr-5.36.0-latest
- emr-5.36.0-20221219
- emr-5.36.0-20220620
- emr-5.36.0-20220525

Notas da versão 5.36.0 do Amazon EMR

- Os problemas de segurança do log4j2 foram corrigidos.
- Aplicações com suporte: Spark 2.4.8-amzn-2, Jupyter Enterprise Gateway (endpoints e pré-visualização pública; o kernel do Scala não tem suporte), livy-0.7.1 e fluentd-4.0.0.
- Componentes com suporte: aws-hm-client, aws-sagemaker-spark-sdk, emr-ddb, emr-goodies, emr-kinesis e kerberos-server.
- Classificações de configuração com suporte:

Classificações	Descrições
`core-site`	Altera os valores no arquivo core-site.xml do Hadoop.
`emrfs-site`	Alterar as configurações do EMRFS.
`spark-metrics`	Alterar os valores no arquivo metrics.properties do Spark.
`spark-defaults`	Alterar os valores no arquivo spark-defaults.conf do Spark.
`spark-env`	Alterar os valores no ambiente do Spark.
`spark-hive-site`	Altera os valores no arquivo hive-site.xml do Spark.
`spark-log4j`	Alterar os valores no arquivo log4j.properties do Spark.

As classificações de configuração permitem que você personalize aplicações. Elas geralmente correspondem a um arquivo XML de configuração da aplicação, como spark-hive-site.xml. Para obter mais informações, consulte Configuração de aplicações.

emr-5.36.0-latest

Notas de versão: a versão `emr-5.36.0-latest` direciona para `emr-5.36.0-20221219`, no momento.

Regiões: `emr-5.36.0-latest` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-5.36.0:latest`.

emr-5.36.0-20221219

Notas de versão: `emr-5.36.0-20221219` foi liberada em 27 de janeiro de 2023. Em comparação com a versão anterior, esta versão foi atualizada com os pacotes do Amazon Linux atualizados recentemente.

Regiões: `emr-5.36.0-20221219` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-5.36.0:20221219`.

emr-5.36.0-20220620

Notas de versão: `emr-5.36.0-20220620` foi liberada em 27 de julho de 2022. Em comparação com a versão anterior, esta versão foi atualizada com os pacotes do Amazon Linux atualizados recentemente.

Regiões: `emr-5.36.0-20220620` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-5.36.0:20220620`.

emr-5.36.0-20220525

Notas de versão: `emr-5.36.0-20220525` foi liberada em 16 de junho de 2022. Esta é a versão inicial do Amazon EMR 5.36.0.

Regiões: `emr-5.36.0-20220525` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-5.36.0:20220525`.

Versões 5.35.0 do Amazon EMR no EKS

As versões 5.35.0 do Amazon EMR apresentadas a seguir estão disponíveis para o Amazon EMR no EKS. Selecione uma versão específica do emr-5.35.0-XXXX para visualizar mais detalhes, como a etiqueta de imagem do contêiner relacionada.

- emr-5.35.0-latest
- emr-5.35.0-20221219

- emr-5.35.0-20220802
- emr-5.35.0-20220307

Notas da versão 5.35.0 do Amazon EMR

- Os problemas de segurança do log4j2 foram corrigidos.
- Aplicações com suporte: Spark 2.4.8-amzn-1, Hudi 0.9.0-amzn-2 e Jupyter Enterprise Gateway (endpoints e pré-visualização pública; o kernel do Scala não tem suporte).
- Componentes com suporte: aws-hm-client (conector do Glue), aws-sagemaker-spark-sdk, emr-s3-select, emrfs, emr-ddb e hudi-spark.
- Classificações de configuração com suporte:

Classificações	Descrições
`core-site`	Altera os valores no arquivo core-site.xml do Hadoop.
`emrfs-site`	Alterar as configurações do EMRFS.
`spark-metrics`	Alterar os valores no arquivo metrics.properties do Spark.
`spark-defaults`	Alterar os valores no arquivo spark-defaults.conf do Spark.
`spark-env`	Alterar os valores no ambiente do Spark.
`spark-hive-site`	Altera os valores no arquivo hive-site.xml do Spark.
`spark-log4j`	Alterar os valores no arquivo log4j.properties do Spark.

As classificações de configuração permitem que você personalize aplicações. Elas geralmente correspondem a um arquivo XML de configuração da aplicação, como spark-hive-site.xml. Para obter mais informações, consulte Configuração de aplicações.

emr-5.35.0-latest

Notas de versão: a versão `emr-5.35.0-latest` direciona para `emr-5.35.0-20221219`, no momento.

Regiões: `emr-5.35.0-latest` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-5.35.0:latest`.

emr-5.35.0-20221219

Notas de versão: `emr-5.35.0-20221219` foi liberada em 27 de janeiro de 2023. Em comparação com a versão anterior, esta versão foi atualizada com os pacotes do Amazon Linux atualizados recentemente.

Regiões: `emr-5.35.0-20221219` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-5.35.0:20221219`.

emr-5.35.0-20220802

Notas de versão: `emr-5.35.0-20220802` foi liberada em 27 de setembro de 2022. Em comparação com a versão anterior, esta versão foi atualizada com os pacotes do Amazon Linux atualizados recentemente.

Regiões: `emr-5.35.0-20220802` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-5.35.0:20220802`.

emr-5.35.0-20220307

Notas de versão: `emr-5.35.0-20220307` foi liberada em 30 de março de 2022. Em comparação com a versão anterior, esta versão foi atualizada com os pacotes do Amazon Linux atualizados recentemente.

Regiões: `emr-5.35.0-20220307` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-5.35.0:20220307`.

Versões 5.34.0 do Amazon EMR no EKS

As versões 5.34.0 do Amazon EMR apresentadas a seguir estão disponíveis para o Amazon EMR no EKS. Selecione uma versão específica do emr-5.34.0-XXXX para visualizar mais detalhes, como a etiqueta de imagem do contêiner relacionada.

- emr-5.34.0-latest
- emr-5.34.0-20220802

Notas da versão 5.34.0 do Amazon EMR

- Aplicações com suporte: Spark 2.4.8-amzn-0 e Jupyter Enterprise Gateway (endpoints e pré-visualização pública; o kernel do Scala não tem suporte).
- Componentes com suporte: `aws-hm-client` (conector do Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb` e `hudi-spark`.
- Classificações de configuração com suporte:

Classificações	Descrições
`core-site`	Altera os valores no arquivo core-site.xml do Hadoop.
`emrfs-site`	Alterar as configurações do EMRFS.
`spark-metrics`	Alterar os valores no arquivo metrics.properties do Spark.
`spark-defaults`	Alterar os valores no arquivo spark-defaults.conf do Spark.
`spark-env`	Alterar os valores no ambiente do Spark.
`spark-hive-site`	Altera os valores no arquivo hive-site.xml do Spark.
`spark-log4j`	Alterar os valores no arquivo log4j.properties do Spark.

As classificações de configuração permitem que você personalize aplicações. Elas geralmente correspondem a um arquivo XML de configuração da aplicação, como spark-hive-site.xml. Para obter mais informações, consulte Configuração de aplicações.

emr-5.34.0-latest

Notas de versão: a versão `emr-5.34.0-latest` direciona para `emr-5.34.0-20220802`, no momento.

Regiões: `emr-5.34.0-latest` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-5.34.0:latest`.

emr-5.34.0-20220802

Notas de versão: `emr-5.34.0-20220802` foi liberada em 24 de agosto de 2022. Em comparação com a versão anterior, esta versão foi atualizada com os pacotes do Amazon Linux atualizados recentemente.

Regiões: `emr-5.34.0-20220802` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-5.34.0:20220802`.

emr-5.34.0-20211208

Notas de versão: `emr-5.34.0-20211208` foi liberada em 20 de janeiro de 2022. Em comparação com a versão anterior, esta versão foi atualizada com os pacotes do Amazon Linux atualizados recentemente.

Regiões: `emr-5.34.0-20211208` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-5.34.0:20211208`.

Versões 5.33.0 do Amazon EMR no EKS

As versões 5.33.0 do Amazon EMR apresentadas a seguir estão disponíveis para o Amazon EMR no EKS. Selecione uma versão específica do emr-5.33.0-XXXX para visualizar mais detalhes, como a etiqueta de imagem do contêiner relacionada.

- emr-5.33.0-latest
- emr-5.33.0-20221219
- emr-5.33.0-20220802
- emr-5.33.0-20211008
- emr-5.33.0-20210802
- emr-5.33.0-20210615
- emr-5.33.0-20210323

Notas da versão 5.33.0 do Amazon EMR

- Novo recurso: começando com o Amazon EMR 5.33.0 na série de versões 5.x, o Amazon EMR no EKS oferece suporte ao recurso de modelo de pod do Spark. Para obter mais informações, consulte Uso de modelos de pod.
- Aplicações com suporte: Spark 2.4.7-amzn-1 e Jupyter Enterprise Gateway (endpoints e pré-visualização pública; o kernel do Scala não tem suporte).
- Componentes com suporte: `aws-hm-client` (conector do Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb` e `hudi-spark`.
- Classificações de configuração com suporte:

Classificações	Descrições
`core-site`	Altera os valores no arquivo core-site.xml do Hadoop.
`emrfs-site`	Alterar as configurações do EMRFS.
`spark-metrics`	Alterar os valores no arquivo metrics.properties do Spark.

Classificações	Descrições
`spark-defaults`	Alterar os valores no arquivo spark-def aults.conf do Spark.
`spark-env`	Alterar os valores no ambiente do Spark.
`spark-hive-site`	Altera os valores no arquivo hive-site.xml do Spark.
`spark-log4j`	Alterar os valores no arquivo log4j.properties do Spark.

As classificações de configuração permitem que você personalize aplicações. Elas geralmente correspondem a um arquivo XML de configuração da aplicação, como spark-hive-site.xml. Para obter mais informações, consulte Configuração de aplicações.

emr-5.33.0-latest

Notas de versão: a versão `emr-5.33.0-latest` direciona para `emr-5.33.0-20221219`, no momento.

Regiões: `emr-5.33.0-latest` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-5.33.0:latest`.

emr-5.33.0-20221219

Notas de versão: `emr-5.33.0-20221219` foi liberada em 19 de janeiro de 2023. Em comparação com a versão anterior, esta versão foi atualizada com os pacotes do Amazon Linux atualizados recentemente e com as correções críticas.

Regiões: `emr-5.33.0-20221219` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-5.33.0:20221219`.

emr-5.33.0-20220802

Notas de versão: `emr-5.33.0-20220802` foi liberada em 24 de agosto de 2022. Em comparação com a versão anterior, esta versão foi atualizada com os pacotes do Amazon Linux atualizados recentemente.

Regiões: `emr-5.33.0-20220802` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-5.33.0:20220802`.

emr-5.33.0-20211008

Notas de versão: `emr-5.33.0-20211008` foi liberada em 9 de dezembro de 2021. Em comparação com a versão anterior, esta versão contém correções de problemas e atualizações de segurança.

Regiões: `emr-5.33.0-20211008` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-5.33.0:20211008`.

emr-5.33.0-20210802

Notas de versão: `emr-5.33.0-20210802` foi liberada em 2 de agosto de 2021. Em comparação com a versão anterior, esta versão contém correções de problemas e atualizações de segurança.

Regiões: `emr-5.33.0-20210802` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-5.33.0:20210802`.

emr-5.33.0-20210615

Notas de versão: `emr-5.33.0-20210615` foi liberada em 15 de junho de 2021. Em comparação com a versão anterior, esta versão contém correções de problemas e atualizações de segurança.

Regiões: `emr-5.33.0-20210615` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-5.33.0:20210615`.

emr-5.33.0-20210323

Notas de versão: `emr-5.33.0-20210323` foi liberada em 23 de março de 2021. Esta é a versão inicial do Amazon EMR 5.33.0.

Regiões: `emr-5.33.0-20210323` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-5.33.0-20210323`.

Versões 5.32.0 do Amazon EMR no EKS

As versões 5.32.0 do Amazon EMR apresentadas a seguir estão disponíveis para o Amazon EMR no EKS. Selecione uma versão específica do emr-5.32.0-XXXX para visualizar mais detalhes, como a etiqueta de imagem do contêiner relacionada.

- emr-5.32.0-latest
- emr-5.32.0-20220802
- emr-5.32.0-20211008
- emr-5.32.0-20210802
- emr-5.32.0-20210615
- emr-5.32.0-20210129
- emr-5.32.0-20201218
- emr-5.32.0-20201201

Notas da versão 5.32.0 do Amazon EMR

- Aplicações com suporte: Spark 2.4.7-amzn-0 e Jupyter Enterprise Gateway (endpoints e pré-visualização pública; o kernel do Scala não é compatível).
- Componentes com suporte: `aws-hm-client` (conector do Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb` e `hudi-spark`.
- Classificações de configuração com suporte:

Classificações	Descrições
`core-site`	Altera os valores no arquivo core-site.xml do Hadoop.
`emrfs-site`	Alterar as configurações do EMRFS.
`spark-metrics`	Alterar os valores no arquivo metrics.properties do Spark.
`spark-defaults`	Alterar os valores no arquivo spark-defaults.conf do Spark.
`spark-env`	Alterar os valores no ambiente do Spark.
`spark-hive-site`	Altera os valores no arquivo hive-site.xml do Spark.
`spark-log4j`	Alterar os valores no arquivo log4j.properties do Spark.

As classificações de configuração permitem que você personalize aplicações. Elas geralmente correspondem a um arquivo XML de configuração da aplicação, como spark-hive-site.xml. Para obter mais informações, consulte Configuração de aplicações.

emr-5.32.0-latest

Notas de versão: a versão `emr-5.32.0-latest` direciona para `emr-5.32.0-20220802`, no momento.

Regiões: `emr-5.32.0-latest` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-5.32.0:latest`.

emr-5.32.0-20220802

Notas de versão: `emr-5.32.0-20220802` foi liberada em 24 de agosto de 2022. Em comparação com a versão anterior, esta versão foi atualizada com os pacotes do Amazon Linux atualizados recentemente.

Regiões: `emr-5.32.0-20220802` está disponível em todas as regiões com suporte do Amazon EMR no EKS. Para obter mais informações, consulte Endpoints de serviço do Amazon EMR no EKS.

Etiqueta da imagem do contêiner: `emr-5.32.0:20220802`.

emr-5.32.0-20211008

Notas de versão: `emr-5.32.0-20211008` foi liberada em 9 de dezembro de 2021. Em comparação com a versão anterior, esta versão contém correções de problemas e atualizações de segurança.

Regiões: `emr-5.32.0-20211008` está disponível nas seguintes regiões: Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Oregon), Ásia-Pacífico (Tóquio), Europa (Irlanda) e América do Sul (São Paulo).

Etiqueta da imagem do contêiner: `emr-5.32.0:20211008`.

emr-5.32.0-20210802

Notas de versão: `emr-5.32.0-20210802` foi liberada em 2 de agosto de 2021. Em comparação com a versão anterior, esta versão contém correções de problemas e atualizações de segurança.

Regiões: `emr-5.32.0-20210802` está disponível nas seguintes regiões: Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Oregon), Ásia-Pacífico (Tóquio), Europa (Irlanda) e América do Sul (São Paulo).

Etiqueta da imagem do contêiner: `emr-5.32.0:20210802`.

emr-5.32.0-20210615

Notas de versão: `emr-5.32.0-20210615` foi liberada em 15 de junho de 2021. Em comparação com a versão anterior, esta versão contém correções de problemas e atualizações de segurança.

Regiões: `emr-5.32.0-20210615` está disponível nas seguintes regiões: Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Oregon), Ásia-Pacífico (Tóquio), Europa (Irlanda) e América do Sul (São Paulo).

Etiqueta da imagem do contêiner: `emr-5.32.0:20210615`.

emr-5.32.0-20210129

Notas de versão: `emr-5.32.0-20210129` foi liberada em 29 de janeiro de 2021. Em comparação com `emr-5.32.0-20201218`, esta versão contém correções de problemas e atualizações de segurança.

Regiões: `emr-5.32.0-20210129` está disponível nas seguintes regiões: Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Oregon), Ásia-Pacífico (Tóquio), Europa (Irlanda) e América do Sul (São Paulo).

Etiqueta da imagem do contêiner: `emr-5.32.0-20210129`.

emr-5.32.0-20201218

Notas de versão: `5.32.0-20201218` foi liberada em 18 de dezembro de 2020. Em comparação com `5.32.0-20201201`, esta versão contém correções de problemas e atualizações de segurança.

Regiões: `emr-5.32.0-20201218` está disponível nas seguintes regiões: Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Oregon), Ásia-Pacífico (Tóquio), Europa (Irlanda) e América do Sul (São Paulo).

Etiqueta da imagem do contêiner: `emr-5.32.0-20201218`.

emr-5.32.0-20201201

Notas de versão: `5.32.0-20201201` foi liberada em 1.º de dezembro de 2020. Esta é a versão inicial do Amazon EMR 5.32.0.

Regiões: `5.32.0-20201201` está disponível nas seguintes regiões: Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Oregon), Ásia-Pacífico (Tóquio), Europa (Irlanda) e América do Sul (São Paulo).

Etiqueta da imagem do contêiner: `emr-5.32.0-20201201`.

Histórico do documentos

A tabela a seguir descreve as alterações importantes na documentação desde a última versão do Amazon EMR no EKS. Para obter mais informações sobre as atualizações desta documentação, você pode se tornar assinante de um feed RSS.

Alteração	Descrição	Data
Nova versão	Versões 6.14.0 do Amazon EMR no EKS e documentos de pré-visualização pública para Execução de trabalhos do Flink com o Amazon EMR no EKS.	17 de outubro de 2023
Atualização de conteúdo	Renomeação de “endpoints gerenciados” para endpoints interativos. Os endpoints interativos estão com disponibilidade geral.	29 de setembro de 2023
Nova versão	Versões 6.13.0 do Amazon EMR no EKS e documentos de pré-visualização pública para Execução de trabalhos do Flink com o Amazon EMR no EKS.	12 de setembro de 2023
Nova versão	Versões 6.12.0 do Amazon EMR no EKS	21 de julho de 2023
Novo conteúdo	Adicionado Uso do Volcano como um programador personalizado para Apache Spark no Amazon EMR no EKS	13 de junho de 2023
Novo conteúdo	Adicionado Uso do Volcano como um programador personalizado para Apache Spark no Amazon EMR no EKS	13 de junho de 2023
Novo conteúdo	Adicionado Uso da alternância de log do contêiner do Spark	12 de junho de 2023

Alteração	Descrição	Data
Atualização de conteúdo	Atualização da documentação de imagens personalizadas para a descoberta de informações sobre as imagens base na galeria pública do Amazon ECR.	8 de junho de 2023
Nova versão	Versões 6.11.0 do Amazon EMR no EKS	8 de junho de 2023
Novo conteúdo	Adição de Execução de trabalhos do Spark com o operador do Spark e reorganização das seções de execuções de trabalhos em Execução de trabalhos com o Amazon EMR no EKS.	5 de junho de 2023
Novo conteúdo	Adição de duas seções: Uso da escalabilidade automática vertical com trabalhos do Spark no Amazon EMR e Uso de cadernos Jupyter de hospedagem própria.	4 de maio de 2023
Página para o histórico de documentos	Criação de uma página para o histórico de documentos para o Amazon EMR no EKS.	13 de março de 2023
Página para as políticas gerenciadas	Criação de uma página para as políticas gerenciadas para o Amazon EMR no EKS.	13 de março de 2023