aws

Panduan Pengguna

Amazon Managed Workflows for Apache Airflow

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Managed Workflows for Apache Airflow: Panduan Pengguna

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan di antara pelanggan, atau dengan cara apa pun yang merendahkan atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan hak milik masing-masing pemiliknya, yang mungkin atau mungkin tidak terafiliasi, terkait dengan, atau disponsori oleh Amazon.

Table of Contents

Apa itu Amazon MWAA? 1
Fitur 1
Arsitektur 2
Integrasi 3
Versi yang didukung 4
Apa selanjutnya? 4
Quick start 5
Dalam tutorial ini 5
Prasyarat 6
Langkah satu: Simpan CloudFormation template secara lokal 7
Langkah kedua: Buat tumpukan menggunakan AWS CLI 16
Langkah ketiga: Unggah DAG ke Amazon S3 dan jalankan di Apache Airflow UI 17
Langkah empat: akses log di CloudWatch Log 18
Apa selanjutnya? 18
Memulai 19
Prasyarat 19
Tentang panduan ini 19
Sebelum Anda mulai 20
Wilayah yang tersedia 20
Membuat bucket 22
Sebelum Anda mulai 22
Buat ember 23
Apa selanjutnya? 24
Buat jaringan VPC 25
Prasyarat 25
Sebelum Anda mulai 26
Opsi untuk membuat jaringan VPC Amazon 26
Apa selanjutnya? 38
Buat lingkungan 38
Sebelum Anda mulai 39
Versi Apache Airflow 39
Buat lingkungan 41
Apa selanjutnya? 24
Mengelola akses 46

Mengakses lingkungan Amazon MWAA ... 46
Cara kerjanya ... 47
Akses konsol penuh ... 48
Akses API penuh ... 55
Read-only akses konsol ... 60
Akses Apache Airflow UI ... 60
Akses API Apache Airflow Rest ... 61
Akses CLI Apache Airflow ... 62
Membuat kebijakan JSON ... 63
Contoh kasus penggunaan ... 63
Apa selanjutnya? ... 65
Service-linked peran ... 66
Service-linked izin peran untuk Amazon MWAA ... 66
Membuat peran terkait layanan untuk Amazon MWAA ... 70
Mengedit peran terkait layanan untuk Amazon MWAA ... 70
Menghapus peran terkait layanan untuk Amazon MWAA ... 70
Wilayah yang didukung untuk peran terkait layanan Amazon MWAA ... 71
Pembaruan kebijakan ... 71
Peran eksekusi ... 71
Ikhtisar peran eksekusi ... 72
Buat peran baru ... 74
Mengakses dan memperbarui kebijakan peran eksekusi ... 75
Berikan akses ke bucket Amazon S3 dengan blok akses publik tingkat akun ... 77
Gunakan koneksi Apache Airflow ... 77
Contoh kebijakan ... 77
Apa selanjutnya? ... 84
Cross-service pencegahan wakil bingung ... 84
Mode akses Apache Airflow ... 85
Mode akses Apache Airflow ... 86
Ikhtisar mode akses ... 88
Pengaturan untuk mode akses ... 89
Mengakses titik akhir VPC untuk server web Apache Airflow Anda (akses jaringan pribadi) ... 91
Mengakses Apache Airflow ... 92
Prasyarat ... 92
Akses ... 92
AWS CLI ... 93

Buka Apache Airflow UI 93
Masuk ke Apache Airflow 93
Buat token akses server web 93
Prasyarat 94
Menggunakan AWS CLI 95
Menggunakan skrip bash 95
Menggunakan skrip Python 96
Apa selanjutnya? 97
Menyiapkan domain khusus 97
Konfigurasikan domain kustom 98
Menyiapkan infrastruktur jaringan 98
Token CLI Aliran Udara Apache 103
Prasyarat 104
Menggunakan AWS CLI 104
Menggunakan skrip curl 105
Menggunakan skrip bash 107
Menggunakan skrip Python 108
Apa selanjutnya? 111
Menggunakan Apache Airflow REST API 111
Memberikan akses ke Apache Airflow REST API: `airflow:InvokeRestApi` 114
Memanggil API REST Apache Airflow 115
Membuat token sesi server web dan memanggil Apache Airflow REST API 116
Referensi perintah CLI Apache Airflow 122
Prasyarat 123
Apa yang berubah? 123
Perintah CLI yang didukung 124
Kode sampel 131
Mengelola koneksi 134
Gambaran Umum 134
Paket Apache Airflow 135
Kendala berkas 135
Version-specific paket penyedia 136
Tipe koneksi 145
Contoh koneksi URI string 146
Contoh template koneksi 146
Contoh menggunakan template koneksi HTTP untuk koneksi Jdbc 146

Mengkonfigurasi Secrets Manager 147
Langkah satu: Berikan Amazon MWAA izin untuk mengakses kunci rahasia Secrets Manager 148
Langkah kedua: Buat backend Secrets Manager sebagai opsi konfigurasi Apache Airflow ... 149
Langkah ketiga: Hasilkan string URI koneksi Apache AWS Airflow 151
Langkah empat: Tambahkan variabel di Secrets Manager 153
Langkah lima: Tambahkan koneksi di Secrets Manager 154
Kode sampel 156
Sumber Daya 156
Apa selanjutnya? 156
Mengelola lingkungan 157
Mengkonfigurasi kelas lingkungan 157
Kemampuan lingkungan 157
Penjadwal Apache Airflow 160
Mengkonfigurasi penskalaan otomatis pekerja 160
Cara kerja penskalaan pekerja 161
Menggunakan konsol Amazon MWAA 162
Contoh kasus penggunaan kinerja tinggi 162
Memecahkan masalah tugas yang macet dalam status berjalan 164
Apa selanjutnya? 164
Mengkonfigurasi penskalaan otomatis server web 164
Cara kerja penskalaan server web 165
Menggunakan konsol Amazon MWAA 165
Menggunakan opsi konfigurasi 165
Prasyarat 166
Cara kerjanya 167
Menggunakan opsi konfigurasi untuk memuat plugin 167
Ikhtisar opsi konfigurasi 167
Referensi konfigurasi 169
Konfigurasi yang tidak didukung 173
Contoh dan kode sampel 174
Apa selanjutnya? 175
Perbarui lingkungan 175
Sebelum Anda mulai 176
Strategi penggantian pekerja 176
Perbarui sumber daya lingkungan 177

Perbarui lingkungan 177
Mengubah versi 181
Tingkatkan atau turunkan sumber daya alur kerja Anda 182
Tentukan versi baru 183
Menggunakan skrip startup 184
Konfigurasikan skrip startup 184
Instal runtime Linux 188
Tetapkan variabel lingkungan 189
Bekerja dengan DAGs 194
Ikhtisar ember Amazon S3 194
Menambahkan atau memperbarui DAGs 195
Prasyarat 195
Cara kerjanya 196
Apa yang berubah? 196
Pengujian DAGs menggunakan utilitas Amazon MWAA CLI 197
Mengunggah kode DAG ke Amazon S3 197
Menentukan path ke folder DAGs 198
Mengakses perubahan pada Apache Airflow UI Anda 199
Apa selanjutnya? 199
Menginstal plugin kustom 199
Prasyarat 200
Cara kerjanya 200
Kapan menggunakan plugin 201
Ikhtisar plugin kustom 201
Contoh plugin kustom 202
Membuat file plugins.zip 212
Mengunggah `plugins.zip` ke Amazon S3 213
Menginstal plugin khusus di lingkungan Anda 214
Contoh kasus penggunaan untuk plugins.zip 215
Apa selanjutnya? 215
Menginstal dependensi Python 215
Prasyarat 216
Cara kerjanya 217
Ikhtisar dependensi Python 217
Membuat file requirements.txt 218
Mengunggah `requirements.txt` ke Amazon S3 221

Menginstal dependensi Python di lingkungan Anda 222
Mengakses log untuk `requirements.txt` 223
Apa selanjutnya? 224
Menghapus file di Amazon S3 224
Prasyarat 225
Ikhtisar versi 225
Cara kerjanya 226
Menghapus DAG di Amazon S3 226
Menghapus “current” plugins.zip atau requirements.txt 227
Hapus “tidak saat ini” plugins.zip atau requirements.txt 227
Menghapus file dengan siklus hidup 227
Contoh kebijakan siklus hidup 228
Apa selanjutnya? 228
Jaringan 229
Tentang jaringan 229
Istilah 230
Apa yang didukung 230
Ikhtisar infrastruktur VPC 230
Contoh kasus penggunaan untuk VPC Amazon dan mode akses Apache Airflow 234
Keamanan di VPC Anda 236
Ketentuan 237
Ikhtisar keamanan 237
Daftar kontrol akses jaringan (ACLs) 238
Grup keamanan VPC 238
Kebijakan titik akhir VPC (hanya perutean pribadi) 240
Mengelola akses ke titik akhir VPC 241
Harga 242
Ikhtisar titik akhir VPC 243
Izin untuk menggunakan yang lain AWS layanan 243
Mengakses titik akhir VPC 244
Mengakses titik akhir VPC untuk server web Apache Airflow Anda (akses jaringan pribadi) . 246
Titik akhir layanan VPC di VPC Amazon pribadi 247
Harga 248
Jaringan pribadi dan perutean pribadi 248
(Diperlukan) Titik akhir VPC 249
Melampirkan titik akhir VPC yang diperlukan 250

(Opsional) Aktifkan alamat IP pribadi untuk titik akhir antarmuka VPC Amazon S3 Anda 253
Mengelola titik akhir VPC Amazon Anda sendiri 254
Menciptakan lingkungan di VPC Amazon bersama 255
Tutorial 265
Tutorial: AWS Client VPN 265
Jaringan pribadi 266
Kasus penggunaan 266
Sebelum Anda mulai 267
Tujuan 267
(Opsional) Langkah satu: Identifikasi VPC, aturan CIDR, dan keamanan VPC Anda 267
Langkah kedua: Buat server dan sertifikat klien 269
Langkah ketiga: Simpan CloudFormation Template secara lokal 270
Langkah keempat: Buat Client VPN CloudFormation tumpukan 271
Langkah lima: Kaitkan subnet ke Client VPN Anda 272
Langkah enam: Tambahkan aturan masuknya otorisasi ke Client VPN Anda 272
Langkah tujuh: Unduh file konfigurasi titik akhir Client VPN 273
Langkah delapan: Connect ke AWS Client VPN 275
Apa selanjutnya? 275
Tutorial: Linux Bastion Host 275
Jaringan pribadi 276
Kasus penggunaan 277
Sebelum Anda mulai 277
Tujuan 277
Langkah satu: Buat instance bastion 278
Langkah kedua: Buat terowongan ssh 279
Langkah ketiga: Konfigurasikan grup keamanan bastion sebagai aturan masuk 280
Langkah empat: Salin URL Apache Airflow 281
Langkah lima: Konfigurasikan pengaturan proxy 281
Langkah enam: Buka Apache Airflow UI 284
Apa selanjutnya? 284
Tutorial: Membatasi pengguna untuk subset DAGs 284
Prasyarat 285
Langkah pertama: Berikan akses server web Amazon MWAA ke kepala IAM Anda dengan peran default Apache Airflow. `Public` 286
Langkah kedua: Buat peran kustom Apache Airflow baru 287
Langkah ketiga: Tetapkan peran yang Anda buat untuk pengguna Amazon MWAA Anda 287

Langkah selanjutnya ... 289
Sumber daya terkait ... 289
Tutorial: Mengotomatiskan mengelola titik akhir lingkungan Anda sendiri ... 289
Prasyarat ... 290
Buat Amazon VPC ... 290
Buat fungsi Lambda ... 291
Buat EventBridge aturan ... 292
Buat lingkungan ... 292
Contoh kode ... 294
Impor variabel DAG ... 295
Versi ... 295
Prasyarat ... 295
Izin ... 295
Dependensi ... 295
Contoh kode ... 295
Apa selanjutnya? ... 297
Menggunakan `SSHOperator` ... 297
Versi ... 298
Prasyarat ... 298
Izin ... 298
Persyaratan ... 298
Salin kunci rahasia Anda ke Amazon S3 ... 299
Buat koneksi Apache Airflow baru ... 299
Contoh kode ... 300
Koneksi Apache Airflow Snowflake di Secrets Manager ... 302
Versi ... 302
Prasyarat ... 302
Izin ... 302
Persyaratan ... 303
Contoh kode ... 303
Apa selanjutnya? ... 304
Menggunakan DAG untuk menulis metrik khusus ... 304
Versi ... 304
Prasyarat ... 305
Izin ... 305
Dependensi ... 305

Contoh kode 305
Pembersihan basis data Aurora PostgreSQL 308
Versi 308
Prasyarat 308
Dependensi 309
Contoh kode 309
Mengekspor metadata lingkungan ke Amazon S3 311
Versi 312
Prasyarat 312
Izin 312
Persyaratan 312
Contoh kode 313
Menggunakan variabel Apache Airflow di Secrets Manager 315
Versi 316
Prasyarat 316
Izin 316
Persyaratan 316
Contoh kode 316
Apa selanjutnya? 317
Menggunakan koneksi Apache Airflow di Secrets Manager 317
Versi 318
Prasyarat 318
Izin 318
Persyaratan 316
Contoh kode 319
Apa selanjutnya? 320
Plugin kustom dengan Oracle 320
Versi 321
Prasyarat 321
Izin 321
Persyaratan 321
Contoh kode 322
Buat plugin kustom 322
Opsi konfigurasi Airflow 326
Apa selanjutnya? 326
Mengubah zona waktu DAG 326

Versi ... 327
Prasyarat ... 327
Izin ... 327
Buat plugin untuk mengubah zona waktu di log Airflow ... 327
Buat `plugins.zip` ... 328
Contoh kode ... 328
Apa selanjutnya? ... 329
Menyegarkan AWS CodeArtifact token saat runtime ... 330
Versi ... 330
Prasyarat ... 330
Izin ... 330
Contoh kode ... 331
Apa selanjutnya? ... 332
Plugin kustom dengan Apache Hive dan Hadoop ... 333
Versi ... 333
Prasyarat ... 333
Izin ... 333
Persyaratan ... 316
Unduh dependensi ... 334
PluginKhusus ... 335
Plugins.zip ... 335
Contoh kode ... 336
Opsi konfigurasi Airflow ... 336
Apa selanjutnya? ... 336
Plugin kustom untuk menambal PythonVirtualenvOperator ... 337
Versi ... 337
Prasyarat ... 337
Izin ... 338
Persyaratan ... 338
Kode sampel plugin kustom ... 338
Plugins.zip ... 339
Contoh kode ... 339
Opsi konfigurasi Airflow ... 340
Apa selanjutnya? ... 341
Memohon DAGs dengan Lambda ... 341
Versi ... 341

Prasyarat ... 341
Izin ... 342
Dependensi ... 342
Contoh kode ... 343
Memohon DAGs di lingkungan yang berbeda ... 344
Versi ... 344
Prasyarat ... 344
Izin ... 345
Dependensi ... 345
Contoh kode ... 345
Server Amazon RDS ... 347
Versi ... 348
Prasyarat ... 348
Dependensi ... 309
Koneksi Apache Airflow v2 ... 348
Contoh kode ... 349
Apa selanjutnya? ... 351
Amazon EKS (eksctl) ... 351
Versi ... 352
Prasyarat ... 352
Buat kunci publik untuk Amazon EC2 ... 353
Buat cluster ... 353
Buat `mwaa` namespace ... 354
Buat peran untuk `mwaa` namespace ... 354
Buat dan lampirkan peran IAM untuk klaster Amazon EKS ... 355
Buat file requirements.txt ... 358
Membuat pemetaan identitas untuk Amazon EKS ... 359
Buat `kubeconfig` ... 359
Buat DAG ... 359
Tambahkan DAG dan `kube_config.yaml` ke ember Amazon S3 ... 361
Aktifkan dan picu contoh ... 361
Menggunakan `ECSOperator` ... 361
Versi ... 362
Prasyarat ... 362
Izin ... 362
Buat kluster Amazon ECS ... 364

Contoh kode ... 368
Menggunakan dbt dengan Amazon MWAA ... 371
Versi ... 372
Prasyarat ... 372
Dependensi ... 372
Unggah proyek dbt ke Amazon S3 ... 374
Gunakan DAG untuk memverifikasi instalasi ketergantungan dbt ... 374
Gunakan DAG untuk menjalankan proyek dbt ... 375
AWS Blog dan Tutorial ... 376
Praktik terbaik ... 377
Penyetelan kinerja untuk Apache Airflow ... 377
Menambahkan opsi konfigurasi Apache Airflow ... 377
Penjadwal Apache Airflow ... 378
Folder DAG ... 382
File DAG ... 383
Tugas ... 386
Mengelola dependensi Python ... 390
Menguji DAG menggunakan utilitas Amazon MWAA CLI ... 391
Menginstal dependensi Python menggunakan Format File Persyaratan PyPi.org ... 391
Mengaktifkan log di konsol Amazon MWAA ... 398
Mengakses log di konsol CloudWatch Log ... 398
Mengakses kesalahan di Apache Airflow UI ... 399
Contoh skenario `requirements.txt` ... 400
Pemantauan dan metrik ... 401
Ikhtisar ... 401
CloudWatch Ikhtisar Amazon ... 402
AWS CloudTrail selayang pandang ... 402
Mengakses log audit ... 402
Membuat jejak di CloudTrail ... 403
Mengakses acara dengan Riwayat CloudTrail Acara ... 403
Contoh jejak untuk `CreateEnvironment` ... 403
Apa selanjutnya? ... 405
Mengakses log Aliran Udara ... 405
Harga ... 405
Sebelum Anda mulai ... 406
Jenis log ... 406

Mengaktifkan log Apache Airflow 406
Mengakses log Apache Airflow 407
Contoh log penjadwal 407
Apa selanjutnya? 408
Memantau dasbor dan alarm 408
Metrik-metrik 409
Ikhtisar status alarm 409
Contoh dasbor dan alarm khusus 409
Menghapus metrik dan dasbor 414
Apa selanjutnya? 414
Metrik lingkungan Apache Airflow 414
Istilah 415
Dimensi 415
Mengakses metrik di konsol CloudWatch 416
Metrik Apache Airflow tersedia di CloudWatch 417
Memilih metrik mana yang dilaporkan 437
Apa selanjutnya? 437
Metrik kontainer, antrian, dan basis data 437
Istilah 438
Dimensi 439
Mengakses metrik 439
Daftar metrik 440
Keamanan 444
Perlindungan Data 445
Enkripsi 446
Menggunakan kunci yang dikelola pelanggan 448
AWS Identity and Access Management 451
Audiens 452
Mengautentikasi Menggunakan Identitas 452
Mengelola Akses Menggunakan Kebijakan 454
Memungkinkan pengguna untuk mengakses izin mereka sendiri 455
Memecahkan Masalah Alur Kerja Terkelola Amazon untuk identitas dan akses Apache Airflow 456
Bagaimana Amazon MWAA bekerja dengan IAM 458
Validasi Kepatuhan 463
Ketahanan 464

Keamanan Infrastruktur ... 464
Analisis Konfigurasi dan Kelemahan ... 464
Praktik terbaik ... 465
Praktik terbaik keamanan di Apache Airflow ... 465
Versi ... 468
Tentang versi Amazon MWAA ... 468
Versi terbaru ... 468
Versi Apache Airflow ... 468
Komponen Apache Airflow ... 470
Penjadwal ... 470
Pekerja ... 470
Memutakhirkan versi Apache Airflow ... 471
Menurunkan versi Apache Airflow ... 471
Apache Airflow versi usang ... 471
Dukungan dan FAQ versi Apache Airflow ... 472
Pertanyaan umum ... 472
Titik akhir dan kuota ... 474
Titik akhir layanan ... 474
Kuota layanan ... 474
Meningkatkan kuota ... 475
FAQ ... 476
Versi yang didukung ... 477
Dukungan Apache Airflow ... 477
Versi Python ... 477
Kasus penggunaan ... 479
Bisakah saya menggunakan Amazon MWAA dengan Amazon SageMaker Unified Studio? ... 479
Kapan saya bisa menggunakan AWS Step Functions vs Amazon MWAA? ... 479
Spesifikasi lingkungan ... 479
Berapa banyak penyimpanan tugas yang tersedia untuk setiap lingkungan? ... 479
OS standar ... 479
Gambar kustom ... 480
Kepatuhan HIPAA ... 480
Apakah Amazon MWAA mendukung Instans Spot? ... 480
Domain kustom ... 480
Akses SSH ... 480

Self-referencing aturan 481
Metrik-metrik kustom 481
Menyimpan data 481
Kuota pekerja 482
Amazon VPC Bersama 482
Amazon VPC Bersama 482
Metrik-metrik 482
Metrik pekerja 482
Metrik-metrik kustom 482
DAG, Operator, Koneksi, dan pertanyaan lainnya 483
PythonVirtualenvOperator 483
Berapa lama waktu yang dibutuhkan Amazon MWAA untuk mengenali file DAG baru? 483
Mengapa file DAG saya tidak diambil oleh Apache Airflow? 483
Hapus plugins.zip atau requirements.txt 483
Hapus plugins.zip atau requirements.txt 484
Dapatkah saya menggunakan AWS Operator Database Migration Service (DMS)? 484
Saat saya mengakses Airflow REST API menggunakan AWS kredensialnya, dapatkah saya meningkatkan batas throttling menjadi lebih dari 10 transaksi per detik (TPS)? 484
Di mana server API Eksekusi Tugas Airflow berjalan di Amazon MWAA? 484
Pemecahan Masalah 485
Apache Airflow v2 dan v3 487
Koneksi 488
Server Web 490
Tugas 491
CLI 493
Operator 494
Amazon MWAA Buat/Perbarui 496
Memperbarui `requirements.txt` 497
Plugin 498
Buat ember 498
Ciptakan lingkungan 499
Perbarui lingkungan 501
Lingkungan akses 502
CloudWatch Log dan CloudTrail 502
Beberapa catatan 503
Riwayat Panduan Pengguna Amazon MWAA 509

..... dlxxviii

Apa Alur Kerja yang Dikelola Amazon untuk Apache Airflow?

Gunakan Alur Kerja Terkelola Amazon untuk Apache Airflow, layanan terkelola untuk Apache Airflow, untuk menyiapkan dan menjalankan pipeline data di cloud dalam skala besar. Apache Airflow adalah alat sumber terbuka yang digunakan untuk membuat, menjadwalkan, dan memantau alur kerja.

Dengan Amazon MWAA, Anda dapat menggunakan Apache Airflow dan Python untuk membuat alur kerja tanpa mengelola infrastruktur untuk skalabilitas, ketersediaan, dan keamanan. Amazon MWAA secara otomatis menskalakan untuk memenuhi kebutuhan alur kerja Anda. Ini terintegrasi dengan layanan AWS keamanan untuk menyediakan akses cepat dan aman ke data Anda.

Daftar isi

- Fitur
- Arsitektur
- Integrasi
- Versi yang didukung
- Apa selanjutnya?

Fitur

Tinjau fitur-fitur berikut untuk mempelajari bagaimana Amazon MWAA dapat menyederhanakan pengelolaan alur kerja Apache Airflow Anda.

- Pengaturan Aliran Udara Otomatis - Siapkan Apache Airflow dengan cepat dengan memilih versi Apache Airflow saat Anda membuat lingkungan Amazon MWAA. Amazon MWAA menyiapkan Apache Airflow untuk Anda menggunakan antarmuka pengguna Apache Airflow yang sama dan kode sumber terbuka yang tersedia di internet.
- Penskalaan otomatis — Secara otomatis menskalakan pekerja Apache Airflow (sumber daya komputasi yang menjalankan tugas Anda) dengan menetapkan batas minimum dan maksimum. Amazon MWAA memantau pekerja di lingkungan Anda dan menggunakan komponen penskalaan otomatisnya untuk menambahkan pekerja guna memenuhi permintaan, hingga jumlah maksimum yang Anda tentukan.
- Otentikasi bawaan — Aktifkan otentikasi dan otorisasi berbasis peran untuk server web Apache Airflow Anda dengan mendefinisikan kebijakan kontrol akses di (IAM). AWS Identity and Access

Management Pekerja Apache Airflow mengasumsikan kebijakan ini untuk akses aman ke layanan. AWS

- Keamanan bawaan - Pekerja dan penjadwal Apache Airflow berjalan di Amazon VPC Amazon MWAA Amazon. Data juga dienkripsi secara otomatis menggunakan AWS Key Management Service, sehingga lingkungan Anda aman secara default.
- Mode akses publik atau pribadi — Akses server web Apache Airflow Anda menggunakan mode akses pribadi atau publik. Mode akses jaringan Publik menggunakan titik akhir VPC untuk server web Apache Airflow Anda yang dapat diakses melalui internet. Mode akses jaringan pribadi menggunakan titik akhir VPC untuk server web Apache Airflow Anda yang dapat diakses di VPC Anda. Dalam kedua kasus tersebut, akses untuk pengguna Apache Airflow Anda dikontrol oleh kebijakan kontrol akses yang Anda tentukan AWS Identity and Access Management (IAM), dan SSO. AWS
- Peningkatan dan tambalan yang disederhanakan - Amazon MWAA menyediakan versi baru Apache Airflow secara berkala. Tim Amazon MWAA akan memperbarui dan menambal gambar untuk versi ini.
- Pemantauan alur kerja — akses log Apache Airflow dan metrik Apache Airflow di CloudWatch Amazon untuk mengidentifikasi penundaan tugas Apache Airflow atau kesalahan alur kerja tanpa memerlukan alat pihak ketiga tambahan. Amazon MWAA secara otomatis mengirimkan metrik lingkungan—dan jika diaktifkan—log Apache Airflow ke. CloudWatch
- AWS integrasi - Amazon MWAA mendukung integrasi sumber terbuka dengan Amazon Athena,, Amazon AWS Batch, Amazon CloudWatch DynamoDB,, Amazon AWS DataSync EMR, Amazon EKS, AWS Fargate Amazon Data Firehose,, Amazon AWS Glue AWS Lambda Redshift, Amazon SQS, Amazon SNS, Amazon AI, dan Amazon S3, serta ratusan operator dan sensor bawaan dan dibuat SageMaker komunitas.
- Armada pekerja — Amazon MWAA menawarkan dukungan untuk menggunakan kontainer untuk meningkatkan skala armada pekerja sesuai permintaan dan mengurangi pemadaman penjadwal menggunakan Amazon ECS. AWS Fargate Operator yang menjalankan tugas di container Amazon ECS, dan operator Kubernetes yang membuat dan menjalankan pod pada klaster Kubernetes didukung.

Arsitektur

Semua komponen yang terdapat di kotak luar (pada gambar berikut) ditampilkan sebagai lingkungan Amazon MWAA tunggal di akun Anda. Penjadwal dan pekerja Apache Airflow AWS Fargate adalah wadah yang terhubung ke subnet pribadi di VPC Amazon untuk lingkungan Anda. Setiap lingkungan

memiliki metadatabase Apache Airflow sendiri yang dikelola AWS oleh yang dapat diakses oleh penjadwal dan pekerja kontainer Fargate melalui titik akhir VPC yang diamankan secara pribadi.

Amazon CloudWatch, Amazon S3, Amazon SQS, AWS KMS dan terpisah dari Amazon MWAA dan harus dapat diakses dari penjadwal dan pekerja Apache Airflow di wadah Fargate. Beberapa penjadwal Apache Airflow hanya tersedia dengan Apache Airflow v2 dan yang lebih baru. Pelajari lebih lanjut tentang siklus hidup tugas Apache Airflow di Concepts dalam panduan referensi Apache Airflow.

Server web Apache Airflow dapat diakses baik melalui internet dengan memilih mode akses Apache Airflow jaringan Publik, atau dalam VPC Anda dengan memilih mode akses Apache Airflow jaringan Pribadi. Dalam kedua kasus tersebut, akses untuk pengguna Apache Airflow Anda dikontrol oleh kebijakan kontrol akses yang Anda tentukan AWS Identity and Access Management (IAM).

> **Note**
>
> Dimulai dengan Apache Airflow v3, server web Amazon MWAA juga menghosting server API eksekusi Apache Airflow.

Integrasi

Komunitas open-source Apache Airflow yang aktif dan berkembang menyediakan operator (plugin yang menyederhanakan koneksi ke layanan) untuk Apache Airflow untuk diintegrasikan dengan layanan. AWS Ini termasuk layanan seperti Amazon S3, Amazon Redshift, Amazon AWS Batch EMR, SageMaker dan Amazon AI, serta layanan di platform cloud lainnya.

Menggunakan Apache Airflow dengan Amazon MWAA sepenuhnya mendukung AWS integrasi dengan layanan dan alat pihak ketiga yang populer seperti Apache Hadoop, Presto, Hive, dan Spark untuk melakukan tugas pemrosesan data. Amazon MWAA berkomitmen untuk menjaga kompatibilitas dengan Apache Airflow API, dan Amazon MWAA bermaksud untuk menyediakan AWS integrasi yang andal ke layanan dan membuatnya tersedia bagi komunitas, dan terlibat dalam pengembangan fitur komunitas.

Untuk kode sampel, lihatContoh kode untuk Alur Kerja Terkelola Amazon untuk Apache Airflow.

Versi yang didukung

Amazon MWAA mendukung beberapa versi Apache Airflow. Untuk informasi lebih lanjut tentang versi Apache Airflow yang kami dukung dan komponen Apache Airflow yang disertakan dengan setiap versi, lihat. Versi Apache Airflow di Amazon Managed Workflows for Apache Airflow

Apa selanjutnya?

- Mulailah dengan satu CloudFormation templat yang membuat bucket Amazon S3 untuk Airflow DAGs dan file pendukung Anda, VPC Amazon dengan perutean publik, dan lingkungan Amazon MWAA di dalamnya. Tutorial mulai cepat untuk Alur Kerja Terkelola Amazon untuk Apache Airflow
- Mulailah secara bertahap dengan membuat bucket Amazon S3 untuk DAGs Airflow dan file pendukung Anda, memilih salah satu dari tiga opsi jaringan VPC Amazon, dan menciptakan lingkungan Amazon MWAA di dalamnya. Memulai dengan Amazon Managed Workflows for Apache Airflow

Tutorial mulai cepat untuk Alur Kerja Terkelola Amazon untuk Apache Airflow

Tutorial mulai cepat ini menggunakan AWS CloudFormation template yang membuat infrastruktur VPC Amazon, bucket Amazon S3 dengan `dags` folder, dan Amazon Managed Workflows untuk lingkungan Apache Airflow secara bersamaan.

Topik

- Dalam tutorial ini
- Prasyarat
- Langkah satu: Simpan CloudFormation template secara lokal
- Langkah kedua: Buat tumpukan menggunakan AWS CLI
- Langkah ketiga: Unggah DAG ke Amazon S3 dan jalankan di Apache Airflow UI
- Langkah empat: akses log di CloudWatch Log
- Apa selanjutnya?

Dalam tutorial ini

Gunakan tutorial ini untuk mengunggah DAG ke Amazon S3, jalankan DAG di Apache Airflow, dan akses log CloudWatch menggunakan perintah tiga (). AWS Command Line Interface AWS CLI Akhirnya, Anda akan belajar membuat kebijakan IAM untuk tim pengembangan Apache Airflow.

> **Note**
>
> CloudFormation Template pada halaman ini membuat Alur Kerja Terkelola Amazon untuk lingkungan Apache Airflow untuk versi terbaru Apache Airflow yang tersedia di. CloudFormation Versi terbaru yang tersedia adalah Apache Airflow v3.0.6.

CloudFormation Template menciptakan yang berikut:

- Infrastruktur VPC. Template menggunakanRouting publik melalui internet. Ini menggunakan Mode akses jaringan publik untuk server web Apache Airflow di. `WebserverAccessMode: PUBLIC_ONLY`

- Ember Amazon S3. Template membuat bucket Amazon S3 dengan folder. `dags` Ini dikonfigurasi untuk Memblokir semua akses publik, dengan Bucket Versioning diaktifkan, seperti yang didefinisikan dalam. Buat bucket Amazon S3 untuk Amazon MWAA
- Lingkungan Amazon MWAA. Template membuat lingkungan Amazon MWAA yang terkait dengan `dags` folder di bucket Amazon S3, peran eksekusi dengan izin AWS ke layanan yang digunakan oleh Amazon MWAA, dan default untuk enkripsi menggunakan kunci yang dimiliki, seperti AWS yang didefinisikan dalam. Buat lingkungan Amazon MWAA
- CloudWatch Log. Template mengaktifkan log Apache Airflow CloudWatch di tingkat INFO dan ke atas untuk grup log penjadwal Airflow, grup log server web Airflow, grup log pekerja Airflow, grup log pemrosesan Airflow DAG, dan grup log tugas Airflow, seperti yang didefinisikan dalam. Mengakses log Aliran Udara di Amazon CloudWatch

Dalam tutorial ini, Anda akan menyelesaikan tugas-tugas berikut:

- Unggah dan jalankan DAG. Unggah tutorial Apache Airflow DAG untuk Amazon MWAA terbaru yang didukung versi Apache Airflow ke Amazon S3, lalu jalankan di Apache Airflow UI, seperti yang didefinisikan dalam. Menambahkan atau memperbarui DAGs
- Akses log. Akses grup log server web Airflow di CloudWatch Log, seperti yang didefinisikan dalam. Mengakses log Aliran Udara di Amazon CloudWatch
- Buat kebijakan kontrol akses. Buat kebijakan kontrol akses di IAM untuk tim pengembangan Apache Airflow Anda, sebagaimana didefinisikan dalam. Mengakses lingkungan Amazon MWAA

> **Note**
>
> Di VPC yang menjadi tuan rumah lingkungan Amazon MWAA, diatur `assignIpv6AddressOnCreation` ke `true` untuk semua subnet yang terpasang. Pengaturan ini memastikan penetapan alamat Internet Protocol versi 6 (IPv6) otomatis ke sumber daya dalam subnet ini.

Prasyarat

The AWS Command Line Interface (AWS CLI) adalah alat open source yang dapat Anda gunakan untuk berinteraksi dengan AWS layanan menggunakan perintah di shell baris perintah Anda. Untuk menyelesaikan langkah-langkah di halaman ini, Anda memerlukan yang berikut:

- AWS CLI — Instal versi 2.
- AWS CLI - Konfigurasi cepat dengan `aws configure`.

Langkah satu: Simpan CloudFormation template secara lokal

- Salin isi template berikut dan simpan secara lokal sebagai`mwaa-public-network.yml`. Anda juga dapat mengunduh template.

```
AWSTemplateFormatVersion: "2010-09-09"

Parameters:

  EnvironmentName:
    Description: An environment name that is prefixed to resource names
    Type: String
    Default: MWAAEnvironment

  VpcCIDR:
    Description: The IP range (CIDR notation) for this VPC
    Type: String
    Default: 10.192.0.0/16

  PublicSubnet1CIDR:
    Description: The IP range (CIDR notation) for the public subnet in the first
 Availability Zone
    Type: String
    Default: 10.192.10.0/24

  PublicSubnet2CIDR:
    Description: The IP range (CIDR notation) for the public subnet in the second
 Availability Zone
    Type: String
    Default: 10.192.11.0/24

  PrivateSubnet1CIDR:
    Description: The IP range (CIDR notation) for the private subnet in the first
 Availability Zone
    Type: String
    Default: 10.192.20.0/24
  PrivateSubnet2CIDR:
```

```
    Description: The IP range (CIDR notation) for the private subnet in the second
 Availability Zone
    Type: String
    Default: 10.192.21.0/24
  MaxWorkerNodes:
    Description: The maximum number of workers that can run in the environment
    Type: Number
    Default: 2
  DagProcessingLogs:
    Description: Log level for DagProcessing
    Type: String
    Default: INFO
  SchedulerLogsLevel:
    Description: Log level for SchedulerLogs
    Type: String
    Default: INFO
  TaskLogsLevel:
    Description: Log level for TaskLogs
    Type: String
    Default: INFO
  WorkerLogsLevel:
    Description: Log level for WorkerLogs
    Type: String
    Default: INFO
  WebserverLogsLevel:
    Description: Log level for WebserverLogs
    Type: String
    Default: INFO

Resources:

 ##################################################################################################
  # CREATE VPC

 ##################################################################################################

  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcCIDR
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Name
```

```
        Value: MWAAEnvironment

  InternetGateway:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
        - Key: Name
          Value: MWAAEnvironment

  InternetGatewayAttachment:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      InternetGatewayId: !Ref InternetGateway
      VpcId: !Ref VPC

  PublicSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs '' ]
      CidrBlock: !Ref PublicSubnet1CIDR
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Subnet (AZ1)

  PublicSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs  '' ]
      CidrBlock: !Ref PublicSubnet2CIDR
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Subnet (AZ2)

  PrivateSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs  '' ]
      CidrBlock: !Ref PrivateSubnet1CIDR
      MapPublicIpOnLaunch: false
```

```
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Subnet (AZ1)

  PrivateSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs  '' ]
      CidrBlock: !Ref PrivateSubnet2CIDR
      MapPublicIpOnLaunch: false
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Subnet (AZ2)

  NatGateway1EIP:
    Type: AWS::EC2::EIP
    DependsOn: InternetGatewayAttachment
    Properties:
      Domain: vpc

  NatGateway2EIP:
    Type: AWS::EC2::EIP
    DependsOn: InternetGatewayAttachment
    Properties:
      Domain: vpc

  NatGateway1:
    Type: AWS::EC2::NatGateway
    Properties:
      AllocationId: !GetAtt NatGateway1EIP.AllocationId
      SubnetId: !Ref PublicSubnet1

  NatGateway2:
    Type: AWS::EC2::NatGateway
    Properties:
      AllocationId: !GetAtt NatGateway2EIP.AllocationId
      SubnetId: !Ref PublicSubnet2

  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
```

```
      - Key: Name
        Value: !Sub ${EnvironmentName} Public Routes

DefaultPublicRoute:
  Type: AWS::EC2::Route
  DependsOn: InternetGatewayAttachment
  Properties:
    RouteTableId: !Ref PublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway

PublicSubnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PublicRouteTable
    SubnetId: !Ref PublicSubnet1

PublicSubnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PublicRouteTable
    SubnetId: !Ref PublicSubnet2


PrivateRouteTable1:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Private Routes (AZ1)

DefaultPrivateRoute1:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable1
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NatGateway1

PrivateSubnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PrivateRouteTable1
    SubnetId: !Ref PrivateSubnet1
```

```
  PrivateRouteTable2:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Routes (AZ2)

  DefaultPrivateRoute2:
    Type: AWS::EC2::Route
    Properties:
      RouteTableId: !Ref PrivateRouteTable2
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId: !Ref NatGateway2

  PrivateSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PrivateRouteTable2
      SubnetId: !Ref PrivateSubnet2

  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: "mwaa-security-group"
      GroupDescription: "Security group with a self-referencing inbound rule."
      VpcId: !Ref VPC

  SecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !Ref SecurityGroup
      IpProtocol: "-1"
      SourceSecurityGroupId: !Ref SecurityGroup

  EnvironmentBucket:
    Type: AWS::S3::Bucket
    Properties:
      VersioningConfiguration:
        Status: Enabled
      PublicAccessBlockConfiguration:
        BlockPublicAcls: true
        BlockPublicPolicy: true
```

```
        IgnorePublicAcls: true
        RestrictPublicBuckets: true


###########################################################################################
 # CREATE MWAA

###########################################################################################

 MwaaEnvironment:
   Type: AWS::MWAA::Environment
   DependsOn: MwaaExecutionPolicy
   Properties:
     Name: !Sub "${AWS::StackName}-MwaaEnvironment"
     SourceBucketArn: !GetAtt EnvironmentBucket.Arn
     ExecutionRoleArn: !GetAtt MwaaExecutionRole.Arn
     DagS3Path: dags/
     NetworkConfiguration:
       SecurityGroupIds:
         - !GetAtt SecurityGroup.GroupId
       SubnetIds:
         - !Ref PrivateSubnet1
         - !Ref PrivateSubnet2
     WebserverAccessMode: PUBLIC_ONLY
     MaxWorkers: !Ref MaxWorkerNodes
     LoggingConfiguration:
       DagProcessingLogs:
         LogLevel: !Ref DagProcessingLogs
         Enabled: true
       SchedulerLogs:
         LogLevel: !Ref SchedulerLogsLevel
         Enabled: true
       TaskLogs:
         LogLevel: !Ref TaskLogsLevel
         Enabled: true
       WorkerLogs:
         LogLevel: !Ref WorkerLogsLevel
         Enabled: true
       WebserverLogs:
         LogLevel: !Ref WebserverLogsLevel
         Enabled: true

 MwaaExecutionRole:
   Type: AWS::IAM::Role
```

```
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17&TCX5-2025-waiver;
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - airflow-env.amazonaws.com
                - airflow.amazonaws.com
            Action:
             - "sts:AssumeRole"
      Path: "/service-role/"

  MwaaExecutionPolicy:
    DependsOn: EnvironmentBucket
    Type: AWS::IAM::ManagedPolicy
    Properties:
      Roles:
        - !Ref MwaaExecutionRole
      PolicyDocument:
        Version: 2012-10-17&TCX5-2025-waiver;
        Statement:
          - Effect: Allow
            Action: airflow:PublishMetrics
            Resource:
              - !Sub "arn:aws:airflow:${AWS::Region}:${AWS::AccountId}:environment/
${EnvironmentName}"
          - Effect: Deny
            Action: s3:ListAllMyBuckets
            Resource:
              - !Sub "${EnvironmentBucket.Arn}"
              - !Sub "${EnvironmentBucket.Arn}/*"

          - Effect: Allow
            Action:
              - "s3:GetObject*"
              - "s3:GetBucket*"
              - "s3:List*"
            Resource:
              - !Sub "${EnvironmentBucket.Arn}"
              - !Sub "${EnvironmentBucket.Arn}/*"
          - Effect: Allow
            Action:
              - logs:DescribeLogGroups
```

```
            Resource: "*"

          - Effect: Allow
            Action:
              - logs:CreateLogStream
              - logs:CreateLogGroup
              - logs:PutLogEvents
              - logs:GetLogEvents
              - logs:GetLogRecord
              - logs:GetLogGroupFields
              - logs:GetQueryResults
              - logs:DescribeLogGroups
            Resource:
              - !Sub "arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-
group:airflow-${AWS::StackName}*"
          - Effect: Allow
            Action: cloudwatch:PutMetricData
            Resource: "*"
          - Effect: Allow
            Action:
              - sqs:ChangeMessageVisibility
              - sqs:DeleteMessage
              - sqs:GetQueueAttributes
              - sqs:GetQueueUrl
              - sqs:ReceiveMessage
              - sqs:SendMessage
            Resource:
              - !Sub "arn:aws:sqs:${AWS::Region}:*:airflow-celery-*"
          - Effect: Allow
            Action:
              - kms:Decrypt
              - kms:DescribeKey
              - "kms:GenerateDataKey*"
              - kms:Encrypt
            NotResource: !Sub "arn:aws:kms:*:${AWS::AccountId}:key/*"
            Condition:
              StringLike:
                "kms:ViaService":
                  - !Sub "sqs.${AWS::Region}.amazonaws.com"
Outputs:
  VPC:
    Description: A reference to the created VPC
    Value: !Ref VPC
```

```
  PublicSubnets:
    Description: A list of the public subnets
    Value: !Join [ ",", [ !Ref PublicSubnet1, !Ref PublicSubnet2 ]]

  PrivateSubnets:
    Description: A list of the private subnets
    Value: !Join [ ",", [ !Ref PrivateSubnet1, !Ref PrivateSubnet2 ]]

  PublicSubnet1:
    Description: A reference to the public subnet in the 1st Availability Zone
    Value: !Ref PublicSubnet1

  PublicSubnet2:
    Description: A reference to the public subnet in the 2nd Availability Zone
    Value: !Ref PublicSubnet2

  PrivateSubnet1:
    Description: A reference to the private subnet in the 1st Availability Zone
    Value: !Ref PrivateSubnet1

  PrivateSubnet2:
    Description: A reference to the private subnet in the 2nd Availability Zone
    Value: !Ref PrivateSubnet2

  SecurityGroupIngress:
    Description: Security group with self-referencing inbound rule
    Value: !Ref SecurityGroupIngress

  MwaaApacheAirflowUI:
    Description: MWAA Environment
    Value: !Sub  "https://${MwaaEnvironment.WebserverUrl}"
```

Langkah kedua: Buat tumpukan menggunakan AWS CLI

1. Di prompt perintah Anda, arahkan ke direktori tempat `mwaa-public-network.yml` disimpan. Contoh:

```
cd mwaaproject
```

2. Gunakan `aws cloudformation create-stack` perintah untuk membuat tumpukan menggunakan file AWS CLI.

```
aws cloudformation create-stack --stack-name mwaa-environment-public-network --
template-body file://mwaa-public-network.yml --capabilities CAPABILITY_IAM
```

> **Note**
>
> Dibutuhkan lebih dari 30 menit untuk membuat infrastruktur VPC Amazon, bucket Amazon S3, dan lingkungan Amazon MWAA.

Langkah ketiga: Unggah DAG ke Amazon S3 dan jalankan di Apache Airflow UI

1. Salin konten `tutorial.py` file untuk versi Apache Airflow terbaru yang didukung dan simpan secara lokal sebagai file. `tutorial.py`
2. Di prompt perintah Anda, arahkan ke direktori tempat `tutorial.py` disimpan. Contoh:

   ```
   cd mwaaproject
   ```
3. Gunakan perintah berikut untuk membuat daftar semua bucket Amazon S3 Anda.

   ```
   aws s3 ls
   ```
4. Gunakan perintah berikut untuk mencantumkan file dan folder di bucket Amazon S3 untuk lingkungan Anda.

   ```
   aws s3 ls s3://YOUR_S3_BUCKET_NAME
   ```
5. Gunakan skrip berikut untuk mengunggah `tutorial.py` file ke `dags` folder Anda. Gantikan nilai sampel di*amzn-s3-demo-bucket*.

   ```
   aws s3 cp tutorial.py s3://amzn-s3-demo-bucket/dags/
   ```
6. Buka halaman Lingkungan di konsol Amazon MWAA.
7. Pilih lingkungan.
8. Pilih Buka UI Aliran Udara.
9. Pada Apache Airflow UI, dari daftar yang DAGs tersedia, pilih tutorial DAG.

10. Pada halaman detail DAG, pilih sakelar Jeda/Unpause DAG di sebelah nama DAG Anda untuk membatalkan jeda DAG.
11. Pilih Trigger DAG.

Langkah empat: akses log di CloudWatch Log

Anda dapat mengakses log Apache Airflow di CloudWatch konsol untuk semua log Apache Airflow yang diaktifkan oleh tumpukan. CloudFormation Bagian berikut menjelaskan cara mengakses log untuk grup log server web Airflow.

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.
3. Pilih grup log server web Aliran udara di panel Pemantauan.
4. Pilih `webserver_console_ip` log masuk Aliran log.

Apa selanjutnya?

- Pelajari lebih lanjut tentang cara mengunggah DAGs, menentukan dependensi Python di plugin `requirements.txt` dan kustom di dalam. `plugins.zip` Bekerja dengan DAGs di Amazon MWAA
- Pelajari lebih lanjut tentang praktik terbaik yang kami rekomendasikan untuk menyesuaikan kinerja lingkungan AndaPenyetelan kinerja untuk Apache Airflow di Amazon MWAA.
- Buat dasbor pemantauan untuk lingkungan Anda diMemantau dasbor dan alarm di Amazon MWAA.
- Jalankan beberapa contoh kode DAG diContoh kode untuk Alur Kerja Terkelola Amazon untuk Apache Airflow.

Memulai dengan Amazon Managed Workflows for Apache Airflow

Alur Kerja Terkelola Amazon untuk Apache Airflow menggunakan file Amazon VPC, DAG, dan file pendukung di bucket penyimpanan Amazon S3 Anda untuk menciptakan lingkungan. Bab ini menjelaskan prasyarat dan AWS sumber daya yang dibutuhkan untuk memulai Amazon MWAA.

Topik

- Prasyarat
- Tentang panduan ini
- Sebelum Anda mulai
- Wilayah yang tersedia
- Buat bucket Amazon S3 untuk Amazon MWAA
- Buat jaringan VPC
- Buat lingkungan Amazon MWAA
- Apa selanjutnya?

Prasyarat

Untuk membuat lingkungan Amazon MWAA, pastikan Anda memiliki izin untuk AWS sumber daya yang perlu Anda buat.

- Akun AWS— Akun AWS Dengan izin untuk menggunakan Amazon MWAA dan AWS layanan serta sumber daya yang digunakan oleh lingkungan Anda.

Tentang panduan ini

Panduan ini mencakup AWS infrastruktur dan sumber daya yang akan Anda buat.

- Amazon VPC - Komponen jaringan Amazon VPC yang dibutuhkan oleh lingkungan Amazon MWAA. Anda dapat mengonfigurasi VPC yang ada yang memenuhi persyaratan ini (lanjutan) seperti yang ditemukan diTentang jaringan di Amazon MWAA, atau membuat VPC dan komponen jaringan, seperti yang didefinisikan dalam. the section called “Buat jaringan VPC”

- Bucket Amazon S3 — Bucket Amazon S3 untuk menyimpan file DAGs Anda dan file terkait, seperti dan. `plugins.zip` `requirements.txt` Bucket Amazon S3 Anda harus dikonfigurasi untuk Memblokir semua akses publik, dengan Pembuatan Versi Bucket diaktifkan, seperti yang didefinisikan dalam. Buat bucket Amazon S3 untuk Amazon MWAA
- Lingkungan Amazon MWAA - Lingkungan Amazon MWAA yang dikonfigurasi dengan lokasi bucket Amazon S3 Anda, jalur ke kode DAG Anda dan semua plugin khusus atau dependensi Python, dan VPC Amazon Anda dan grup keamanannya, sebagaimana didefinisikan dalam. Buat lingkungan Amazon MWAA

Sebelum Anda mulai

Untuk membuat lingkungan Amazon MWAA, Anda dapat mengambil langkah tambahan untuk membuat dan mengonfigurasi AWS sumber daya lain sebelum membuat lingkungan.

Untuk menciptakan lingkungan, Anda memerlukan yang berikut ini:

- AWS KMS kunci — AWS KMS Kunci untuk enkripsi data di lingkungan Anda. Anda dapat memilih opsi default di konsol Amazon MWAA untuk membuat kunci yang AWS dimiliki saat Anda membuat lingkungan, atau menentukan kunci yang dikelola Pelanggan yang ada dengan izin ke AWS layanan lain yang digunakan oleh lingkungan yang dikonfigurasi (lanjutan). Untuk mempelajari lebih lanjut, lihatMenggunakan kunci yang dikelola pelanggan untuk enkripsi.
- Peran eksekusi — Peran eksekusi yang memungkinkan Amazon MWAA mengakses AWS sumber daya di lingkungan Anda. Anda dapat memilih opsi default di konsol Amazon MWAA untuk membuat peran eksekusi saat Anda membuat lingkungan. Untuk mempelajari lebih lanjut, lihatPeran eksekusi Amazon MWAA.
- Grup keamanan VPC — Grup keamanan VPC yang memungkinkan Amazon MWAA mengakses sumber daya AWS lain di jaringan VPC Anda. Anda dapat memilih opsi default di konsol Amazon MWAA untuk membuat grup keamanan saat Anda membuat lingkungan, atau menyediakan grup keamanan dengan aturan masuk dan keluar yang sesuai (lanjutan). Untuk mempelajari lebih lanjut, lihatKeamanan di VPC Anda di Amazon MWAA.

Wilayah yang tersedia

Amazon MWAA tersedia dalam hal berikut. Wilayah AWS Untuk mempelajari lebih lanjut tentang setiap wilayah, seperti yang diaktifkan atau dinonaktifkan secara default, lihat Wilayah AWS.

Code	Nama
us-east-1	AS Timur (Virginia Utara)
us-east-2	AS Timur (Ohio)
us-west-1	AS Barat (California Utara)
us-west-2	AS Barat (Oregon)
af-south-1	Africa (Cape Town)
ap-east-1	Asia Pasifik (Hong Kong)
ap-south-2	Asia Pasifik (Hyderabad)
ap-southeast-3	Asia Pasifik (Jakarta)
ap-southeast-5	Asia Pasifik (Malaysia)
ap-southeast-4	Asia Pacific (Melbourne)
ap-south-1	Asia Pasifik (Mumbai)
ap-northeast-3	Asia Pasifik (Osaka)
ap-northeast-2	Asia Pasifik (Seoul)
ap-southeast-1	Asia Pasifik (Singapura)
ap-southeast-2	Asia Pasifik (Sydney)
ap-northeast-1	Asia Pasifik (Tokyo)
ca-central-1	Kanada (Pusat)
ca-west-1	Kanada Barat (Calgary)
eu-central-1	Eropa (Frankfurt)
eu-west-1	Eropa (Irlandia)

Code	Nama
eu-west-2	Eropa (London)
eu-south-1	Europe (Milan)
eu-west-3	Eropa (Paris)
eu-south-2	Eropa (Spanyol)
eu-north-1	Eropa (Stockholm)
eu-central-2	Europe (Zurich)
il-central-1	Israel (Tel Aviv)
me-south-1	Timur Tengah (Bahrain)
me-central-1	Timur Tengah (UAE)
sa-east-1	Amerika Selatan (Sao Paulo)

Buat bucket Amazon S3 untuk Amazon MWAA

Panduan ini menjelaskan langkah-langkah untuk membuat bucket Amazon S3 untuk menyimpan Apache Airflow Directed Acyclic Graphs DAGs (), plugin kustom dalam file, dan dependensi Python dalam file. `plugins.zip` `requirements.txt`

Daftar Isi

- Sebelum Anda mulai
- Buat ember
- Apa selanjutnya?

Sebelum Anda mulai

- Nama bucket Amazon S3 tidak dapat diubah setelah Anda membuat bucket. Untuk mempelajari selengkapnya, lihat Aturan penamaan bucket di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

- Bucket Amazon S3 yang digunakan untuk lingkungan Amazon MWAA harus dikonfigurasi untuk Memblokir semua akses publik, dengan Pembuatan Versi Bucket diaktifkan.
- Bucket Amazon S3 yang digunakan untuk lingkungan Amazon MWAA harus ditempatkan sama dengan lingkungan Amazon Wilayah AWS MWAA. Untuk mengakses daftar Wilayah AWS untuk Amazon MWAA, lihat titik akhir dan kuota Amazon MWAA di. Referensi Umum AWS

Buat ember

Bagian ini menjelaskan langkah-langkah untuk membuat bucket Amazon S3 untuk lingkungan Anda.

Untuk membuat bucket

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon S3 di. https://console.aws.amazon.com/s3/
2. Pilih Buat bucket.
3. Di Nama bucket, masukkan nama yang sesuai dengan DNS untuk bucket Anda.

   Nama kelompok harus:

   - Unik di seluruh Amazon S3.
   - Panjangnya antara 3 dan 63 karakter.
   - Tidak mengandung karakter huruf besar.
   - Mulai dengan huruf kecil atau angka.

   > **Important**
   >
   > Hindari menyertakan informasi sensitif, seperti nomor akun, dalam nama bucket. Nama ember tersedia di titik URLs itu ke objek di ember.

4. Pilih Wilayah AWS di Wilayah. Ini harus Wilayah AWS sama dengan lingkungan Amazon MWAA Anda.

   - Sebaiknya pilih wilayah yang dekat dengan Anda untuk meminimalkan latensi dan biaya serta memenuhi persyaratan peraturan.

5. Pilih Blokir semua akses publik.
6. Pilih Aktifkan dalam Pembuatan Versi Bucket.

7. Opsional - Tag. Tambahkan pasangan tag nilai kunci untuk mengidentifikasi bucket Amazon S3 Anda di Tag. Misalnya, `Bucket : Staging`.
8. Opsional - Enkripsi sisi server. Anda dapat secara opsional Mengaktifkan salah satu opsi enkripsi berikut di bucket Amazon S3 Anda.

    a. Pilih kunci Amazon S3 (SSE-S3) di enkripsi sisi server untuk mengaktifkan enkripsi sisi server untuk bucket.

    b. Pilih AWS Key Management Service kunci (SSE-KMS) untuk menggunakan AWS KMS kunci enkripsi di bucket Amazon S3 Anda:

        i. AWS kunci terkelola (aws/s3) - Jika Anda memilih opsi ini, Anda dapat menggunakan kunci AWS milik yang dikelola oleh Amazon MWAA, atau menentukan kunci yang dikelola Pelanggan untuk enkripsi lingkungan Amazon MWAA Anda.

        ii. Pilih dari AWS KMS kunci Anda atau Masukkan AWS KMS kunci ARN - Jika Anda memilih untuk menentukan kunci yang dikelola Pelanggan dalam langkah ini, Anda harus menentukan AWS KMS ID kunci atau ARN. AWS KMS alias dan kunci multi-wilayah tidak didukung oleh Amazon MWAA. AWS KMS Kunci yang Anda tentukan juga harus digunakan untuk enkripsi di lingkungan Amazon MWAA Anda.

9. Opsional - Pengaturan lanjutan. Jika Anda ingin mengaktifkan Amazon S3 Object Lock:

    a. Pilih Pengaturan lanjutan, Aktifkan.

    > **Important**
    >
    > Mengaktifkan Object Lock akan secara permanen memungkinkan objek dalam bucket ini untuk dikunci. Untuk mempelajari lebih lanjut, lihat Mengunci Objek Menggunakan Kunci Objek Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

    b. Pilih pengakuan.

10. Pilih Buat bucket.

Apa selanjutnya?

- Pelajari cara membuat jaringan VPC Amazon yang diperlukan untuk lingkungan di. Buat jaringan VPC

- Pelajari cara mengelola izin akses di Bagaimana cara menyetel izin bucket ACL?
- Pelajari cara menghapus bucket penyimpanan di Bagaimana cara menghapus Bucket S3? .

Buat jaringan VPC

Alur Kerja Terkelola Amazon untuk Apache Airflow memerlukan VPC Amazon dan komponen jaringan tertentu untuk mendukung lingkungan. Panduan ini menjelaskan berbagai opsi untuk membuat jaringan VPC Amazon untuk lingkungan Alur Kerja Terkelola Amazon untuk Apache Airflow.

> **Note**
>
> Apache Airflow bekerja paling baik di lingkungan jaringan latensi rendah. Jika Anda menggunakan VPC Amazon yang sudah ada yang merutekan lalu lintas ke wilayah lain atau ke lingkungan lokal, sebaiknya tambahkan titik akhir AWS PrivateLink untuk Amazon SQS, Amazon S3, dan. CloudWatch AWS KMS Untuk informasi selengkapnya tentang mengonfigurasi AWS PrivateLink Amazon MWAA, lihat Membuat jaringan VPC Amazon tanpa akses internet.

Daftar Isi

- Prasyarat
- Sebelum Anda mulai
- Opsi untuk membuat jaringan VPC Amazon
  - Opsi satu: Membuat jaringan VPC di konsol Amazon MWAA
  - Opsi dua: Membuat jaringan VPC Amazon dengan akses internet
  - Opsi tiga: Membuat jaringan VPC Amazon tanpa akses internet
- Apa selanjutnya?

Prasyarat

The AWS Command Line Interface (AWS CLI) adalah alat open source yang dapat Anda gunakan untuk berinteraksi dengan AWS layanan menggunakan perintah di shell baris perintah Anda. Untuk menyelesaikan langkah-langkah di halaman ini, Anda memerlukan yang berikut:

- AWS CLI — Instal versi 2.
- AWS CLI - Konfigurasi cepat dengan `aws configure`.

Sebelum Anda mulai

- Jaringan VPC yang Anda tentukan untuk lingkungan Anda tidak dapat diubah setelah lingkungan dibuat.
- Anda dapat menggunakan perutean pribadi atau publik untuk server web Amazon VPC dan Apache Airflow Anda. Untuk mengakses daftar opsi, lihatthe section called “Contoh kasus penggunaan untuk VPC Amazon dan mode akses Apache Airflow”.

Opsi untuk membuat jaringan VPC Amazon

Bagian berikut menjelaskan opsi yang tersedia untuk membuat jaringan VPC Amazon untuk suatu lingkungan.

> **Note**
>
> Amazon MWAA tidak mendukung penggunaan `use1-az3` Availability Zone (AZ) di Wilayah AS Timur (Virginia N.). Saat membuat VPC untuk Amazon MWAA di wilayah AS Timur (Virginia N.), Anda harus secara eksplisit menetapkan templat in (CFN). `AvailabilityZone` CloudFormation Nama zona ketersediaan yang ditetapkan tidak boleh dipetakan. `use1-az3` Anda dapat mengambil pemetaan rinci nama AZ ke AZ yang sesuai IDs dengan menjalankan perintah berikut:
>
> ```
> aws ec2 describe-availability-zones --region us-east-1
> ```

Opsi satu: Membuat jaringan VPC di konsol Amazon MWAA

Bagian berikut menjelaskan cara membuat jaringan VPC Amazon di konsol Amazon MWAA. Opsi ini menggunakanRouting publik melalui internet. Ini dapat digunakan untuk server web Apache Airflow dengan jaringan Pribadi atau mode akses jaringan Publik.

Gambar berikut menggambarkan di mana Anda dapat menemukan tombol Create MWAA VPC di konsol Amazon MWAA.

Opsi dua: Membuat jaringan VPC Amazon dengan akses internet

CloudFormation Template berikut membuat jaringan VPC Amazon dengan akses internet di default Anda. Wilayah AWS Opsi ini menggunakanRouting publik melalui internet. Template ini dapat digunakan untuk webserver Apache Airflow dengan jaringan Private atau mode akses jaringan Publik.

1. Salin isi template berikut dan simpan secara lokal sebagai`cfn-vpc-public-private.yaml`. Anda juga dapat mengunduh template.

```
Description:  This template deploys a VPC, with a pair of public and private
 subnets spread
  across two Availability Zones. It deploys an internet gateway, with a default
  route on the public subnets. It deploys a pair of NAT gateways (one in each AZ),
  and default routes for them in the private subnets.

Parameters:
  EnvironmentName:
    Description: An environment name that is prefixed to resource names
    Type: String
    Default: mwaa-

  VpcCIDR:
    Description: Please enter the IP range (CIDR notation) for this VPC
    Type: String
    Default: 10.192.0.0/16

  PublicSubnet1CIDR:
    Description: Please enter the IP range (CIDR notation) for the public subnet in
 the first Availability Zone
    Type: String
    Default: 10.192.10.0/24

  PublicSubnet2CIDR:
    Description: Please enter the IP range (CIDR notation) for the public subnet in
 the second Availability Zone
    Type: String
    Default: 10.192.11.0/24

  PrivateSubnet1CIDR:
    Description: Please enter the IP range (CIDR notation) for the private subnet
 in the first Availability Zone
    Type: String
    Default: 10.192.20.0/24
```

```
  PrivateSubnet2CIDR:
    Description: Please enter the IP range (CIDR notation) for the private subnet
 in the second Availability Zone
    Type: String
    Default: 10.192.21.0/24

Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcCIDR
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Name
          Value: !Ref EnvironmentName

  InternetGateway:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
        - Key: Name
          Value: !Ref EnvironmentName

  InternetGatewayAttachment:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      InternetGatewayId: !Ref InternetGateway
      VpcId: !Ref VPC

  PublicSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs '' ]
      CidrBlock: !Ref PublicSubnet1CIDR
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Subnet (AZ1)

  PublicSubnet2:
    Type: AWS::EC2::Subnet
```

```
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs  '' ]
      CidrBlock: !Ref PublicSubnet2CIDR
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Subnet (AZ2)

  PrivateSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs  '' ]
      CidrBlock: !Ref PrivateSubnet1CIDR
      MapPublicIpOnLaunch: false
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Subnet (AZ1)

  PrivateSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs  '' ]
      CidrBlock: !Ref PrivateSubnet2CIDR
      MapPublicIpOnLaunch: false
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Subnet (AZ2)

  NatGateway1EIP:
    Type: AWS::EC2::EIP
    DependsOn: InternetGatewayAttachment
    Properties:
      Domain: vpc

  NatGateway2EIP:
    Type: AWS::EC2::EIP
    DependsOn: InternetGatewayAttachment
    Properties:
      Domain: vpc

  NatGateway1:
```

```
    Type: AWS::EC2::NatGateway
    Properties:
      AllocationId: !GetAtt NatGateway1EIP.AllocationId
      SubnetId: !Ref PublicSubnet1

  NatGateway2:
    Type: AWS::EC2::NatGateway
    Properties:
      AllocationId: !GetAtt NatGateway2EIP.AllocationId
      SubnetId: !Ref PublicSubnet2

  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Routes

  DefaultPublicRoute:
    Type: AWS::EC2::Route
    DependsOn: InternetGatewayAttachment
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway

  PublicSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet1

  PublicSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet2


  PrivateRouteTable1:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
```

```
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Routes (AZ1)

  DefaultPrivateRoute1:
    Type: AWS::EC2::Route
    Properties:
      RouteTableId: !Ref PrivateRouteTable1
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId: !Ref NatGateway1

  PrivateSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PrivateRouteTable1
      SubnetId: !Ref PrivateSubnet1

  PrivateRouteTable2:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Routes (AZ2)

  DefaultPrivateRoute2:
    Type: AWS::EC2::Route
    Properties:
      RouteTableId: !Ref PrivateRouteTable2
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId: !Ref NatGateway2

  PrivateSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PrivateRouteTable2
      SubnetId: !Ref PrivateSubnet2

  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: "mwaa-security-group"
      GroupDescription: "Security group with a self-referencing inbound rule."
      VpcId: !Ref VPC
```

```
  SecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !Ref SecurityGroup
      IpProtocol: "-1"
      SourceSecurityGroupId: !Ref SecurityGroup

Outputs:
  VPC:
    Description: A reference to the created VPC
    Value: !Ref VPC

  PublicSubnets:
    Description: A list of the public subnets
    Value: !Join [ ",", [ !Ref PublicSubnet1, !Ref PublicSubnet2 ]]

  PrivateSubnets:
    Description: A list of the private subnets
    Value: !Join [ ",", [ !Ref PrivateSubnet1, !Ref PrivateSubnet2 ]]

  PublicSubnet1:
    Description: A reference to the public subnet in the 1st Availability Zone
    Value: !Ref PublicSubnet1

  PublicSubnet2:
    Description: A reference to the public subnet in the 2nd Availability Zone
    Value: !Ref PublicSubnet2

  PrivateSubnet1:
    Description: A reference to the private subnet in the 1st Availability Zone
    Value: !Ref PrivateSubnet1

  PrivateSubnet2:
    Description: A reference to the private subnet in the 2nd Availability Zone
    Value: !Ref PrivateSubnet2

  SecurityGroupIngress:
    Description: Security group with self-referencing inbound rule
    Value: !Ref SecurityGroupIngress
```

2. Di prompt perintah Anda, arahkan ke direktori tempat `cfn-vpc-public-private.yaml` disimpan. Contoh:

```
cd mwaaproject
```

3. Gunakan `aws cloudformation create-stack`perintah untuk membuat tumpukan menggunakan AWS CLI.

```
aws cloudformation create-stack --stack-name mwaa-environment --template-body
 file://cfn-vpc-public-private.yaml
```

> **Note**
>
> Dibutuhkan sekitar 30 menit untuk membuat infrastruktur VPC Amazon.

Opsi tiga: Membuat jaringan VPC Amazon tanpa akses internet

CloudFormation Template berikut membuat jaringan VPC Amazon tanpa akses internet di default Anda. Wilayah AWS

Opsi ini menggunakanPerutean pribadi tanpa akses internet. Template ini dapat digunakan untuk webserver Apache Airflow dengan mode akses jaringan Private saja. Ini menciptakan titik akhir VPC yang diperlukan untuk AWS layanan yang digunakan oleh lingkungan.

1. Salin isi template berikut dan simpan secara lokal sebagai`cfn-vpc-private.yaml`. Anda juga dapat mengunduh template.

```
AWSTemplateFormatVersion: "2010-09-09"

Parameters:
   VpcCIDR:
     Description: The IP range (CIDR notation) for this VPC
     Type: String
     Default: 10.192.0.0/16

   PrivateSubnet1CIDR:
     Description: The IP range (CIDR notation) for the private subnet in the first
 Availability Zone
     Type: String
     Default: 10.192.10.0/24

   PrivateSubnet2CIDR:
```

```
    Description: The IP range (CIDR notation) for the private subnet in the second
 Availability Zone
    Type: String
    Default: 10.192.11.0/24

Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcCIDR
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
       - Key: Name
         Value: !Ref AWS::StackName

  RouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
       - Key: Name
         Value: !Sub "${AWS::StackName}-route-table"

  PrivateSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs  '' ]
      CidrBlock: !Ref PrivateSubnet1CIDR
      MapPublicIpOnLaunch: false
      Tags:
       - Key: Name
         Value: !Sub "${AWS::StackName} Private Subnet (AZ1)"

  PrivateSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs  '' ]
      CidrBlock: !Ref PrivateSubnet2CIDR
      MapPublicIpOnLaunch: false
      Tags:
       - Key: Name
```

```
          Value: !Sub "${AWS::StackName} Private Subnet (AZ2)"

  PrivateSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref RouteTable
      SubnetId: !Ref PrivateSubnet1

  PrivateSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref RouteTable
      SubnetId: !Ref PrivateSubnet2

  S3VpcEndoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      ServiceName: !Sub "com.amazonaws.${AWS::Region}.s3"
      VpcEndpointType: Gateway
      VpcId: !Ref VPC
      RouteTableIds:
       - !Ref RouteTable

  SecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      VpcId: !Ref VPC
      GroupDescription: Security Group for Amazon MWAA Environments to access VPC
endpoints
      GroupName: !Sub "${AWS::StackName}-mwaa-vpc-endpoints"

  SecurityGroupIngress:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !Ref SecurityGroup
      IpProtocol: "-1"
      SourceSecurityGroupId: !Ref SecurityGroup

  SqsVpcEndoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      ServiceName: !Sub "com.amazonaws.${AWS::Region}.sqs"
      VpcEndpointType: Interface
      VpcId: !Ref VPC
```

```
      PrivateDnsEnabled: true
      SubnetIds:
       - !Ref PrivateSubnet1
       - !Ref PrivateSubnet2
      SecurityGroupIds:
       - !Ref SecurityGroup

  CloudWatchLogsVpcEndoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      ServiceName: !Sub "com.amazonaws.${AWS::Region}.logs"
      VpcEndpointType: Interface
      VpcId: !Ref VPC
      PrivateDnsEnabled: true
      SubnetIds:
       - !Ref PrivateSubnet1
       - !Ref PrivateSubnet2
      SecurityGroupIds:
       - !Ref SecurityGroup

  CloudWatchMonitoringVpcEndoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      ServiceName: !Sub "com.amazonaws.${AWS::Region}.monitoring"
      VpcEndpointType: Interface
      VpcId: !Ref VPC
      PrivateDnsEnabled: true
      SubnetIds:
       - !Ref PrivateSubnet1
       - !Ref PrivateSubnet2
      SecurityGroupIds:
       - !Ref SecurityGroup

  KmsVpcEndoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      ServiceName: !Sub "com.amazonaws.${AWS::Region}.kms"
      VpcEndpointType: Interface
      VpcId: !Ref VPC
      PrivateDnsEnabled: true
      SubnetIds:
       - !Ref PrivateSubnet1
       - !Ref PrivateSubnet2
      SecurityGroupIds:
```

```
        - !Ref SecurityGroup


Outputs:
   VPC:
     Description: A reference to the created VPC
     Value: !Ref VPC

   MwaaSecurityGroupId:
     Description: Associates the Security Group to the environment to allow access
 to the VPC endpoints
     Value: !Ref SecurityGroup

   PrivateSubnets:
     Description: A list of the private subnets
     Value: !Join [ ",", [ !Ref PrivateSubnet1, !Ref PrivateSubnet2 ]]

   PrivateSubnet1:
     Description: A reference to the private subnet in the 1st Availability Zone
     Value: !Ref PrivateSubnet1

   PrivateSubnet2:
     Description: A reference to the private subnet in the 2nd Availability Zone
     Value: !Ref PrivateSubnet2
```

2. Di prompt perintah Anda, arahkan ke direktori tempat `cfn-vpc-private.yml` disimpan. Contoh:

```
cd mwaaproject
```

3. Gunakan `aws cloudformation create-stack`perintah untuk membuat tumpukan menggunakan AWS CLI.

```
aws cloudformation create-stack --stack-name mwaa-private-environment --template-
body file://cfn-vpc-private.yml
```

> **Note**
>
> Dibutuhkan sekitar 30 menit untuk membuat infrastruktur VPC Amazon.

4. Anda harus membuat mekanisme untuk mengakses titik akhir VPC ini dari komputer Anda. Untuk mempelajari lebih lanjut, lihatMengelola akses ke titik akhir VPC Amazon khusus layanan di Amazon MWAA.

> **Note**
>
> Anda selanjutnya dapat membatasi akses keluar di CIDR grup keamanan Amazon MWAA Anda. Misalnya, Anda dapat membatasi dirinya sendiri dengan menambahkan aturan keluar referensi sendiri, daftar awalan untuk Amazon S3, dan CIDR VPC Amazon Anda.

Apa selanjutnya?

- Pelajari cara membuat lingkungan Amazon MWAA di. Buat lingkungan Amazon MWAA
- Pelajari cara membuat terowongan VPN dari komputer Anda ke VPC Amazon Anda dengan perutean pribadi. Tutorial: Mengkonfigurasi akses jaringan pribadi menggunakan AWS Client VPN

Buat lingkungan Amazon MWAA

Alur Kerja Terkelola Amazon untuk Apache Airflow menyiapkan Apache Airflow di lingkungan dalam versi pilihan Anda menggunakan Apache Airflow sumber terbuka yang sama dan antarmuka pengguna yang tersedia dari Apache. Panduan ini menjelaskan langkah-langkah untuk membuat lingkungan Amazon MWAA.

Daftar Isi

- Sebelum Anda mulai
- Versi Apache Airflow
- Buat lingkungan
  - Langkah satu: Tentukan detail
  - Langkah kedua: Konfigurasikan pengaturan lanjutan
  - Langkah ketiga: Tinjau dan buat

Sebelum Anda mulai

- Jaringan VPC yang Anda tentukan untuk lingkungan Anda tidak dapat dimodifikasi setelah lingkungan dibuat.
- Anda memerlukan bucket Amazon S3 yang dikonfigurasi untuk Memblokir semua akses publik, dengan Versi Bucket diaktifkan.
- Anda memerlukan izin Akun AWS dengan untuk menggunakan Amazon MWAA, dan izin di AWS Identity and Access Management (IAM) untuk membuat peran IAM. Jika Anda memilih mode akses jaringan Pribadi untuk server web Apache Airflow, yang membatasi akses Apache Airflow dalam VPC Amazon Anda, Anda memerlukan izin di IAM untuk membuat titik akhir Amazon VPC.

> **Note**
>
> Amazon MWAA secara dinamis menentukan jaringan selama pembuatan. Jika Anda menggunakan subnet IPv6, Amazon MWAA membuat konektivitas tautan pribadi IPv6 ke database dan server web. Amazon MWAA tidak mendukung transisi antar jenis jaringan dan tidak dapat meningkatkan lingkungan yang ada ke IPv6.

Versi Apache Airflow

Versi Apache Airflow berikut didukung di Alur Kerja Terkelola Amazon untuk Apache Airflow.

> **Note**
>
> - Efektif 30 Desember 2025, Amazon MWAA akan mengakhiri dukungan untuk Apache Airflow versi v2.4.3, v2.5.1, dan v2.6.3. Untuk informasi lebih lanjut, lihatDukungan dan FAQ versi Apache Airflow.
> - Dimulai dengan Apache Airflow v2.2.2, Amazon MWAA mendukung penginstalan persyaratan Python, paket penyedia, dan plugin khusus langsung di server web Apache Airflow.
> - Dimulai dengan Apache Airflow v2.7.2, file persyaratan Anda harus menyertakan pernyataan. `--constraint` Jika Anda tidak memberikan batasan, Amazon MWAA akan menentukan satu untuk Anda untuk memastikan paket yang tercantum dalam persyaratan Anda kompatibel dengan versi Apache Airflow yang Anda gunakan.

Untuk informasi selengkapnya tentang pengaturan batasan dalam file persyaratan Anda, lihat Menginstal dependensi Python.

Versi Apache Airflow	Tanggal rilis Apache Airflow	Tanggal ketersediaan Amazon MWAA	Kendala Apache Airflow	Versi Python
v3.2.1	April 22, 2026	19 Mei 2026	v3.2.1 kendala file	Python 3.12
v2.11.0	Mei 20, 2025	Januari 7, 2026	v2.11.0 kendala file	Python 3.12
v3.0.6	Agustus 29, 2025	Oktober 1, 2025	v3.0.6 kendala file	Python 3.12
v2.10.3	November 4, 2024	Desember 18, 2024	v2.10.3 kendala file	Python 3.11
v2.10.1	September 5, 2024	September 26, 2024	v2.10.1 kendala file	Python 3.11
v2.9.2	Juni 10, 2024	9 Juli 2024	v2.9.2 kendala file	Python 3.11
v2.8.1	Januari 19, 2024	Februari 23, 2024	v2.8.1 kendala file	Python 3.11
v2.7.2	Oktober 12, 2023	6 November 2023	v2.7.2 kendala file	Python 3.11

Untuk informasi selengkapnya tentang memigrasi penerapan Apache Airflow yang dikelola sendiri, atau memigrasi lingkungan Amazon MWAA yang ada, termasuk petunjuk untuk membuat cadangan basis data metadata Anda, lihat Panduan Migrasi Amazon MWAA.

Buat lingkungan

Bagian berikut menjelaskan langkah-langkah untuk membuat lingkungan Amazon MWAA.

Langkah satu: Tentukan detail

Untuk menentukan detail untuk lingkungan

1. Buka konsol Amazon MWAA.
2. Pilih Anda Wilayah AWS.
3. Pilih Buat lingkungan.
4. Pada halaman Tentukan detail, di bawah Detail lingkungan:
   a. Masukkan nama unik untuk lingkungan Anda di Nama.
   b. Pilih versi Apache Airflow di bagian Versi Airflow.

   > **Note**
   >
   > Jika tidak ada nilai yang ditentukan, default ke versi Apache Airflow terbaru. Versi terbaru yang tersedia adalah Apache Airflow v3.0.6.

5. Di bawah kode DAG di Amazon S3 tentukan yang berikut:
   a. Ember S3. Pilih Jelajahi S3 dan pilih bucket Amazon S3 Anda, atau masukkan URI Amazon S3.
   b. Folder DAGs. Pilih Jelajahi S3 dan pilih `dags` folder di bucket Amazon S3 Anda, atau masukkan URI Amazon S3.
   c. File plugin - opsional. Pilih Jelajahi S3 dan pilih `plugins.zip` file di bucket Amazon S3 Anda, atau masukkan URI Amazon S3.
   d. File persyaratan - opsional. Pilih Jelajahi S3 dan pilih `requirements.txt` file di bucket Amazon S3 Anda, atau masukkan URI Amazon S3.
   e. File skrip startup - opsional, Pilih Browse S3 dan pilih file skrip di bucket Amazon S3 Anda, atau masukkan URI Amazon S3.
6. Pilih Berikutnya.

Langkah kedua: Konfigurasikan pengaturan lanjutan

Untuk mengonfigurasi pengaturan lanjutan

1. Pada halaman Konfigurasi pengaturan lanjutan, di bawah Jaringan:

   - Pilih Amazon VPC Anda.

     Langkah ini mengisi dua subnet pribadi di VPC Amazon Anda.

2. Di bawah akses server web, pilih mode akses Apache Airflow pilihan Anda:

   a. Jaringan pribadi. Ini membatasi akses UI Apache Airflow ke pengguna dalam VPC Amazon Anda yang telah diberikan akses ke kebijakan IAM untuk lingkungan Anda. Anda memerlukan izin untuk membuat titik akhir Amazon VPC untuk langkah ini.

   > **Note**
   >
   > Pilih opsi Jaringan pribadi jika UI Apache Airflow Anda hanya diakses dalam jaringan perusahaan, dan Anda tidak memerlukan akses ke repositori publik untuk instalasi persyaratan server web. Jika Anda memilih opsi mode akses ini, Anda perlu membuat mekanisme untuk mengakses server web Apache Airflow Anda di Amazon VPC Anda. Untuk informasi lebih lanjut, lihatMengakses titik akhir VPC untuk server web Apache Airflow Anda (akses jaringan pribadi).

   b. Jaringan publik. Hal ini memungkinkan Apache Airflow UI untuk diakses melalui internet oleh pengguna yang diberikan akses ke kebijakan IAM untuk lingkungan Anda.

   c. Baik akses jaringan publik maupun pribadi. Tersedia untuk Apache Airflow versi 3.2.1 dan yang lebih baru. Hal ini memungkinkan Apache Airflow UI untuk diakses melalui internet, sementara pekerja berkomunikasi dengan server web melalui titik akhir VPC pribadi. Pilih opsi ini jika VPC Amazon yang menghosting lingkungan Anda tidak memiliki akses internet.

3. Di bawah Grup keamanan, pilih grup keamanan yang digunakan untuk mengamankan VPC Amazon Anda:

   a. Secara default, Amazon MWAA membuat grup keamanan di VPC Amazon Anda dengan aturan masuk dan keluar tertentu di Buat grup keamanan baru.

   b. Opsional. Hapus centang kotak di Buat grup keamanan baru untuk memilih hingga 5 grup keamanan.

> **Note**
>
> Grup keamanan Amazon VPC yang ada harus dikonfigurasi dengan aturan masuk dan keluar tertentu untuk memungkinkan lalu lintas jaringan. Untuk mempelajari lebih lanjut, lihatKeamanan di VPC Anda di Amazon MWAA.

4. Di bawah kelas Lingkungan, pilih kelas lingkungan.

   Sebaiknya pilih ukuran terkecil yang diperlukan untuk mendukung beban kerja Anda. Anda dapat mengubah kelas lingkungan kapan saja.

5. Untuk jumlah pekerja Maksimum, tentukan jumlah maksimum pekerja Apache Airflow yang akan dijalankan di lingkungan.

   Untuk informasi lebih lanjut, lihatContoh kasus penggunaan kinerja tinggi.

6. Tentukan jumlah server web maksimum dan Jumlah server web minimum untuk mengonfigurasi cara Amazon MWAA menskalakan server web Apache Airflow di lingkungan Anda.

   Untuk informasi lebih lanjut tentang penskalaan otomatis server web, lihat. the section called “Mengkonfigurasi penskalaan otomatis server web”

7. Di bawah Enkripsi, pilih opsi enkripsi data:

   a. Secara default, Amazon MWAA menggunakan kunci AWS milik untuk mengenkripsi data Anda.

   b. Opsional. Pilih Sesuaikan pengaturan enkripsi (lanjutan) untuk memilih AWS KMS kunci yang berbeda. Jika Anda memilih untuk menentukan Customer-managed kunci dalam langkah ini, Anda harus menentukan ID AWS KMS kunci atau ARN. AWS KMS alias dan kunci multi-wilayah tidak didukung oleh Amazon MWAA. Jika Anda menentukan kunci Amazon S3 untuk enkripsi sisi server di bucket Amazon S3, Anda harus menentukan kunci yang sama untuk lingkungan Amazon MWAA Anda.

   > **Note**
   >
   > Anda harus memiliki izin untuk kunci untuk memilihnya di konsol Amazon MWAA. Anda juga harus memberikan izin kepada Amazon MWAA untuk menggunakan kunci dengan melampirkan kebijakan yang dijelaskan di. Lampirkan kebijakan kunci

8. Direkomendasikan. Di bawah Monitoring, pilih satu atau beberapa kategori log untuk konfigurasi pencatatan Aliran Udara untuk mengirim log Apache Airflow ke Log: CloudWatch

    a. Log tugas aliran udara. Pilih jenis log tugas Apache Airflow untuk dikirim CloudWatch ke tingkat Log di Log.

    b. Log server web aliran udara. Pilih jenis log server web Apache Airflow untuk dikirim CloudWatch ke tingkat Log di Log.

    c. Log penjadwal aliran udara. Pilih jenis log penjadwal Apache Airflow untuk dikirim CloudWatch ke tingkat Log di Log.

    d. Log pekerja aliran udara. Pilih jenis log pekerja Apache Airflow untuk dikirim CloudWatch ke tingkat Log di Log.

    e. Log pemrosesan DAG aliran udara. Pilih jenis log pemrosesan Apache Airflow DAG untuk dikirim CloudWatch ke tingkat Log di Log.

9. Opsional. Untuk opsi konfigurasi Aliran Udara, pilih opsi Tambahkan konfigurasi khusus.

    Anda dapat memilih dari daftar dropdown yang disarankan dari opsi konfigurasi Apache Airflow untuk versi Apache Airflow Anda, atau menentukan opsi konfigurasi khusus. Misalnya, `core.default_task_retries` : `3`.

10. Opsional. Di bawah Tag, pilih Tambahkan tag baru untuk mengaitkan tag ke lingkungan Anda. Misalnya, `Environment` : `Staging`.

11. Di bawah Izin, pilih peran eksekusi:

    a. Secara default, Amazon MWAA membuat peran eksekusi di Buat peran baru. Anda harus memiliki izin untuk membuat peran IAM untuk menggunakan opsi ini.

    b. Opsional. Pilih Masukkan peran ARN untuk memasukkan Nama Sumber Daya Amazon (ARN) dari peran eksekusi yang ada.

12. Pilih Berikutnya.

Langkah ketiga: Tinjau dan buat

Untuk meninjau ringkasan lingkungan

- Tinjau ringkasan lingkungan, pilih Buat lingkungan.

> **Note**
>
> Dibutuhkan sekitar dua puluh hingga tiga puluh menit untuk menciptakan lingkungan.

Apa selanjutnya?

- Pelajari cara membuat bucket Amazon S3. Buat bucket Amazon S3 untuk Amazon MWAA

Mengelola akses ke lingkungan Amazon MWAA

Alur Kerja Terkelola Amazon untuk Apache Airflow harus diizinkan untuk menggunakan layanan dan sumber daya AWS lain yang digunakan oleh lingkungan. Anda juga perlu diberikan izin untuk mengakses lingkungan Amazon MWAA dan UI Apache Airflow Anda di (IAM). AWS Identity and Access Management Bagian ini menjelaskan peran eksekusi yang digunakan untuk memberikan akses ke AWS sumber daya untuk lingkungan Anda dan cara menambahkan izin, serta izin yang Anda perlukan untuk mengakses lingkungan Amazon MWAA dan UI Apache Airflow. Akun AWS

Topik

- Mengakses lingkungan Amazon MWAA
- Service-linked peran untuk Amazon MWAA
- Peran eksekusi Amazon MWAA
- Cross-service pencegahan wakil bingung
- Mode akses Apache Airflow

Mengakses lingkungan Amazon MWAA

Untuk menggunakan Alur Kerja Terkelola Amazon untuk Apache Airflow, Anda harus menggunakan akun dan entitas IAM dengan izin yang diperlukan. Topik ini menjelaskan kebijakan akses yang dapat Anda lampirkan ke tim pengembangan Apache Airflow dan pengguna Apache Airflow untuk lingkungan Alur Kerja Terkelola Amazon untuk lingkungan Apache Airflow.

Sebaiknya gunakan kredensi sementara dan konfigurasi identitas gabungan dengan grup dan peran untuk mengakses sumber daya Amazon MWAA Anda. Sebagai praktik terbaik, hindari melampirkan kebijakan langsung ke pengguna IAM Anda. Sebaliknya, tentukan grup atau peran untuk menyediakan akses sementara ke AWS sumber daya.

Sebuah peran IAM adalah identitas IAM yang dapat Anda buat di akun yang memiliki izin tertentu. Peran IAM mirip dengan pengguna IAM karena merupakan AWS identitas dengan kebijakan izin yang menentukan apa yang dapat dan tidak dapat dilakukan identitas. AWS Namun, alih-alih secara unik terkait dengan satu orang, peran dimaksudkan untuk menjadi dapat diambil oleh siapa pun yang membutuhkannya. Selain itu, peran tidak memiliki kredensial jangka panjang standar seperti kata sandi atau kunci akses yang terkait dengannya. Sebagai gantinya, saat Anda mengambil peran, peran tersebut akan memberikan kredensial keamanan sementara untuk sesi peran.

Untuk menetapkan izin ke identitas federasi, Anda membuat peran dan menentukan izin untuk peran tersebut. Ketika identitas terfederasi mengautentikasi, identitas tersebut terhubung dengan peran dan diberi izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat Buat peran untuk penyedia identitas pihak ketiga dalam Panduan Pengguna IAM. Jika menggunakan Pusat Identitas IAM, Anda harus mengonfigurasi set izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM akan mengorelasikan set izin ke peran dalam IAM. Untuk informasi tentang set izin, lihat Set izin dalam Panduan Pengguna AWS IAM Identity Center .

Anda dapat menggunakan peran IAM di akun Anda untuk memberikan Akun AWS izin lain untuk mengakses sumber daya akun Anda. Sebagai contoh, lihat Tutorial IAM: Mendelegasikan akses Akun AWS menggunakan peran IAM dalam Panduan Pengguna IAM.

Bagian-bagian

- Cara kerjanya
- Kebijakan akses konsol penuh: AmazonMWAAFullConsoleAccess
- Kebijakan akses API dan konsol lengkap: AmazonMWAAFullApiAccess
- Read-only kebijakan akses konsol: AmazonMWAAReadOnlyAccess
- Kebijakan akses Apache Airflow UI: AmazonMWAAWebServerAccess
- Kebijakan akses API Apache Airflow Rest: AmazonMWAARestAPIAccess
- Kebijakan CLI Apache Airflow: AmazonMWAAAirflowCliAccess
- Membuat kebijakan JSON
- Contoh kasus penggunaan untuk melampirkan kebijakan ke grup pengembang
- Apa selanjutnya?

Cara kerjanya

Sumber daya dan layanan yang digunakan dalam lingkungan Amazon MWAA tidak dapat diakses oleh semua entitas AWS Identity and Access Management (IAM). Anda harus membuat kebijakan yang memberikan izin kepada pengguna Apache Airflow untuk mengakses sumber daya ini. Misalnya, Anda perlu memberikan akses ke tim pengembangan Apache Airflow Anda.

Amazon MWAA menggunakan kebijakan ini untuk memvalidasi apakah pengguna memiliki izin yang diperlukan untuk melakukan tindakan di AWS konsol atau melalui API yang digunakan oleh lingkungan.

Anda dapat menggunakan kebijakan JSON dalam topik ini untuk membuat kebijakan bagi pengguna Apache Airflow Anda di IAM, lalu melampirkan kebijakan tersebut ke pengguna, grup, atau peran di IAM.

- AmazonMWAAFullConsoleAccess— Gunakan kebijakan ini untuk memberikan izin mengonfigurasi lingkungan di konsol Amazon MWAA.
- AmazonMWAAFullApiAccess— Gunakan kebijakan ini untuk memberikan akses ke semua API Amazon MWAA yang digunakan untuk mengelola lingkungan.
- AmazonMWAAReadOnlyAccess— Gunakan kebijakan ini untuk memberikan akses ke sumber daya yang digunakan oleh lingkungan di konsol Amazon MWAA.
- AmazonMWAAWebServerAccessGunakan kebijakan ini untuk memberikan akses ke server web Apache Airflow.
- AmazonMWAAAirflowCliAccess— Gunakan kebijakan ini untuk memberikan akses untuk menjalankan perintah Apache Airflow CLI.

Untuk memberikan akses dan menambahkan izin bagi pengguna, grup, atau peran Anda:

- Pengguna dan grup di AWS IAM Identity Center:

  Buat rangkaian izin. Ikuti instruksi di Buat rangkaian izin dalam Panduan Pengguna AWS IAM Identity Center .
- Pengguna yang dikelola di IAM melalui penyedia identitas:

  Buat peran untuk federasi identitas. Ikuti instruksi dalam Buat peran untuk penyedia identitas pihak ketiga (federasi) dalam Panduan Pengguna IAM.
- Pengguna IAM:
  - Buat peran yang dapat diambil pengguna Anda. Ikuti instruksi dalam Buat peran untuk pengguna IAM dalam Panduan Pengguna IAM.
  - (Tidak disarankan) Lampirkan kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti petunjuk dalam Menambahkan izin ke pengguna (konsol) dalam Panduan Pengguna IAM.

Kebijakan akses konsol penuh: AmazonMWAAFullConsoleAccess

Pengguna mungkin memerlukan akses ke kebijakan `AmazonMWAAFullConsoleAccess` izin jika mereka perlu mengonfigurasi lingkungan di konsol Amazon MWAA.

> **Note**
>
> Kebijakan akses konsol lengkap Anda harus menyertakan izin untuk melakukan`iam:PassRole`. Hal ini memungkinkan pengguna untuk meneruskan peran terkait layanan, dan peran eksekusi, ke Amazon MWAA. Amazon MWAA mengasumsikan setiap peran untuk memanggil AWS layanan lain atas nama Anda. Contoh berikut menggunakan kunci `iam:PassedToService` kondisi untuk menentukan Amazon MWAA service principal (`airflow.amazonaws.com`) sebagai layanan yang dapat diteruskan peran.
> Untuk informasi selengkapnya`iam:PassRole`, lihat Memberikan izin pengguna untuk meneruskan peran ke AWS layanan di Panduan Pengguna IAM.

Gunakan kebijakan berikut jika Anda ingin membuat, dan mengelola, lingkungan Amazon MWAA Anda menggunakan enkripsi Kunci milik AWSfor at-rest.

Menggunakan sebuah Kunci milik AWS

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "airflow:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "airflow.amazonaws.com"
                }
            }
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "iam:ListRoles"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreatePolicy"
            ],
            "Resource": "arn:aws:iam::111122223333:policy/service-role/MWAA-
Execution-Policy*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:CreateRole"
            ],
            "Resource": "arn:aws:iam::111122223333:role/service-role/AmazonMWAA*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/
airflow.amazonaws.com/AWSServiceRoleForAmazonMWAA"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:ListBucketVersions"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
```

```
                "s3:PutObject",
                "s3:GetEncryptionConfiguration"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:DescribeRouteTables"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateSecurityGroup"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group/airflow-security-group-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:ListAliases"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateVpcEndpoint",
            "Resource": [
                "arn:aws:ec2:*:*:vpc-endpoint/*",
                "arn:aws:ec2:*:*:vpc/*",
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:security-group/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface"
```

```
            ],
            "Resource": [
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

Gunakan kebijakan berikut jika Anda ingin membuat, dan mengelola, lingkungan Amazon MWAA Anda menggunakan kunci terkelola pelanggan untuk enkripsi saat istirahat. Untuk menggunakan kunci yang dikelola pelanggan, prinsipal IAM harus memiliki izin untuk mengakses AWS KMS sumber daya menggunakan kunci yang disimpan di akun Anda.

Menggunakan kunci yang dikelola pelanggan

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "airflow:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "airflow.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "iam:ListRoles"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreatePolicy"
            ],
            "Resource": "arn:aws:iam::111122223333:policy/service-role/MWAA-
Execution-Policy*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:CreateRole"
            ],
            "Resource": "arn:aws:iam::111122223333:role/service-role/AmazonMWAA*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/
airflow.amazonaws.com/AWSServiceRoleForAmazonMWAA"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:ListBucketVersions"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:PutObject",
                "s3:GetEncryptionConfiguration"
```

```
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:DescribeRouteTables"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateSecurityGroup"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group/airflow-security-group-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:ListAliases"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:DescribeKey",
                "kms:ListGrants",
                "kms:CreateGrant",
                "kms:RevokeGrant",
                "kms:Decrypt",
                "kms:Encrypt",
                "kms:GenerateDataKey*",
                "kms:ReEncrypt*"
            ],
            "Resource": "arn:aws:kms:*:111122223333:key/YOUR_KMS_ID"
        },
        {
            "Effect": "Allow",
```

```
            "Action": "ec2:CreateVpcEndpoint",
            "Resource": [
                "arn:aws:ec2:*:*:vpc-endpoint/*",
                "arn:aws:ec2:*:*:vpc/*",
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:security-group/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

Kebijakan akses API dan konsol lengkap: AmazonMWAAFullApiAccess

Pengguna mungkin memerlukan akses ke kebijakan `AmazonMWAAFullApiAccess` izin jika mereka memerlukan akses ke semua API Amazon MWAA yang digunakan untuk mengelola lingkungan. Itu tidak memberikan izin untuk mengakses Apache Airflow UI.

> **Note**
>
> Kebijakan akses API lengkap harus menyertakan izin untuk melakukan`iam:PassRole`. Hal ini memungkinkan pengguna untuk meneruskan peran terkait layanan, dan peran eksekusi, ke Amazon MWAA. Amazon MWAA mengasumsikan setiap peran untuk memanggil AWS layanan lain atas nama Anda. Contoh berikut menggunakan kunci `iam:PassedToService` kondisi untuk menentukan Amazon MWAA service principal (`airflow.amazonaws.com`) sebagai layanan yang dapat diteruskan peran.
> Untuk informasi selengkapnya`iam:PassRole`, lihat Memberikan izin pengguna untuk meneruskan peran ke AWS layanan di Panduan Pengguna IAM.

Gunakan kebijakan berikut jika Anda ingin membuat, dan mengelola, lingkungan Amazon MWAA Anda menggunakan enkripsi Kunci milik AWS for at-rest.

Menggunakan sebuah Kunci milik AWS

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"airflow:*",
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:PassRole"
         ],
         "Resource":"*",
         "Condition":{
            "StringLike":{
               "iam:PassedToService":"airflow.amazonaws.com"
            }
         }
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:CreateServiceLinkedRole"
         ],
         "Resource":"arn:aws:iam::*:role/aws-service-role/airflow.amazonaws.com/
AWSServiceRoleForAmazonMWAA"
      },
      {
         "Effect":"Allow",
         "Action":[
            "ec2:DescribeSecurityGroups",
            "ec2:DescribeSubnets",
            "ec2:DescribeVpcs",
            "ec2:DescribeRouteTables"
         ],
```

```
            "Resource":"*"
         },
         {
            "Effect":"Allow",
            "Action":[
               "s3:GetEncryptionConfiguration"
            ],
            "Resource":"arn:aws:s3:::*"
         },
         {
            "Effect":"Allow",
            "Action":"ec2:CreateVpcEndpoint",
            "Resource":[
               "arn:aws:ec2:*:*:vpc-endpoint/*",
               "arn:aws:ec2:*:*:vpc/*",
               "arn:aws:ec2:*:*:subnet/*",
               "arn:aws:ec2:*:*:security-group/*"
            ]
         },
         {
            "Effect":"Allow",
            "Action":[
               "ec2:CreateNetworkInterface"
            ],
            "Resource":[
               "arn:aws:ec2:*:*:subnet/*",
               "arn:aws:ec2:*:*:network-interface/*"
            ]
         }
      ]
}
```

Gunakan kebijakan berikut jika Anda ingin membuat, dan mengelola, lingkungan Amazon MWAA Anda menggunakan kunci terkelola pelanggan untuk enkripsi saat istirahat. Untuk menggunakan kunci yang dikelola pelanggan, prinsipal IAM harus memiliki izin untuk mengakses AWS KMS sumber daya menggunakan kunci yang disimpan di akun Anda.

Menggunakan kunci yang dikelola pelanggan

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "airflow:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "airflow.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/
airflow.amazonaws.com/AWSServiceRoleForAmazonMWAA"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:DescribeRouteTables"
            ],
            "Resource": "*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "kms:DescribeKey",
                "kms:ListGrants",
                "kms:CreateGrant",
                "kms:RevokeGrant",
                "kms:Decrypt",
                "kms:Encrypt",
                "kms:GenerateDataKey*",
                "kms:ReEncrypt*"
            ],
            "Resource": "arn:aws:kms:*:111122223333:key/YOUR_KMS_ID"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetEncryptionConfiguration"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateVpcEndpoint",
            "Resource": [
                "arn:aws:ec2:*:*:vpc-endpoint/*",
                "arn:aws:ec2:*:*:vpc/*",
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:security-group/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

Read-only kebijakan akses konsol: AmazonMWAAReadOnlyAccess

Pengguna mungkin memerlukan akses ke kebijakan `AmazonMWAAReadOnlyAccess` izin jika mereka perlu mengakses sumber daya yang digunakan oleh lingkungan di halaman detail lingkungan konsol Amazon MWAA. Itu tidak memungkinkan pengguna untuk membuat lingkungan baru, mengedit lingkungan yang ada, atau mengizinkan pengguna untuk mengakses UI Apache Airflow.

JSON

```
{
        "Version":"2012-10-17",
        "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "airflow:ListEnvironments",
                "airflow:GetEnvironment",
                "airflow:ListTagsForResource"
            ],
            "Resource": "*"
        }
    ]
}
```

Kebijakan akses Apache Airflow UI: AmazonMWAAWebServerAccess

Pengguna mungkin memerlukan akses ke kebijakan `AmazonMWAAWebServerAccess` izin jika mereka perlu mengakses UI Apache Airflow. Itu tidak memungkinkan pengguna untuk mengakses lingkungan di konsol Amazon MWAA atau menggunakan Amazon MWAA API untuk melakukan tindakan apa pun. Tentukan `Admin`Op,`User`,, `Viewer` atau `Public` peran dalam `{airflow-role}` untuk menyesuaikan tingkat akses bagi pengguna token web. Untuk informasi selengkapnya, lihat Peran Default dalam panduan referensi Apache Airflow.

JSON

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

```
        {
            "Effect": "Allow",
            "Action": "airflow:CreateWebLoginToken",
            "Resource": [
             "arn:aws:airflow:us-east-1:111122223333:role/{your-environment-name}/{airflow-role}"
            ]
        }
    ]
}
```

Note

- Amazon MWAA menyediakan integrasi IAM dengan lima peran kontrol akses berbasis peran Apache Airflow (RBAC) default. Untuk informasi selengkapnya tentang bekerja dengan peran Apache Airflow kustom, lihat. the section called “Tutorial: Membatasi pengguna untuk subset DAGs”
- `Resource`Bidang dalam kebijakan ini dapat digunakan untuk menentukan peran kontrol akses berbasis peran Apache Airflow untuk lingkungan Amazon MWAA. Namun, itu tidak mendukung lingkungan Amazon MWAA ARN (Nama Sumber Daya Amazon) di bidang `Resource` kebijakan.

Kebijakan akses API Apache Airflow Rest: AmazonMWAARestAPIAccess

Untuk mengakses Apache Airflow REST API, Anda harus memberikan `airflow:InvokeRestApi` izin dalam kebijakan IAM Anda. Dalam contoh kebijakan berikut, tentukan `Admin`Op,`User`,, `Viewer` atau `Public` peran `{airflow-role}` untuk menyesuaikan tingkat akses pengguna. Untuk informasi selengkapnya, lihat Peran Default dalam panduan referensi Apache Airflow.

JSON

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

```
            "Sid": "AllowMwaaRestApiAccess",
            "Effect": "Allow",
            "Action": "airflow:InvokeRestApi",
            "Resource": [
            "arn:aws:airflow:us-east-1:111122223333:role/{your-environment-name}/
{airflow-role}"
            ]
        }
    ]
}
```

Note

- Saat mengonfigurasi server web pribadi, `InvokeRestApi` tindakan tidak dapat dipanggil dari luar Virtual Private Cloud (VPC). Anda dapat menggunakan `aws:SourceVpc` kunci untuk menerapkan kontrol akses yang lebih terperinci untuk operasi ini. Untuk informasi lebih lanjut, lihat aws: SourceVpc
- `Resource`Bidang dalam kebijakan ini dapat digunakan untuk menentukan peran kontrol akses berbasis peran Apache Airflow untuk lingkungan Amazon MWAA. Namun, itu tidak mendukung lingkungan Amazon MWAA ARN (Nama Sumber Daya Amazon) di bidang `Resource` kebijakan.

Kebijakan CLI Apache Airflow: AmazonMWAAAirflowCliAccess

Pengguna mungkin memerlukan akses ke kebijakan `AmazonMWAAAirflowCliAccess` izin jika mereka perlu menjalankan perintah Apache Airflow CLI (seperti). `trigger_dag` Itu tidak memungkinkan pengguna untuk mengakses lingkungan di konsol Amazon MWAA atau menggunakan Amazon MWAA API untuk melakukan tindakan apa pun.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "airflow:CreateCliToken"
            ],
            "Resource": "arn:aws:airflow:us-east-1:111122223333:environment/
${EnvironmentName}"
        }
    ]
}
```

Membuat kebijakan JSON

Anda dapat membuat kebijakan JSON, dan melampirkan kebijakan tersebut ke pengguna, peran, atau grup Anda di konsol IAM. Langkah-langkah berikut menjelaskan cara membuat kebijakan JSON di IAM.

Untuk membuat kebijakan JSON

1. Buka halaman Kebijakan di konsol IAM.
2. Pilih Buat kebijakan.
3. Pilih tab JSON.
4. Tambahkan kebijakan JSON Anda.
5. Pilih Tinjau kebijakan.
6. Masukkan nilai di bidang teks untuk Nama dan Deskripsi (opsional).

   Misalnya, Anda dapat memberi nama kebijakan`AmazonMWAAReadOnlyAccess`.
7. Pilih Buat kebijakan.

Contoh kasus penggunaan untuk melampirkan kebijakan ke grup pengembang

Katakanlah Anda menggunakan grup di IAM bernama `AirflowDevelopmentGroup` untuk menerapkan izin ke semua pengembang di tim pengembangan Apache Airflow Anda. Pengguna ini memerlukan akses ke`AmazonMWAAFullConsoleAccess`,`AmazonMWAAAirflowCliAccess`, dan kebijakan `AmazonMWAAWebServerAccess` izin. Bagian ini menjelaskan cara membuat grup di IAM, membuat dan melampirkan kebijakan ini, dan mengaitkan grup ke pengguna IAM. Langkah-langkah mengasumsikan Anda menggunakan kunci yang AWS dimiliki.

Untuk membuat AmazonMWAAFullConsoleAccess kebijakan

1. Unduh kebijakan AmazonMWAAFullConsoleAccess akses.
2. Buka halaman Kebijakan di konsol IAM.
3. Pilih Buat kebijakan.
4. Pilih tab JSON.
5. Tempel kebijakan JSON untuk`AmazonMWAAFullConsoleAccess`.
6. Gantikan nilai-nilai berikut:
   a. *123456789012*— Akun AWS ID Anda (seperti`0123456789`)
   b. *{your-kms-id}*— Pengidentifikasi unik untuk kunci yang dikelola pelanggan, hanya berlaku jika Anda menggunakan kunci yang dikelola pelanggan untuk enkripsi saat istirahat.
7. Pilih kebijakan Tinjauan.
8. `AmazonMWAAFullConsoleAccess`Ketik Nama.
9. Pilih Buat kebijakan.

Untuk membuat AmazonMWAAWebServerAccess kebijakan

1. Unduh kebijakan AmazonMWAAWebServerAccess akses.
2. Buka halaman Kebijakan di konsol IAM.
3. Pilih Buat kebijakan.
4. Pilih tab JSON.
5. Tempel kebijakan JSON untuk`AmazonMWAAWebServerAccess`.
6. Gantikan nilai-nilai berikut:
   a. *us-east-1*— wilayah lingkungan Amazon MWAA Anda (seperti) `us-east-1`
   b. *123456789012*— Akun AWS ID Anda (seperti`0123456789`)
   c. *{your-environment-name}*— nama lingkungan Amazon MWAA Anda (seperti) `MyAirflowEnvironment`
   d. *{airflow-role}*— Peran `Admin` Default Apache Airflow
7. Pilih Tinjau kebijakan.
8. `AmazonMWAAWebServerAccess`Ketik Nama.
9. Pilih Buat kebijakan.

Untuk membuat AmazonMWAAAirflowCliAccess kebijakan

1. Unduh kebijakan AmazonMWAAAirflowCliAccess akses.
2. Buka halaman Kebijakan di konsol IAM.
3. Pilih Buat kebijakan.
4. Pilih tab JSON.
5. Tempel kebijakan JSON untuk`AmazonMWAAAirflowCliAccess`.
6. Pilih kebijakan Tinjauan.
7. `AmazonMWAAAirflowCliAccess`Ketik Nama.
8. Pilih Buat kebijakan.

Untuk membuat grup

1. Buka halaman Grup di konsol IAM.
2. Masukkan nama`AirflowDevelopmentGroup`.
3. Pilih Langkah Selanjutnya.
4. Ketik `AmazonMWAA` untuk memfilter hasil di Filter.
5. Pilih tiga kebijakan yang Anda buat.
6. Pilih Langkah Selanjutnya.
7. Pilih Buat group.

Untuk mengaitkan dengan pengguna

1. Buka halaman Pengguna di konsol IAM.
2. Pilih pengguna.
3. Pilih Grup.
4. Pilih Tambahkan pengguna ke grup.
5. Pilih AirflowDevelopmentGroup.
6. Pilih Tambahkan ke Grup.

Apa selanjutnya?

- Pelajari cara membuat token untuk mengakses UI Apache Airflow. Mengakses Apache Airflow

- Pelajari lebih lanjut cara membuat kebijakan IAM di Membuat kebijakan IAM.

Service-linked peran untuk Amazon MWAA

Alur Kerja Terkelola Amazon untuk Apache Airflow AWS Identity and Access Management menggunakan peran terkait layanan (IAM). Peran terkait layanan adalah jenis peran IAM unik yang ditautkan langsung ke Amazon MWAA. Service-linked peran telah ditentukan sebelumnya oleh Amazon MWAA dan mencakup semua izin yang diperlukan layanan untuk memanggil AWS layanan lain atas nama Anda.

Peran terkait layanan membuat pengaturan Amazon MWAA lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. Amazon MWAA mendefinisikan izin peran terkait layanannya, dan kecuali ditentukan lain, hanya Amazon MWAA yang dapat mengambil perannya. Izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin, dan kebijakan izin tersebut tidak dapat dilampirkan ke entitas IAM lainnya.

Anda dapat menghapus peran tertaut layanan hanya setelah menghapus sumber daya terkait terlebih dahulu. Ini melindungi sumber daya Amazon MWAA Anda karena Anda tidak dapat secara tidak sengaja menghapus izin untuk mengakses sumber daya.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, lihat AWS Layanan yang Bekerja dengan IAM dan cari layanan yang memiliki Ya di kolom peran. Service-linked Pilih Ya dengan tautan untuk mengakses dokumentasi peran terkait layanan untuk layanan tersebut.

Service-linked izin peran untuk Amazon MWAA

Amazon MWAA menggunakan peran terkait layanan bernama `AWSServiceRoleForAmazonMWAA` — Peran terkait layanan yang dibuat di akun Anda memberi Amazon MWAA akses ke layanan berikut: AWS

- Amazon CloudWatch Logs (CloudWatch Log) - Untuk membuat grup log untuk log Apache Airflow.
- Amazon CloudWatch (CloudWatch) — Untuk mempublikasikan metrik yang terkait dengan lingkungan Anda dan komponen dasarnya ke akun Anda.
- Amazon Elastic Compute Cloud (Amazon EC2) — Untuk membuat sumber daya berikut:
  - Titik akhir Amazon VPC di VPC Anda untuk kluster database Amazon Aurora PostgreSQL yang AWS dikelola untuk digunakan oleh penjadwal dan pekerja Apache Airflow.
  - Endpoint Amazon VPC tambahan untuk mengaktifkan akses jaringan ke server web jika Anda memilih opsi jaringan pribadi untuk server web Apache Airflow Anda.

- Antarmuka Jaringan Elastis (ENI) di VPC Amazon Anda untuk mengaktifkan akses jaringan ke AWS sumber daya yang dihosting di VPC Amazon Anda.

Kebijakan kepercayaan berikut memungkinkan kepala layanan untuk mengambil peran terkait layanan. Prinsip layanan untuk Amazon MWAA adalah `airflow.amazonaws.com` seperti yang ditunjukkan oleh kebijakan.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "airflow.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Kebijakan izin peran bernama `AmazonMWAAServiceRolePolicy` memungkinkan Amazon MWAA menyelesaikan tindakan berikut pada sumber daya yang ditentukan:

JSON

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

```
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AttachNetworkInterface",
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeVpcs",
                "ec2:DetachNetworkInterface"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateVpcEndpoint",
            "Resource": "arn:aws:ec2:*:*:vpc-endpoint/*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": "AmazonMWAAManaged"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:ModifyVpcEndpoint",
                "ec2:DeleteVpcEndpoints"
            ],
            "Resource": "arn:aws:ec2:*:*:vpc-endpoint/*",
            "Condition": {
                "Null": {
                    "aws:ResourceTag/AmazonMWAAManaged": false
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "ec2:CreateVpcEndpoint",
                "ec2:ModifyVpcEndpoint"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:vpc/*",
                "arn:aws:ec2:*:*:security-group/*",
                "arn:aws:ec2:*:*:subnet/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*:*:vpc-endpoint/*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": "CreateVpcEndpoint"
                },
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": "AmazonMWAAManaged"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricData",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": [
                        "AWS/MWAA"
                    ]
                }
            }
        }
    ]
}
```

Anda harus mengonfigurasi izin agar entitas IAM (seperti pengguna, grup, atau peran) dapat membuat, mengedit, atau menghapus peran terkait layanan. Untuk informasi selengkapnya, lihat izin Service-linked peran di Panduan Pengguna IAM.

Membuat peran terkait layanan untuk Amazon MWAA

Anda tidak perlu membuat peran terkait layanan secara manual. Saat Anda membuat lingkungan Amazon MWAA baru menggunakan, API Konsol Manajemen AWS, atau AWS API AWS CLI, Amazon MWAA membuat peran terkait layanan untuk Anda.

Jika Anda menghapus peran terkait layanan ini, dan ingin membuatnya lagi, Anda dapat mengulangi proses yang sama untuk membuat kembali peran tersebut di akun Anda. Saat Anda membuat lingkungan lain, Amazon MWAA membuat peran terkait layanan untuk Anda lagi.

Mengedit peran terkait layanan untuk Amazon MWAA

Amazon MWAA tidak mengizinkan pengeditan peran terkait AWSServiceRoleForAmazonMWAA layanan. Setelah membuat peran terkait layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin merujuk peran tersebut. Namun, Anda dapat mengedit penjelasan peran menggunakan IAM. Untuk informasi selengkapnya, lihat Mengedit peran terkait layanan di Panduan Pengguna IAM.

Menghapus peran terkait layanan untuk Amazon MWAA

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran terkait layanan, sebaiknya hapus peran tersebut. Dengan begitu, Anda tidak perlu lagi memantau atau memelihara entitas yang tidak digunakan.

Saat Anda menghapus lingkungan Amazon MWAA, Amazon MWAA menghapus semua sumber daya terkait yang digunakannya sebagai bagian dari layanan. Namun, Anda harus menunggu sebelum Amazon MWAA selesai menghapus lingkungan Anda, sebelum mencoba menghapus peran terkait layanan. Jika Anda menghapus peran terkait layanan sebelum Amazon MWAA menghapus lingkungan, Amazon MWAA mungkin tidak dapat menghapus semua sumber daya terkait lingkungan.

Untuk menghapus peran tertaut layanan secara manual menggunakan IAM

Gunakan konsol IAM, the AWS CLI, atau AWS API untuk menghapus peran AWSServiceRoleForAmazonMWAA terkait layanan. Untuk informasi selengkapnya, lihat Menghapus peran terkait layanan di Panduan Pengguna IAM.

Wilayah yang didukung untuk peran terkait layanan Amazon MWAA

Amazon MWAA mendukung penggunaan peran terkait layanan di semua wilayah tempat layanan tersedia. Untuk informasi selengkapnya, lihat Alur Kerja Terkelola Amazon untuk titik akhir dan kuota Apache Airflow.

Pembaruan kebijakan

Ubah	Deskripsi	Date
Amazon MWAA memperbarui kebijakan izin peran terkait layanannya	`AmazonMWAAServiceRolePolicy` - Amazon MWAA memperbarui kebijakan izin untuk peran terkait layanannya untuk memberikan izin Amazon MWAA untuk mempublikasikan metrik tambahan yang terkait dengan sumber daya dasar layanan ke akun pelanggan. Metrik baru ini diterbitkan di `AWS/MWAA`	18 November 2022
Amazon MWAA mulai melacak perubahan	Amazon MWAA mulai melacak perubahan untuk kebijakan izin peran AWS terkait layanan yang dikelola.	18 November 2022

Peran eksekusi Amazon MWAA

Peran eksekusi adalah peran AWS Identity and Access Management (IAM) dengan kebijakan izin yang memberikan izin kepada Alur Kerja Terkelola Amazon untuk Apache Airflow untuk memanggil sumber daya layanan lain atas nama Anda. AWS Ini dapat mencakup sumber daya seperti bucket Amazon S3, kunci AWS milik, dan Log. CloudWatch Lingkungan Amazon MWAA membutuhkan satu peran eksekusi per lingkungan. Topik ini menjelaskan cara menggunakan dan mengonfigurasi peran

eksekusi untuk lingkungan Anda agar Amazon MWAA dapat mengakses AWS sumber daya lain yang digunakan oleh lingkungan Anda.

Daftar Isi

- Ikhtisar peran eksekusi
  - Izin dilampirkan secara default
  - Cara menambahkan izin untuk menggunakan yang lain AWS layanan
  - Cara mengaitkan peran eksekusi baru
- Buat peran baru
- Mengakses dan memperbarui kebijakan peran eksekusi
  - Lampirkan kebijakan JSON untuk menggunakan yang lain AWS layanan
- Berikan akses ke bucket Amazon S3 dengan blok akses publik tingkat akun
- Gunakan koneksi Apache Airflow
- Contoh kebijakan JSON untuk peran eksekusi
  - Contoh kebijakan untuk kunci yang dikelola pelanggan
  - Contoh kebijakan untuk AWS-kunci yang dimiliki
- Apa selanjutnya?

Ikhtisar peran eksekusi

Izin Amazon MWAA untuk menggunakan AWS layanan lain yang digunakan oleh lingkungan Anda berasal dari peran eksekusi. Peran eksekusi Amazon MWAA memerlukan izin untuk AWS layanan berikut yang digunakan oleh lingkungan:

- Amazon CloudWatch (CloudWatch) — untuk mengirim metrik dan log Apache Airflow.
- Amazon Simple Storage Service (Amazon S3) — untuk mengurai kode DAG lingkungan Anda dan file pendukung (seperti a). `requirements.txt`
- Amazon Simple Queue Service (Amazon SQS) — untuk mengantri tugas Apache Airflow lingkungan Anda dalam antrian Amazon SQS yang dimiliki oleh Amazon MWAA.
- AWS Key Management Service (AWS KMS) — untuk enkripsi data lingkungan Anda (menggunakan kunci yang AWS dimiliki atau Customer-managed kunci Anda).

> **Note**
>
> Jika Anda telah memilih Amazon MWAA untuk menggunakan kunci KMS yang AWS dimiliki untuk mengenkripsi data Anda, maka Anda harus menentukan izin dalam kebijakan yang dilampirkan pada peran eksekusi Amazon MWAA Anda yang memberikan akses ke kunci KMS arbitrer yang disimpan di luar akun Anda melalui Amazon SQS. Dua kondisi berikut diperlukan agar peran eksekusi lingkungan Anda dapat mengakses kunci KMS arbitrer:
>
> - Kunci KMS di akun pihak ketiga perlu mengizinkan akses lintas akun ini melalui kebijakan sumber dayanya.
> - Kode DAG Anda perlu mengakses antrean Amazon SQS yang dimulai dengan `airflow-celery-` di akun pihak ketiga dan menggunakan kunci KMS yang sama untuk enkripsi.
>
> Untuk mengurangi risiko yang terkait dengan akses lintas akun ke sumber daya, sebaiknya tinjau kode yang ditempatkan di DAG Anda untuk memastikan bahwa alur kerja Anda tidak mengakses antrian Amazon SQS sewenang-wenang di luar akun Anda. Selanjutnya, Anda dapat menggunakan kunci KMS yang dikelola pelanggan yang disimpan di akun Anda sendiri untuk mengelola enkripsi di Amazon MWAA. Ini membatasi peran eksekusi lingkungan Anda untuk hanya mengakses kunci KMS di akun Anda.
> Perlu diingat bahwa setelah Anda memilih opsi enkripsi, Anda tidak dapat mengubah pilihan Anda untuk lingkungan yang ada.

Peran eksekusi juga memerlukan izin untuk tindakan IAM berikut:

- `airflow:PublishMetrics`— untuk memungkinkan Amazon MWAA memantau kesehatan suatu lingkungan.

Izin dilampirkan secara default

Anda dapat menggunakan opsi default di konsol Amazon MWAA untuk membuat peran eksekusi dan kunci yang AWS dimiliki, lalu gunakan langkah-langkah di halaman ini untuk menambahkan kebijakan izin ke peran eksekusi Anda.

- Saat Anda memilih opsi Buat peran baru di konsol, Amazon MWAA melampirkan izin minimal yang diperlukan oleh lingkungan ke peran eksekusi Anda.

- Dalam beberapa kasus, Amazon MWAA melampirkan izin maksimum. Misalnya, sebaiknya pilih opsi di konsol Amazon MWAA untuk membuat peran eksekusi saat Anda membuat lingkungan. Amazon MWAA menambahkan kebijakan izin untuk semua grup CloudWatch Log secara otomatis dengan menggunakan pola regex dalam peran eksekusi sebagai. `"arn:aws:logs:us-east-1:111122223333:log-group:airflow-your-environment-name-*"`

Cara menambahkan izin untuk menggunakan yang lain AWS layanan

Amazon MWAA tidak dapat menambahkan atau mengedit kebijakan izin ke peran eksekusi yang ada setelah lingkungan dibuat. Anda harus memperbarui peran eksekusi Anda dengan kebijakan izin tambahan yang diperlukan oleh lingkungan Anda. Misalnya, jika DAG Anda memerlukan akses ke AWS Glue, Amazon MWAA tidak dapat secara otomatis mendeteksi izin ini diperlukan oleh lingkungan Anda, atau menambahkan izin ke peran eksekusi Anda.

Anda dapat menambahkan izin ke peran eksekusi dengan dua cara:

- Dengan memodifikasi kebijakan JSON untuk peran eksekusi Anda secara inline. Anda dapat menggunakan contoh dokumen kebijakan JSON di halaman ini untuk menambah atau mengganti kebijakan JSON dari peran eksekusi Anda di konsol IAM.
- Dengan membuat kebijakan JSON untuk AWS layanan dan melampirkannya ke peran eksekusi Anda. Anda dapat menggunakan langkah-langkah di halaman ini untuk mengaitkan dokumen kebijakan JSON baru untuk AWS layanan ke peran eksekusi Anda di konsol IAM.

Dengan asumsi peran eksekusi sudah terkait dengan lingkungan Anda, Amazon MWAA dapat segera mulai menggunakan kebijakan izin yang ditambahkan. Ini juga berarti jika Anda menghapus izin yang diperlukan dari peran eksekusi, DAG Anda mungkin gagal.

Cara mengaitkan peran eksekusi baru

Anda dapat mengubah peran eksekusi untuk lingkungan Anda kapan saja. Jika peran eksekusi baru belum dikaitkan dengan lingkungan Anda, gunakan langkah-langkah di halaman ini untuk membuat kebijakan peran eksekusi baru, dan kaitkan peran tersebut dengan lingkungan Anda.

Buat peran baru

Secara default, Amazon MWAA membuat kunci AWS milik untuk enkripsi data dan peran eksekusi atas nama Anda. Anda dapat memilih opsi default di konsol Amazon MWAA saat Anda membuat

lingkungan. Gambar berikut menampilkan opsi default untuk membuat peran eksekusi untuk lingkungan.

> **Important**
>
> Saat Anda membuat peran eksekusi baru, jangan gunakan kembali nama peran eksekusi yang dihapus. Nama unik dapat membantu mencegah konflik dan memastikan manajemen sumber daya yang tepat.

Mengakses dan memperbarui kebijakan peran eksekusi

Anda dapat mengakses peran eksekusi untuk lingkungan Anda di konsol Amazon MWAA, dan memperbarui kebijakan JSON untuk peran tersebut di konsol IAM.

Untuk memperbarui kebijakan peran eksekusi

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.
3. Pilih peran eksekusi pada panel Izin untuk membuka halaman izin di IAM.
4. Pilih nama peran eksekusi untuk membuka kebijakan izin.
5. Pilih Sunting kebijakan.
6. Pilih tab JSON.
7. Perbarui kebijakan JSON Anda.
8. Pilih Tinjau kebijakan.
9. Pilih Simpan perubahan.

Lampirkan kebijakan JSON untuk menggunakan yang lain AWS layanan

Anda dapat membuat kebijakan JSON untuk AWS layanan dan melampirkannya ke peran eksekusi Anda. Misalnya, Anda dapat melampirkan kebijakan JSON berikut untuk memberikan akses hanya-baca ke semua sumber daya di. AWS Secrets Manager

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "secretsmanager:GetResourcePolicy",
            "secretsmanager:GetSecretValue",
            "secretsmanager:DescribeSecret",
            "secretsmanager:ListSecretVersionIds"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

Untuk melampirkan kebijakan ke peran eksekusi Anda

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.
3. Pilih peran eksekusi Anda di panel Izin.
4. Pilih Lampirkan kebijakan.
5. Pilih Buat kebijakan.
6. PilihJSON.
7. Tempel kebijakan JSON.
8. Pilih Berikutnya: Tag, Berikutnya: Ulasan.
9. Masukkan nama deskriptif (sepertiSecretsManagerReadPolicy) dan deskripsi untuk kebijakan tersebut.
10. Pilih Buat kebijakan.

Berikan akses ke bucket Amazon S3 dengan blok akses publik tingkat akun

Anda mungkin ingin memblokir akses ke semua bucket di akun Anda dengan menggunakan operasi `PutPublicAccessBlock`Amazon S3. Saat Anda memblokir akses ke semua bucket di akun, peran eksekusi lingkungan Anda harus menyertakan `s3:GetAccountPublicAccessBlock` tindakan tersebut dalam kebijakan izin.

Contoh berikut menunjukkan kebijakan yang harus Anda lampirkan ke peran eksekusi saat memblokir akses ke semua bucket Amazon S3 di akun Anda.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
     {
       "Effect": "Allow",
       "Action": "s3:GetAccountPublicAccessBlock",
       "Resource": "*"
     }
  ]
}
```

Untuk informasi selengkapnya tentang membatasi akses ke bucket Amazon S3 Anda, lihat Memblokir akses publik ke penyimpanan Amazon S3 Anda di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Gunakan koneksi Apache Airflow

Anda juga dapat membuat koneksi Apache Airflow dan menentukan peran eksekusi Anda dan ARN di objek koneksi Apache Airflow Anda. Untuk mempelajari lebih lanjut, lihatMengelola koneksi ke Apache Airflow.

Contoh kebijakan JSON untuk peran eksekusi

Anda dapat menggunakan dua contoh kebijakan izin di bagian ini untuk mengganti kebijakan izin yang digunakan untuk peran eksekusi yang ada, atau untuk membuat peran eksekusi baru dan digunakan untuk lingkungan Anda. Kebijakan ini berisi placeholder ARN Sumber Daya untuk grup log Apache Airflow, bucket Amazon S3, dan lingkungan Amazon MWAA.

Sebaiknya salin kebijakan contoh, ganti contoh ARN atau placeholder, lalu gunakan kebijakan JSON untuk membuat atau memperbarui peran eksekusi.

Contoh kebijakan untuk kunci yang dikelola pelanggan

Contoh berikut menyajikan kebijakan peran eksekusi yang dapat Anda gunakan untuk Customer-managed kunci.

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Deny",
   "Action": "s3:ListAllMyBuckets",
   "Resource": [
     "arn:aws:s3:::amzn-s3-demo-bucket",
    "arn:aws:s3:::amzn-s3-demo-bucket/*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "s3:GetObject*",
    "s3:GetBucket*",
    "s3:List*"
   ],
   "Resource": [
     "arn:aws:s3:::amzn-s3-demo-bucket",
    "arn:aws:s3:::amzn-s3-demo-bucket/*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "logs:CreateLogStream",
    "logs:CreateLogGroup",
    "logs:PutLogEvents",
    "logs:GetLogEvents",
    "logs:GetLogRecord",
    "logs:GetLogGroupFields",
```

```
    "logs:GetQueryResults"
   ],
   "Resource": [
    "arn:aws:logs:us-east-1:111122223333:log-group:airflow-your-environment-
name:*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "logs:DescribeLogGroups"
   ],
   "Resource": [
    "*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "s3:GetAccountPublicAccessBlock"
   ],
   "Resource": [
    "*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": "cloudwatch:PutMetricData",
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "sqs:ChangeMessageVisibility",
    "sqs:DeleteMessage",
    "sqs:GetQueueAttributes",
    "sqs:GetQueueUrl",
    "sqs:ReceiveMessage",
    "sqs:SendMessage"
   ],
   "Resource": "arn:aws:sqs:us-east-1:*:airflow-celery-*"
  },
  {
   "Effect": "Allow",
```

```
    "Action": [
     "kms:Decrypt",
     "kms:DescribeKey",
     "kms:GenerateDataKey*",
     "kms:Encrypt"
    ],
    "Resource": "arn:aws:kms:us-east-1:111122223333:key/your-kms-cmk-id",
    "Condition": {
     "StringLike": {
      "kms:ViaService": [
       "sqs.us-east-1.amazonaws.com",
       "s3.us-east-1.amazonaws.com"
      ]
     }
    }
   }
  ]
}
```

Selanjutnya, Anda perlu mengizinkan Amazon MWAA untuk mengambil peran ini untuk melakukan tindakan atas nama Anda. Hal ini dapat dilakukan dengan menambahkan `"airflow.amazonaws.com"` dan `"airflow-env.amazonaws.com"` melayani prinsipal ke daftar entitas tepercaya untuk peran eksekusi ini menggunakan konsol IAM, atau dengan menempatkan prinsip-prinsip layanan ini dalam dokumen kebijakan peran asumsi untuk peran eksekusi ini melalui perintah IAM create-role menggunakan perintah. AWS CLI Lihat contoh dokumen kebijakan peran asumsi berikut:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
            "Service": ["airflow.amazonaws.com","airflow-env.amazonaws.com"]
        },
        "Action": "sts:AssumeRole"
      }
   ]
```

```
}
```

Kemudian lampirkan kebijakan JSON berikut ke Customer-managed kunci Anda. Kebijakan ini menggunakan awalan kunci `kms:EncryptionContext`kondisi untuk mengizinkan akses ke grup log Apache Airflow Anda di Log. CloudWatch

```
{
  "Sid": "Allow logs access",
  "Effect": "Allow",
  "Principal": {
    "Service": "logs.us-east-1.amazonaws.com"
  },
  "Action": [
    "kms:Encrypt*",
    "kms:Decrypt*",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:Describe*"
  ],
  "Resource": "*",
  "Condition": {
    "ArnLike": {
      "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-east-1:111122223333:*"
    }
  }
}
```

Contoh kebijakan untuk AWS-kunci yang dimiliki

Contoh berikut menyajikan kebijakan peran eksekusi yang dapat Anda gunakan untuk kunci yang AWS dimiliki.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "airflow:PublishMetrics",
```

```
            "Resource": "arn:aws:airflow:us-east-1:111122223333:environment/
{your-environment-name}"
        },
        {
            "Effect": "Deny",
            "Action": "s3:ListAllMyBuckets",
            "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject*",
                "s3:GetBucket*",
                "s3:List*"
            ],
            "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "logs:PutLogEvents",
                "logs:GetLogEvents",
                "logs:GetLogRecord",
                "logs:GetLogGroupFields",
                "logs:GetQueryResults"
            ],
            "Resource": [
            "arn:aws:logs:us-east-1:111122223333:log-group:airflow-{your-
environment-name}-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
```

```
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetAccountPublicAccessBlock"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricData",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "sqs:ChangeMessageVisibility",
                "sqs:DeleteMessage",
                "sqs:GetQueueAttributes",
                "sqs:GetQueueUrl",
                "sqs:ReceiveMessage",
                "sqs:SendMessage"
            ],
            "Resource": "arn:aws:sqs:us-east-1:*:airflow-celery-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey*",
                "kms:Encrypt"
            ],
            "NotResource": "arn:aws:kms:*:111122223333:key/*",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": [
                    "sqs.us-east-1.amazonaws.com"
                    ]
```

```
                }
            }
        }
    ]
}
```

Apa selanjutnya?

- Pelajari tentang izin yang diperlukan yang Anda dan pengguna Apache Airflow Anda perlukan untuk mengakses lingkungan Anda. Mengakses lingkungan Amazon MWAA
- Pelajari tentang Menggunakan kunci yang dikelola pelanggan untuk enkripsi.
- Jelajahi lebih banyak contoh Customer-managed kebijakan.

Cross-service pencegahan wakil bingung

Masalah "confused deputy" adalah masalah keamanan di mana entitas yang tidak memiliki izin untuk melakukan tindakan dapat memengaruhi entitas yang memiliki hak akses lebih tinggi untuk melakukan tindakan. Pada tahun AWS, peniruan lintas layanan dapat mengakibatkan masalah wakil yang membingungkan. Cross-service peniruan identitas dapat terjadi ketika satu layanan (layanan panggilan) memanggil layanan lain (layanan yang disebut). Layanan panggilan dapat dimanipulasi untuk menggunakan izinnya untuk bertindak atas sumber daya pelanggan lain dengan cara yang tidak memiliki izin untuk mengakses. Untuk mencegah hal ini, AWS sediakan alat yang membantu Anda melindungi data Anda untuk semua layanan dengan prinsip layanan yang telah diberikan akses ke sumber daya di akun Anda.

Sebaiknya gunakan kunci konteks kondisi `aws:SourceAccount`global `aws:SourceArn`dan dalam peran eksekusi lingkungan Anda untuk membatasi izin yang diberikan Amazon MWAA layanan lain untuk mengakses sumber daya. Gunakan `aws:SourceArn` jika Anda ingin hanya satu sumber daya yang akan dikaitkan dengan akses lintas layanan. Gunakan `aws:SourceAccount` jika Anda ingin mengizinkan sumber daya apa pun di akun tersebut dikaitkan dengan penggunaan lintas layanan.

Cara paling efektif untuk melindungi dari masalah "confused deputy" adalah dengan menggunakan kunci konteks kondisi global `aws:SourceArn` dengan ARN lengkap sumber daya. Jika Anda tidak mengetahui ARN lengkap sumber daya atau jika Anda menentukan beberapa sumber daya, gunakan kunci kondisi konteks global `aws:SourceArn` dengan karakter wildcard (`*`) untuk bagian ARN yang tidak diketahui. Misalnya, `arn:aws:airflow:*:123456789012:environment/*`.

Nilai `aws:SourceArn` harus menjadi ARN lingkungan Amazon MWAA Anda, tempat Anda membuat peran eksekusi.

Gunakan contoh berikut untuk menerapkan kunci konteks kondisi `aws:SourceAccount` global `aws:SourceArn` dan global dalam kebijakan kepercayaan peran eksekusi lingkungan Anda untuk mencegah masalah deputi yang membingungkan. Anda dapat menggunakan kebijakan kepercayaan berikut saat membuat peran eksekusi baru.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "airflow.amazonaws.com",
                    "airflow-env.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:airflow:us-
east-1:123456789012:environment/your-environment-name"
                },
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                }
            }
        }
    ]
}
```

Mode akses Apache Airflow

Konsol Alur Kerja Terkelola Amazon untuk Apache Airflow berisi opsi bawaan untuk mengonfigurasi perutean pribadi, publik, atau publik dan pribadi ke server web Apache Airflow di lingkungan Anda. Panduan ini menjelaskan mode akses yang tersedia untuk server web Apache Airflow di lingkungan

Alur Kerja Terkelola Amazon untuk Apache Airflow, dan sumber daya tambahan yang perlu Anda konfigurasikan di VPC Amazon jika memilih opsi jaringan pribadi.

Daftar Isi

- Mode akses Apache Airflow
  - Jaringan publik
  - Jaringan pribadi
  - Akses jaringan publik dan pribadi
- Ikhtisar mode akses
  - Mode akses jaringan publik
  - Mode akses jaringan pribadi
  - Mode akses jaringan publik dan pribadi
- Pengaturan untuk mode akses
  - Pengaturan untuk jaringan publik
  - Pengaturan untuk jaringan pribadi
  - Pengaturan untuk akses jaringan publik dan pribadi
- Mengakses titik akhir VPC untuk server web Apache Airflow Anda (akses jaringan pribadi)

Mode akses Apache Airflow

Anda dapat memilih perutean pribadi, publik, atau publik dan pribadi untuk server web Apache Airflow Anda. Untuk mengaktifkan perutean privat, pilih Jaringan privat. Ini membatasi akses pengguna ke server web Apache Airflow dalam VPC Amazon. Untuk mengaktifkan perutean publik, pilih Jaringan publik. Hal ini memungkinkan pengguna mengakses server web Apache Airflow melalui Internet. Untuk mengaktifkan perutean publik dan pribadi, pilih Akses jaringan publik dan pribadi. Ini memungkinkan pengguna untuk mengakses server web Apache Airflow melalui internet sementara pekerja berkomunikasi dengan server web melalui titik akhir VPC pribadi.

Jaringan publik

Diagram arsitektur berikut menggambarkan lingkungan Amazon MWAA dengan server web publik.

Mode akses jaringan publik memungkinkan UI Apache Airflow diakses melalui internet oleh pengguna yang diberikan akses ke kebijakan IAM untuk lingkungan Anda.

> **Important**
>
> Jika lingkungan Anda menggunakan Apache Airflow versi 3 atau yang lebih baru dengan mode akses jaringan Publik, pekerja harus dapat menjangkau server web melalui internet untuk mengkomunikasikan status tugas. Jika subnet yang menghosting pekerja Anda tidak memiliki akses internet (misalnya, subnet pribadi tanpa gateway NAT), tugas DAG akan gagal. Untuk mengatasi hal ini, tingkatkan ke Apache Airflow versi 3.2.1 atau yang lebih baru dan beralih ke mode akses jaringan publik dan pribadi, yang merutekan komunikasi pekerja melalui titik akhir VPC pribadi.

Gambar berikut menggambarkan di mana menemukan opsi jaringan Publik di konsol Amazon MWAA.

Jaringan pribadi

Diagram arsitektur berikut menggambarkan lingkungan Amazon MWAA dengan server web pribadi.

Mode akses jaringan pribadi membatasi akses ke UI Apache Airflow kepada pengguna dalam VPC Amazon Anda yang telah diberikan akses ke kebijakan IAM untuk lingkungan Anda.

Saat Anda membuat lingkungan dengan akses server web pribadi, Anda harus mengemas semua dependensi Anda dalam arsip roda Python (`.whl`), lalu mereferensikan di file Anda. `.whl` `requirements.txt` Untuk petunjuk tentang pengemasan dan pemasangan dependensi Anda menggunakan roda, lihat Mengelola dependensi menggunakan roda Python.

Gambar berikut menggambarkan di mana menemukan opsi Jaringan pribadi di konsol Amazon MWAA.

Akses jaringan publik dan pribadi

Tersedia untuk Apache Airflow versi 3.2.1 dan yang lebih baru. Di Apache Airflow versi 3 dan yang lebih baru, pekerja mengkomunikasikan status tugas ke server web melalui Task API. Jika VPC Amazon Anda tidak memiliki akses internet, pekerja tidak dapat menjangkau server web publik, menyebabkan tugas DAG gagal. Mode ini menciptakan penyeimbang beban jaringan publik

untuk akses browser ke Apache Airflow UI dan titik akhir VPC pribadi untuk komunikasi worker-to-webserver, memungkinkan pekerja untuk mencapai server web tanpa akses internet. Lihat diagram arsitektur jaringan Publik dan jaringan pribadi di atas untuk setiap komponen.

> **Note**
>
> Dengan mode ini, akses browser ke Apache Airflow UI melewati URL publik. Titik akhir VPC pribadi digunakan oleh pekerja untuk komunikasi internal dan tidak dimaksudkan untuk akses browser.

Ikhtisar mode akses

Bagian ini menjelaskan titik akhir VPC (AWS PrivateLink) yang dibuat di VPC Amazon Anda saat Anda memilih jaringan Publik, Jaringan pribadi, atau mode akses jaringan publik dan pribadi.

Mode akses jaringan publik

Jika Anda memilih mode akses jaringan Publik untuk server web Apache Airflow Anda, lalu lintas jaringan dirutekan secara publik melalui internet.

- Amazon MWAA membuat titik akhir antarmuka VPC untuk database metadata Amazon Aurora PostgreSQL Anda. Titik akhir dibuat di Availability Zones yang dipetakan ke subnet pribadi Anda dan independen dari yang lain. Akun AWS
- Amazon MWAA kemudian mengikat alamat IP dari subnet pribadi Anda ke titik akhir antarmuka. Ini dirancang untuk mendukung praktik terbaik mengikat satu IP dari setiap Availability Zone dari VPC Amazon.

Mode akses jaringan pribadi

Jika Anda memilih mode akses jaringan Pribadi untuk server web Apache Airflow Anda, lalu lintas jaringan dirutekan secara pribadi dalam VPC Amazon Anda.

- Amazon MWAA membuat titik akhir antarmuka VPC untuk server web Apache Airflow Anda, dan titik akhir antarmuka untuk basis data metadata Amazon Aurora PostgreSQL Anda. Titik akhir dibuat di Availability Zones yang dipetakan ke subnet pribadi Anda dan independen dari yang lain. Akun AWS

- Amazon MWAA kemudian mengikat alamat IP dari subnet pribadi Anda ke titik akhir antarmuka. Ini dirancang untuk mendukung praktik terbaik mengikat satu IP dari setiap Availability Zone dari VPC Amazon.

Mode akses jaringan publik dan pribadi

Jika Anda memilih mode akses jaringan publik dan pribadi untuk server web Apache Airflow Anda, lalu lintas jaringan ke Apache Airflow UI dirutekan secara publik melalui internet, sementara komunikasi pekerja-ke-server web dirutekan secara pribadi dalam VPC Amazon Anda.

- Amazon MWAA membuat titik akhir antarmuka VPC untuk server web Apache Airflow Anda (untuk konektivitas pekerja), dan titik akhir antarmuka untuk database metadata Amazon Aurora PostgreSQL Anda. Titik akhir dibuat di Availability Zones yang dipetakan ke subnet pribadi Anda dan independen dari yang lain. Akun AWS
- Amazon MWAA kemudian mengikat alamat IP dari subnet pribadi Anda ke titik akhir antarmuka. Ini dirancang untuk mendukung praktik terbaik mengikat satu IP dari setiap Availability Zone dari VPC Amazon.
- Apache Airflow UI dapat diakses melalui internet melalui penyeimbang beban jaringan publik. Pengguna mengakses UI dengan cara yang sama seperti dengan mode akses jaringan Publik.

Untuk mempelajari lebih lanjut, lihatthe section called “Contoh kasus penggunaan untuk VPC Amazon dan mode akses Apache Airflow”.

Pengaturan untuk mode akses

Bagian berikut menjelaskan pengaturan dan konfigurasi tambahan yang Anda perlukan berdasarkan mode akses Apache Airflow yang Anda pilih untuk lingkungan Anda.

Pengaturan untuk jaringan publik

Jika Anda memilih opsi jaringan Publik untuk server web Apache Airflow Anda, Anda dapat mulai menggunakan Apache Airflow UI setelah Anda membuat lingkungan Anda.

Anda harus mengambil langkah-langkah berikut untuk mengonfigurasi akses bagi pengguna Anda, dan izin untuk lingkungan Anda untuk menggunakan AWS layanan lain.

1. Tambahkan izin. Amazon MWAA memerlukan izin untuk menggunakan layanan lain AWS . Saat Anda membuat lingkungan, Amazon MWAA membuat peran terkait layanan yang

memungkinkannya menggunakan tindakan IAM tertentu untuk Amazon Elastic Container Registry (Amazon ECR) Registry ECR), Log, dan Amazon EC2 CloudWatch .

Anda dapat menambahkan izin untuk menggunakan tindakan tambahan untuk layanan ini, atau menggunakan AWS layanan lain dengan menambahkan izin ke peran eksekusi Anda. Untuk mempelajari lebih lanjut, lihatPeran eksekusi Amazon MWAA.

2. Buat kebijakan pengguna. Anda mungkin perlu membuat beberapa kebijakan IAM agar pengguna dapat mengonfigurasi akses ke lingkungan dan UI Apache Airflow. Untuk mempelajari lebih lanjut, lihatMengakses lingkungan Amazon MWAA.

Pengaturan untuk jaringan pribadi

Jika Anda memilih opsi Jaringan pribadi untuk server web Apache Airflow Anda, Anda harus mengonfigurasi akses untuk pengguna Anda, izin lingkungan Anda untuk menggunakan layanan AWS lain, dan membuat mekanisme untuk mengakses sumber daya di VPC Amazon Anda dari komputer Anda.

1. Tambahkan izin. Amazon MWAA memerlukan izin untuk menggunakan layanan lain AWS . Saat Anda membuat lingkungan, Amazon MWAA membuat peran terkait layanan yang memungkinkannya menggunakan tindakan IAM tertentu untuk Amazon Elastic Container Registry (Amazon ECR) Registry ECR), Log, dan Amazon EC2 CloudWatch .

   Anda dapat menambahkan izin untuk menggunakan tindakan tambahan untuk layanan ini, atau menggunakan AWS layanan lain dengan menambahkan izin ke peran eksekusi Anda. Untuk mempelajari lebih lanjut, lihatPeran eksekusi Amazon MWAA.
2. Buat kebijakan pengguna. Anda mungkin perlu membuat beberapa kebijakan IAM agar pengguna dapat mengonfigurasi akses ke lingkungan dan UI Apache Airflow. Untuk mempelajari lebih lanjut, lihatMengakses lingkungan Amazon MWAA.
3. Aktifkan akses jaringan. Anda harus membuat mekanisme di VPC Amazon Anda untuk terhubung ke titik akhir VPC () untuk server web Apache Airflow AWS PrivateLink Anda. Misalnya, dengan membuat terowongan VPN dari komputer Anda menggunakan file AWS Client VPN.

Pengaturan untuk akses jaringan publik dan pribadi

Jika Anda memilih opsi akses jaringan publik dan pribadi untuk server web Apache Airflow Anda, Anda dapat mulai menggunakan UI Apache Airflow setelah Anda membuat lingkungan Anda. Tidak

diperlukan mekanisme akses titik akhir VPN atau VPC untuk akses browser. Apache Airflow UI dapat diakses melalui internet. Pekerja terhubung ke server web melalui titik akhir VPC pribadi secara otomatis.

Anda harus mengambil langkah-langkah berikut untuk mengonfigurasi akses bagi pengguna Anda, dan izin untuk lingkungan Anda untuk menggunakan AWS layanan lain.

1. Tambahkan izin. Amazon MWAA memerlukan izin untuk menggunakan layanan lain AWS . Saat Anda membuat lingkungan, Amazon MWAA membuat peran terkait layanan yang memungkinkannya menggunakan tindakan IAM tertentu untuk Amazon Elastic Container Registry (Amazon ECR) Registry ECR), Log, dan Amazon EC2 CloudWatch .

   Anda dapat menambahkan izin untuk menggunakan tindakan tambahan untuk layanan ini, atau menggunakan AWS layanan lain dengan menambahkan izin ke peran eksekusi Anda. Untuk mempelajari lebih lanjut, lihatPeran eksekusi Amazon MWAA.
2. Buat kebijakan pengguna. Anda mungkin perlu membuat beberapa kebijakan IAM agar pengguna dapat mengonfigurasi akses ke lingkungan dan UI Apache Airflow. Untuk mempelajari lebih lanjut, lihatMengakses lingkungan Amazon MWAA.

Mengakses titik akhir VPC untuk server web Apache Airflow Anda (akses jaringan pribadi)

Jika Anda telah memilih opsi Jaringan pribadi, Anda harus membuat mekanisme di VPC Amazon Anda untuk mengakses titik akhir VPC () untuk server web Apache Airflow AWS PrivateLink Anda. Sebaiknya gunakan VPC Amazon, grup keamanan VPC, dan subnet pribadi yang sama dengan lingkungan Amazon MWAA Anda untuk sumber daya ini.

Jika Anda telah memilih Akses jaringan publik dan pribadi, Anda tidak perlu membuat mekanisme untuk mengakses UI Apache Airflow. Ini dapat diakses melalui internet. Titik akhir VPC pribadi digunakan secara otomatis oleh pekerja untuk komunikasi internal.

Untuk mempelajari lebih lanjut, lihat Mengelola akses untuk titik akhir VPC.

Mengakses Apache Airflow

Amazon MWAA memungkinkan Anda mengakses lingkungan Apache Airflow menggunakan beberapa metode: konsol antarmuka pengguna Apache Airflow (UI), Apache Airflow CLI, dan Apache Airflow REST API. Anda dapat menggunakan konsol Amazon MWAA untuk mengakses dan memanggil DAG di Apache Airflow UI Anda, atau menggunakan APIs Amazon MWAA untuk mendapatkan token dan memanggil DAG. Bagian ini menjelaskan izin yang diperlukan untuk mengakses Apache Airflow UI, cara membuat token untuk membuat panggilan Amazon MWAA API langsung di shell perintah Anda, dan perintah yang didukung di Apache Airflow CLI.

Topik

- Prasyarat
- Buka Apache Airflow UI
- Masuk ke Apache Airflow
- Buat token akses server web Apache Airflow
- Menyiapkan domain khusus untuk server web Apache Airflow
- Membuat token CLI Apache Airflow
- Menggunakan Apache Airflow REST API
- Referensi perintah CLI Apache Airflow

Prasyarat

Bagian berikut menjelaskan langkah-langkah awal yang diperlukan untuk menggunakan perintah dan skrip di bagian ini.

Akses

- Akun AWS akses di AWS Identity and Access Management (IAM) ke kebijakan izin Amazon MWAA di. Kebijakan akses Apache Airflow UI: AmazonMWAAWebServerAccess
- Akun AWS akses di AWS Identity and Access Management (IAM) ke kebijakan izin Amazon MWAA. Kebijakan akses API dan konsol lengkap: AmazonMWAAFullApiAccess

AWS CLI

The AWS Command Line Interface (AWS CLI) adalah alat open source yang dapat Anda gunakan untuk berinteraksi dengan AWS layanan menggunakan perintah di shell baris perintah Anda. Untuk menyelesaikan langkah-langkah di halaman ini, Anda memerlukan yang berikut:

- AWS CLI — Instal versi 2.
- AWS CLI - Konfigurasi cepat dengan `aws configure`.

Buka Apache Airflow UI

Gambar berikut menampilkan tautan ke UI Apache Airflow Anda di konsol Amazon MWAA.

Masuk ke Apache Airflow

Anda memerlukan Kebijakan akses Apache Airflow UI: AmazonMWAAWebServerAccess izin untuk Akun AWS in AWS Identity and Access Management (IAM) untuk mengakses UI Apache Airflow Anda.

Untuk mengakses UI Apache Airflow

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.
3. Pilih Buka UI Aliran Udara.

Buat token akses server web Apache Airflow

Anda dapat menggunakan perintah di halaman ini untuk membuat token akses server web. Token akses menyediakan akses ke lingkungan Amazon MWAA Anda. Misalnya, Anda bisa mendapatkan token, lalu menyebarkan secara DAGs terprogram menggunakan Amazon MWAA. APIs Bagian berikut mencakup langkah-langkah untuk membuat token login web Apache Airflow menggunakan AWS CLI, skrip bash, permintaan POST API, atau skrip Python. Token yang dikembalikan dalam respons berlaku selama 60 detik.

> **Important**
>
> Efektif 19 Agustus 2025, Amazon MWAA menambahkan dukungan untuk IPv6 titik akhir, dan sekarang mendukung dan titik akhir. IPv4 IPv6 Mulai tanggal ini, semua lingkungan yang baru dibuat akan menggunakan `.on.aws` domain untuk antarmuka pengguna Airflow (UI). Pelanggan harus memigrasikan UI Airflow mereka dari `.amazonaws.com` ke `.on.aws` domain untuk lingkungan yang baru dibuat ini. Layanan endpoint Virtual Private Cloud (VPC) untuk server web dan database akan mempertahankan domain mereka saat ini `.amazonaws.com` tanpa perubahan yang diperlukan.

Daftar Isi

- Prasyarat
  - Akses
  - AWS CLI
- Menggunakan AWS CLI
- Menggunakan skrip bash
- Menggunakan skrip Python
- Apa selanjutnya?

Prasyarat

Bagian berikut menjelaskan langkah-langkah awal yang diperlukan untuk menggunakan perintah dan skrip di halaman ini.

Akses

- Akun AWS akses di AWS Identity and Access Management (IAM) ke kebijakan izin Amazon MWAA di. Kebijakan akses Apache Airflow UI: AmazonMWAAWebServerAccess
- Akun AWS akses di AWS Identity and Access Management (IAM) ke kebijakan izin Amazon MWAA. Kebijakan akses API dan konsol lengkap: AmazonMWAAFullApiAccess

AWS CLI

The AWS Command Line Interface (AWS CLI) adalah alat open source yang dapat Anda gunakan untuk berinteraksi dengan AWS layanan menggunakan perintah di shell baris perintah Anda. Untuk menyelesaikan langkah-langkah di halaman ini, Anda memerlukan yang berikut:

- AWS CLI — Instal versi 2.
- AWS CLI - Konfigurasi cepat dengan `aws configure`.

Menggunakan AWS CLI

Contoh berikut menggunakan create-web-login-tokenperintah di AWS CLI untuk membuat token login web Apache Airflow.

```
aws mwaa create-web-login-token --name YOUR_ENVIRONMENT_NAME
```

Menggunakan skrip bash

Contoh berikut menggunakan skrip bash untuk memanggil create-web-login-tokenperintah di AWS CLI untuk membuat token login web Apache Airflow.

1. Salin isi contoh kode berikut dan simpan secara lokal sebagai`get-web-token.sh`.

   ```
   #!/bin/bash
   HOST=YOUR_HOST_NAME
   YOUR_URL=https://$HOST/aws_mwaa/aws-console-sso?login=true#
   WEB_TOKEN=$(aws mwaa create-web-login-token --name YOUR_ENVIRONMENT_NAME --query
    WebToken --output text)
   echo $YOUR_URL$WEB_TOKEN
   ```

2. Gantikan placeholder *red* untuk YOUR_HOST_NAME dan. YOUR_ENVIRONMENT_NAME Misalnya, nama host untuk jaringan publik menyerupai (tanpa https://):

   ```
   123456a0-0101-2020-9e11-1b159eec9000.c2.us-east-1.airflow.amazonaws.com
   ```

3. (opsional) pengguna macOS dan Linux mungkin perlu menjalankan perintah berikut untuk memastikan skrip dapat dieksekusi.

   ```
   chmod +x get-web-token.sh
   ```

4. Jalankan skrip berikut untuk mendapatkan token login web.

```
./get-web-token.sh
```

Prompt perintah Anda menampilkan:

```
https://123456a0-0101-2020-9e11-1b159eec9000.c2.us-east-1.airflow.amazonaws.com/
aws_mwaa/aws-console-sso?login=true#{your-web-login-token}
```

Menggunakan skrip Python

Contoh berikut menggunakan metode boto3 create_web_login_token dalam skrip Python untuk membuat token login web Apache Airflow. Anda dapat menjalankan skrip ini di luar Amazon MWAA. Satu-satunya hal yang perlu Anda lakukan adalah menginstal perpustakaan boto3. Anda mungkin ingin membuat lingkungan virtual untuk menginstal perpustakaan. Ini mengasumsikan Anda telah mengonfigurasi kredensi AWS otentikasi untuk akun Anda.

1. Salin isi contoh kode berikut dan simpan secara lokal sebagai`create-web-login-token.py`.

```
import boto3
  mwaa = boto3.client('mwaa')
  response = mwaa.create_web_login_token(
    Name="YOUR_ENVIRONMENT_NAME"
  )
  webServerHostName = response["WebServerHostname"]
  webToken = response["WebToken"]
  airflowUIUrl = 'https://{0}/aws_mwaa/aws-console-sso?
login=true#{1}'.format(webServerHostName, webToken)
  print("Here is your Airflow UI URL: ")
  print(airflowUIUrl)
```

2. Gantikan placeholder untuk. *red* YOUR_ENVIRONMENT_NAME
3. Jalankan skrip berikut untuk mendapatkan token login web.

```
python3 create-web-login-token.py
```

Apa selanjutnya?

- Jelajahi operasi Amazon MWAA API yang digunakan untuk membuat token login web di. CreateWebLoginToken

Menyiapkan domain khusus untuk server web Apache Airflow

Alur Kerja Terkelola Amazon untuk Apache Airflow (Amazon MWAA) memungkinkan Anda menyiapkan domain khusus untuk server web Apache Airflow yang dikelola. Dengan menggunakan domain khusus, Anda dapat mengakses server web Apache Airflow yang dikelola Amazon MWAA di lingkungan Anda menggunakan Apache Airflow UI, Apache Airflow CLI, atau server web Apache Airflow.

> **Note**
>
> Anda hanya dapat menggunakan domain khusus dengan server web pribadi tanpa akses internet.

Kasus penggunaan untuk domain khusus di Amazon MWAA

1. Bagikan domain server web di seluruh aplikasi cloud Anda AWS — Menggunakan domain khusus memungkinkan Anda menentukan URL yang mudah digunakan untuk mengakses server web, bukan nama domain layanan yang dihasilkan. Anda dapat menyimpan domain kustom ini dan membagikannya sebagai variabel lingkungan dalam aplikasi Anda.
2. Akses server web pribadi - Jika Anda ingin mengonfigurasi akses untuk server web di VPC tanpa akses internet, menggunakan domain khusus menyederhanakan alur kerja pengalihan URL.

Topik

- Konfigurasikan domain kustom
- Menyiapkan infrastruktur jaringan

Konfigurasikan domain kustom

Untuk mengonfigurasi fitur domain kustom, Anda perlu memberikan nilai domain khusus melalui konfigurasi `webserver.base_url` Apache Airflow saat membuat atau memperbarui lingkungan Amazon MWAA Anda. Batasan berikut berlaku untuk nama domain kustom Anda:

- Nilai harus berupa nama domain yang sepenuhnya memenuhi syarat (FQDN) tanpa protokol atau jalur apa pun. Misalnya, `your-custom-domain.com`.
- Amazon MWAA tidak mengizinkan jalur di URL. Misalnya, `your-custom-domain.com/dags/` bukan nama domain kustom yang valid.
- Panjang URL dibatasi hingga 255 karakter ASCII.
- Jika Anda memberikan string kosong, secara default, lingkungan akan dibuat dengan URL server web yang dihasilkan oleh Amazon MWAA.

Gunakan contoh berikut untuk membuat lingkungan dengan nama domain webserver kustom menggunakan. AWS CLI

```
aws mwaa create-environment \
--name my-mwaa-env \
--source-bucket-arn arn:aws:s3:::amzn-s3-demo-bucket \
--airflow-configuration-options '{"webserver.base_url":"my-custom-domain.com"}' \
--network-configuration '{"SubnetIds":["subnet-0123456789abcdef","subnet-
fedcba9876543210"]}' \
--execution-role-arn arn:aws:iam::123456789012:role/my-execution-role
```

Setelah lingkungan dibuat atau diperbarui, Anda perlu mengatur infrastruktur jaringan di Anda Akun AWS untuk mengakses server web pribadi melalui domain khusus.

Untuk kembali ke URL default yang dihasilkan layanan, perbarui lingkungan pribadi Anda dan hapus opsi konfigurasi. `webserver.base_url`

Menyiapkan infrastruktur jaringan

Gunakan langkah-langkah berikut untuk menyiapkan infrastruktur jaringan yang diperlukan untuk digunakan dengan domain kustom Anda di Akun AWS.

1. Dapatkan alamat IP untuk Antarmuka Jaringan Endpoint Amazon VPC (ENI). Untuk melakukan ini, pertama, gunakan `get-environment`WebserverVpcEndpointServiceuntuk menemukan lingkungan Anda.

```
aws mwaa get-environment --name your-environment-name
```

Jika berhasil, Anda akan mendapatkan output yang mirip dengan berikut ini.

```
{
  "Environment": {
    "AirflowConfigurationOptions": {},
    "AirflowVersion": "latest-version",
    "Arn": "environment-arn",
    "CreatedAt": "2024-06-01T01:00:00-00:00",
    "DagS3Path": "dags",
    .
    .
    .
    "WebserverVpcEndpointService": "web-server-vpc-endpoint-service",
    "WeeklyMaintenanceWindowStart": "TUE:21:30"
  }
}
```

Perhatikan `WebserverVpcEndpointService` nilainya dan gunakan untuk `web-server-vpc-endpoint-service` EC2 `describe-vpc-endpoints` perintah Amazon berikut. `--filters Name=service-name,Values=`*web-server-vpc-endpoint-service-id*dalam perintah berikut.

2. Ambil detail titik akhir VPC Amazon. Perintah ini mengambil detail tentang titik akhir Amazon VPC yang cocok dengan nama layanan tertentu, mengembalikan ID titik akhir dan IDs antarmuka jaringan terkait dalam format teks.

```
aws ec2 describe-vpc-endpoints \
 --filters Name=service-name,Values=web-server-vpc-endpoint-service \
 --query 'VpcEndpoints[*].
{EndpointId:VpcEndpointId,NetworkInterfaceIds:NetworkInterfaceIds}' \
 --output text
```

3. Dapatkan detail antarmuka jaringan. Perintah ini mengambil alamat IP pribadi untuk setiap antarmuka jaringan yang terkait dengan titik akhir VPC Amazon yang diidentifikasi pada langkah sebelumnya.

```
for eni_id in $(
  aws ec2 describe-vpc-endpoints \
```

```
  --filters Name=service-name,Values=service-id \
  --query 'VpcEndpoints[*].NetworkInterfaceIds' \
  --output text
); do
aws ec2 describe-network-interfaces \
 --network-interface-ids $eni_id \
 --query 'NetworkInterfaces[*].PrivateIpAddresses[*].PrivateIpAddress' \
 --output text
    done
```

4. Gunakan `create-target-group` untuk membuat grup target baru. Anda akan menggunakan grup target ini untuk mendaftarkan alamat IP untuk titik akhir Amazon VPC server web Anda.

```
aws elbv2 create-target-group \
--name new-target-group-namne \
--protocol HTTPS \
--port 443 \
--vpc-id web-server-vpc-id \
--target-type ip \
--health-check-protocol HTTPS \
--health-check-port 443 \
--health-check-path / \
--health-check-enabled \
--matcher 'HttpCode="200,302"'
```

   Daftarkan alamat IP menggunakan `register-targets` perintah.

```
aws elbv2 register-targets \
--target-group-arn target-group-arn \
--targets Id=ip-address-1 Id=ip-address-2
```

5. Minta sertifikat ACM. Lewati langkah ini jika Anda menggunakan sertifikat yang ada.

```
aws acm request-certificate \
--domain-name my-custom-domain.com \
--validation-method DNS
```

6. Konfigurasikan Application Load Balancer. Pertama, buat penyeimbang beban, lalu buat pendengar untuk penyeimbang beban. Tentukan sertifikat ACM yang Anda buat pada langkah sebelumnya.

```
aws elbv2 create-load-balancer \
```

```
--name my-mwaa-lb \
--type application \
--subnets subnet-id-1 subnet-id-2
```

```
aws elbv2 create-listener \
--load-balancer-arn load-balancer-arn \
--protocol HTTPS \
--port 443 \
--ssl-policy ELBSecurityPolicy-2016-08 \
--certificates CertificateArn=acm-certificate-arn \
--default-actions Type=forward,TargetGroupArn=target-group-arn
```

Jika Anda menggunakan Network Load Balancer di subnet pribadi, siapkan host bastion atau Site-to-Site VPN terowongan untuk mengakses server web.

7. Buat zona yang dihosting menggunakan Route 53 untuk domain.

```
aws route53 create-hosted-zone --name my-custom-domain.com \
--caller-reference 1
```

Buat catatan A untuk domain. Untuk melakukan ini menggunakan AWS CLI, dapatkan ID zona yang dihosting menggunakan `list-hosted-zones-by-name` lalu terapkan catatan dengan`change-resource-record-sets`.

```
HOSTED_ZONE_ID=$(aws route53 list-hosted-zones-by-name \
--dns-name my-custom-domain.com \
--query 'HostedZones[0].Id' --output text)
```

```
aws route53 change-resource-record-sets \
--hosted-zone-id $HOSTED_ZONE_ID \
--change-batch '{
  "Changes": [
    {
      "Action": "CREATE",
      "ResourceRecordSet": {
        "Name": "my-custom-domain.com",
        "Type": "A",
        "AliasTarget": {
          "HostedZoneId": "load-balancer-hosted-zone-id>",
          "DNSName": "load-balancer-dns-name",
          "EvaluateTargetHealth": true
```

```
        }
      }
    }
  ]
}'
```

8. Perbarui aturan grup keamanan untuk endpoint Amazon VPC server web untuk mengikuti prinsip hak istimewa paling sedikit dengan mengizinkan lalu lintas HTTPS hanya dari subnet publik tempat Application Load Balancer berada. Simpan JSON berikut secara lokal. Misalnya, sebagai`sg-ingress-ip-permissions.json`.

```
[
  {
    "IpProtocol": "tcp",
    "FromPort": 443,
    "ToPort": 443,
    "UserIdGroupPairs": [
      {
        "GroupId": "load-balancer-security-group-id"
      }
    ],
    "IpRanges": [
      {
        "CidrIp": "public-subnet-1-cidr"
      },
      {
        "CidrIp": "public-subnet-2-cidr"
      }
    ]
  }
]
```

Jalankan EC2 perintah Amazon berikut untuk memperbarui aturan grup keamanan ingress Anda. Tentukan file JSON untuk`--ip-permissions`.

```
aws ec2 authorize-security-group-ingress \
--group-id <security-group-id> \
--ip-permissions file://sg-ingress-ip-permissions.json
```

Jalankan EC2 perintah Amazon berikut untuk memperbarui aturan jalan keluar Anda.

```
aws ec2 authorize-security-group-egress \
--group-id webserver-vpc-endpoint-security-group-id \
--protocol tcp \
--port 443 \
--source-group load-balancer-security-group-id
```

Buka konsol Amazon MWAA dan arahkan ke Apache Airflow UI. Jika Anda menyiapkan Network Load Balancer di subnet pribadi alih-alih Application Load Balancer yang digunakan di sini, Anda harus mengakses server web dengan salah satu opsi berikut.

- the section called “Tutorial: Linux Bastion Host”
- the section called “Tutorial: AWS Client VPN”

Membuat token CLI Apache Airflow

> **Tip**
>
> REST API lebih modern daripada CLI dan dirancang untuk integrasi terprogram dengan sistem eksternal. REST adalah cara yang disukai untuk berinteraksi dengan Apache Airflow.

Anda dapat menggunakan perintah di halaman ini untuk menghasilkan token CLI, dan kemudian membuat Amazon Managed Workflows untuk Apache Airflow API panggilan langsung di shell perintah Anda. Misalnya, Anda bisa mendapatkan token, lalu menyebarkan secara DAGs terprogram menggunakan Amazon MWAA. APIs Bagian berikut mencakup langkah-langkah untuk membuat token Apache Airflow CLI menggunakan AWS CLI, skrip curl, skrip Python, atau skrip bash. Token yang dikembalikan dalam respons berlaku selama 60 detik.

AWS CLI Token dimaksudkan sebagai pengganti tindakan shell sinkron, bukan perintah API asinkron. Dengan demikian, konkurensi yang tersedia terbatas. Untuk memastikan bahwa server web tetap responsif bagi pengguna, kami sarankan untuk tidak membuka AWS CLI permintaan baru sampai yang sebelumnya berhasil diselesaikan.

Daftar Isi

- Prasyarat
  - Akses

  - AWS CLI
- Menggunakan AWS CLI
- Menggunakan skrip curl
- Menggunakan skrip bash
- Menggunakan skrip Python
- Apa selanjutnya?

Prasyarat

Bagian berikut menjelaskan langkah-langkah awal yang diperlukan untuk menggunakan perintah dan skrip di halaman ini.

Akses

- Akun AWS akses di AWS Identity and Access Management (IAM) ke kebijakan izin Amazon MWAA di. Kebijakan akses Apache Airflow UI: AmazonMWAAWebServerAccess
- Akun AWS akses di AWS Identity and Access Management (IAM) ke kebijakan izin Amazon MWAA. Kebijakan akses API dan konsol lengkap: AmazonMWAAFullApiAccess

AWS CLI

The AWS Command Line Interface (AWS CLI) adalah alat open source yang dapat Anda gunakan untuk berinteraksi dengan AWS layanan menggunakan perintah di shell baris perintah Anda. Untuk menyelesaikan langkah-langkah di halaman ini, Anda memerlukan yang berikut:

- AWS CLI — Instal versi 2.
- AWS CLI - Konfigurasi cepat dengan `aws configure`.

Menggunakan AWS CLI

Contoh berikut menggunakan create-cli-tokenperintah di AWS CLI untuk membuat token CLI Apache Airflow.

```
aws mwaa create-cli-token --name YOUR_ENVIRONMENT_NAME
```

Menggunakan skrip curl

Contoh berikut menggunakan skrip curl untuk memanggil create-web-login-tokenperintah dalam untuk memanggil Apache Airflow CLI melalui endpoint pada webserver Apache Airflow. AWS CLI

Apache Airflow v3

1. Salin pernyataan curl dari file teks Anda dan tempel di shell perintah Anda.

   > **Note**
   >
   > Setelah menyalinnya ke clipboard Anda, Anda mungkin perlu menggunakan Edit > Paste dari menu shell Anda.

   ```
   CLI_JSON=$(aws mwaa --region us-east-1 create-cli-token --
   name YOUR_ENVIRONMENT_NAME) \
   && CLI_TOKEN=$(echo $CLI_JSON | jq -r '.CliToken') \
   && WEB_SERVER_HOSTNAME=$(echo $CLI_JSON | jq -r '.WebServerHostname') \
   && CLI_RESULTS=$(curl -L --request POST "https://$WEB_SERVER_HOSTNAME/aws_mwaa/
   cli" \
   --header "Authorization: Bearer $CLI_TOKEN" \
   --header "Content-Type: text/plain" \
   --data-raw "dags trigger YOUR_DAG_NAME --logical-date $(date -u +"%Y-%m-%dT%H:
   %M:%SZ")") \
   && echo "Output:" \
   && echo $CLI_RESULTS | jq -r '.stdout' | base64 --decode \
   && echo "Errors:" \
   && echo $CLI_RESULTS | jq -r '.stderr' | base64 --decode
   ```

2. Gantikan placeholder *red* untuk lingkungan Anda,*YOUR_DAG_NAME*, dan. Wilayah AWS *YOUR_ENVIRONMENT_NAME* Misalnya, nama host untuk jaringan publik menyerupai (tanpa https://):

   ```
   123456a0-0101-2020-9e11-1b159eec9000.c2.us-east-1.airflow.amazonaws.com
   ```

   Prompt perintah Anda menampilkan:

   ```
   {
     "stderr":"<STDERR of the CLI execution (if any), base64 encoded>",
   ```

```
  "stdout":"<STDOUT of the CLI execution, base64 encoded>"
}
```

Apache Airflow v2

1. Salin pernyataan curl dari file teks Anda dan tempel di shell perintah Anda.

   > **Note**
   >
   > Setelah menyalinnya ke clipboard Anda, Anda mungkin perlu menggunakan Edit > Paste dari menu shell Anda.

   ```
   CLI_JSON=$(aws mwaa --region us-east-1 create-cli-token --
   name YOUR_ENVIRONMENT_NAME) \
   && CLI_TOKEN=$(echo $CLI_JSON | jq -r '.CliToken') \
   && WEB_SERVER_HOSTNAME=$(echo $CLI_JSON | jq -r '.WebServerHostname') \
   && CLI_RESULTS=$(curl --request POST "https://$WEB_SERVER_HOSTNAME/aws_mwaa/cli"
    \
   --header "Authorization: Bearer $CLI_TOKEN" \
   --header "Content-Type: text/plain" \
   --data-raw "dags trigger YOUR_DAG_NAME") \
   && echo "Output:" \
   && echo $CLI_RESULTS | jq -r '.stdout' | base64 --decode \
   && echo "Errors:" \
   && echo $CLI_RESULTS | jq -r '.stderr' | base64 --decode
   ```

2. Gantikan placeholder *red* untuk lingkungan Anda,YOUR_DAG_NAME, dan. Wilayah AWS YOUR_ENVIRONMENT_NAME Misalnya, nama host untuk jaringan publik menyerupai (tanpa https://):

   ```
   123456a0-0101-2020-9e11-1b159eec9000.c2.us-east-1.airflow.amazonaws.com
   ```

   Prompt perintah Anda menampilkan:

   ```
   {
     "stderr":"<STDERR of the CLI execution (if any), base64 encoded>",
     "stdout":"<STDOUT of the CLI execution, base64 encoded>"
   }
   ```

Menggunakan skrip bash

Contoh berikut menggunakan skrip bash untuk memanggil create-cli-tokenperintah di AWS CLI untuk membuat token CLI Apache Airflow.

Apache Airflow v3

1. Salin isi contoh kode berikut dan simpan secara lokal sebagai`get-cli-token.sh`.

```
# brew install jq
        aws mwaa create-cli-token --name YOUR_ENVIRONMENT_NAME | export
 CLI_TOKEN=$(jq -r .CliToken) && curl -L --request POST "https://YOUR_HOST_NAME/
aws_mwaa/cli" \
        --header "Authorization: Bearer $CLI_TOKEN" \
        --header "Content-Type: text/plain" \
        --data-raw "dags trigger YOUR_DAG_NAME --logical-date $(date -u +"%Y-%m-
%dT%H:%M:%SZ")"
```

2. Gantikan placeholder *red* untuk`YOUR_ENVIRONMENT_NAME`,`YOUR_HOST_NAME`, dan. `YOUR_DAG_NAME` Misalnya, nama host untuk jaringan publik menyerupai (tanpa https://):

```
123456a0-0101-2020-9e11-1b159eec9000.c2.us-east-1.airflow.amazonaws.com
```

3. (opsional) pengguna macOS dan Linux mungkin perlu menjalankan perintah berikut untuk memastikan skrip dapat dieksekusi.

```
chmod +x get-cli-token.sh
```

4. Jalankan skrip berikut untuk membuat token CLI Apache Airflow.

```
./get-cli-token.sh
```

Apache Airflow v2

1. Salin isi contoh kode berikut dan simpan secara lokal sebagai`get-cli-token.sh`.

```
# brew install jq
aws mwaa create-cli-token --name YOUR_ENVIRONMENT_NAME | export CLI_TOKEN=$(jq -
r .CliToken) && curl --request POST "https://YOUR_HOST_NAME/aws_mwaa/cli" \
--header "Authorization: Bearer $CLI_TOKEN" \
```

```
--header "Content-Type: text/plain" \
--data-raw "dags trigger YOUR_DAG_NAME"
```

2. Gantikan placeholder *red* untukYOUR_ENVIRONMENT_NAME,YOUR_HOST_NAME, dan. YOUR_DAG_NAME Misalnya, nama host untuk jaringan publik menyerupai (tanpa https://):

```
123456a0-0101-2020-9e11-1b159eec9000.c2.us-east-1.airflow.amazonaws.com
```

3. (opsional) pengguna macOS dan Linux dapat menjalankan perintah berikut untuk memastikan skrip dapat dieksekusi.

```
chmod +x get-cli-token.sh
```

4. Jalankan skrip berikut untuk membuat token CLI Apache Airflow.

```
./get-cli-token.sh
```

Menggunakan skrip Python

Contoh berikut menggunakan metode boto3 create_cli_token dalam skrip Python untuk membuat token CLI Apache Airflow dan memicu DAG. Anda dapat menjalankan skrip ini di luar Amazon MWAA. Satu-satunya hal yang perlu Anda lakukan adalah menginstal perpustakaan boto3. Anda mungkin ingin membuat lingkungan virtual untuk menginstal perpustakaan. Ini mengasumsikan Anda telah mengonfigurasi kredensi AWS otentikasi untuk akun Anda.

Apache Airflow v3

1. Salin isi contoh kode berikut dan simpan secara lokal sebagaicreate-cli-token.py.

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
```

```
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
import boto3
import json
import requests
import base64

mwaa_env_name = 'YOUR_ENVIRONMENT_NAME'
dag_name = 'YOUR_DAG_NAME'
mwaa_cli_command = 'dags trigger'

client = boto3.client('mwaa')

mwaa_cli_token = client.create_cli_token(
    Name=mwaa_env_name
)

mwaa_auth_token = 'Bearer ' + mwaa_cli_token['CliToken']
mwaa_webserver_hostname = 'https://{0}/aws_mwaa/
cli'.format(mwaa_cli_token['WebServerHostname'])
raw_data = '{0} {1}'.format(mwaa_cli_command, dag_name)

mwaa_response = requests.post(
        mwaa_webserver_hostname,
        headers={
            'Authorization': mwaa_auth_token,
            'Content-Type': 'text/plain'
            },
        data=raw_data
        )

mwaa_std_err_message = base64.b64decode(mwaa_response.json()
['stderr']).decode('utf8')
mwaa_std_out_message = base64.b64decode(mwaa_response.json()
['stdout']).decode('utf8')

print(mwaa_response.status_code)
print(mwaa_std_err_message)
print(mwaa_std_out_message)
```

2. Gantikan placeholder untuk YOUR_ENVIRONMENT_NAME dan. YOUR_DAG_NAME

3. Jalankan skrip berikut untuk membuat token CLI Apache Airflow.

```
python3 create-cli-token.py
```

Apache Airflow v2

1. Salin isi contoh kode berikut dan simpan secara lokal sebagai`create-cli-token.py`.

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
import boto3
import json
import requests
import base64

mwaa_env_name = 'YOUR_ENVIRONMENT_NAME'
dag_name = 'YOUR_DAG_NAME'
mwaa_cli_command = 'dags trigger'

client = boto3.client('mwaa')

mwaa_cli_token = client.create_cli_token(
    Name=mwaa_env_name
)

mwaa_auth_token = 'Bearer ' + mwaa_cli_token['CliToken']
mwaa_webserver_hostname = 'https://{0}/aws_mwaa/
cli'.format(mwaa_cli_token['WebServerHostname'])
```

```
raw_data = '{0} {1}'.format(mwaa_cli_command, dag_name)

mwaa_response = requests.post(
        mwaa_webserver_hostname,
        headers={
            'Authorization': mwaa_auth_token,
            'Content-Type': 'text/plain'
            },
        data=raw_data
        )

mwaa_std_err_message = base64.b64decode(mwaa_response.json()
['stderr']).decode('utf8')
mwaa_std_out_message = base64.b64decode(mwaa_response.json()
['stdout']).decode('utf8')

print(mwaa_response.status_code)
print(mwaa_std_err_message)
print(mwaa_std_out_message)
```

2. Gantikan placeholder untuk `YOUR_ENVIRONMENT_NAME` dan. `YOUR_DAG_NAME`
3. Jalankan skrip berikut untuk membuat token CLI Apache Airflow.

```
python3 create-cli-token.py
```

Apa selanjutnya?

- Jelajahi operasi Amazon MWAA API yang digunakan untuk membuat token CLI di. CreateCliToken

Menggunakan Apache Airflow REST API

Alur Kerja Terkelola Amazon untuk Apache Airflow (Amazon MWAA) mendukung interaksi dengan lingkungan Apache Airflow Anda secara langsung menggunakan API REST Apache Airflow untuk lingkungan yang menjalankan Apache Airflow v2.4.3 dan yang lebih baru. Ini memungkinkan Anda mengakses dan mengelola lingkungan Amazon MWAA Anda secara terprogram, menyediakan cara standar untuk menjalankan alur kerja orkestrasi data, mengelola, dan DAGs memantau status berbagai komponen Apache Airflow seperti database metadata, pemicu, dan penjadwal.

Untuk mendukung skalabilitas saat menggunakan Apache Airflow REST API, Amazon MWAA memberi Anda opsi untuk menskalakan kapasitas server web secara horizontal untuk menangani peningkatan permintaan, baik dari permintaan REST API, penggunaan antarmuka baris perintah (CLI), atau pengguna antarmuka pengguna Apache Airflow (UI) lainnya secara bersamaan. Untuk informasi lebih lanjut tentang bagaimana Amazon MWAA menskalakan server web, lihat. the section called “Mengkonfigurasi penskalaan otomatis server web”

Anda dapat menggunakan Apache Airflow REST API untuk mengimplementasikan kasus penggunaan berikut untuk lingkungan Anda:

- Akses terprogram - Anda sekarang dapat memulai Apache Airflow DAG berjalan, mengelola kumpulan data, dan mengambil status berbagai komponen seperti database metadata, pemicu, dan penjadwal tanpa bergantung pada Apache Airflow UI atau CLI.
- Integrasikan dengan aplikasi eksternal dan layanan mikro — Dukungan REST API yang dapat Anda gunakan untuk membangun solusi khusus yang mengintegrasikan lingkungan Amazon MWAA Anda dengan sistem lain. Misalnya, Anda dapat memulai alur kerja sebagai respons terhadap peristiwa dari sistem eksternal, seperti pekerjaan database yang diselesaikan atau pendaftaran pengguna baru.
- Pemantauan terpusat — Anda dapat membuat dasbor pemantauan yang menggabungkan status Anda di DAGs beberapa lingkungan Amazon MWAA, memungkinkan pemantauan dan pengelolaan terpusat.

Untuk informasi selengkapnya tentang Apache Airflow REST API, lihat Referensi API Apache Airflow REST.

Dengan menggunakan`InvokeRestApi`, Anda dapat mengakses Apache Airflow REST API menggunakan kredensil. AWS Atau, Anda juga dapat mengaksesnya dengan mendapatkan token akses server web dan kemudian menggunakan token untuk memanggilnya.

Jika Anda menemukan kesalahan dengan pesan `Update your environment to use InvokeRestApi` saat menggunakan `InvokeRestApi` operasi, ini menunjukkan bahwa Anda perlu memperbarui lingkungan Amazon MWAA Anda. Kesalahan ini terjadi ketika lingkungan Amazon MWAA Anda tidak kompatibel dengan perubahan terbaru yang terkait dengan fitur tersebut`InvokeRestApi`. Untuk mengatasi masalah ini, perbarui lingkungan Amazon MWAA Anda untuk memasukkan perubahan yang diperlukan untuk fitur tersebut`InvokeRestApi`.

`InvokeRestApi`Operasi ini memiliki durasi batas waktu default 10 detik. Jika operasi tidak selesai dalam jangka waktu 10 detik ini, operasi secara otomatis dihentikan, dan kesalahan muncul. Pastikan

bahwa panggilan REST API Anda dirancang untuk diselesaikan dalam periode waktu tunggu ini untuk menghindari kesalahan.

Untuk mendukung skalabilitas saat menggunakan Apache Airflow REST API, Amazon MWAA memberi Anda opsi untuk menskalakan kapasitas server web secara horizontal untuk menangani peningkatan permintaan, baik dari permintaan REST API, penggunaan antarmuka baris perintah (CLI), atau pengguna antarmuka pengguna Apache Airflow (UI) yang lebih bersamaan. Untuk informasi selengkapnya tentang bagaimana Amazon MWAA menskalakan server web, lihat. the section called “Mengkonfigurasi penskalaan otomatis server web”

Anda dapat menggunakan Apache Airflow REST API untuk mengimplementasikan kasus penggunaan berikut untuk lingkungan Anda:

- Akses terprogram - Anda sekarang dapat memulai Apache Airflow DAG berjalan, mengelola kumpulan data, dan mengambil status berbagai komponen seperti database metadata, pemicu, dan penjadwal tanpa bergantung pada Apache Airflow UI atau CLI.
- Integrasikan dengan aplikasi eksternal dan layanan mikro — Dukungan REST API yang dapat Anda gunakan untuk membangun solusi khusus yang mengintegrasikan lingkungan Amazon MWAA Anda dengan sistem lain. Misalnya, Anda dapat memulai alur kerja sebagai respons terhadap peristiwa dari sistem eksternal, seperti pekerjaan database yang diselesaikan atau pendaftaran pengguna baru.
- Pemantauan terpusat — Anda dapat membuat dasbor pemantauan yang menggabungkan status Anda di DAGs beberapa lingkungan Amazon MWAA, memungkinkan pemantauan dan pengelolaan terpusat.

Untuk informasi selengkapnya tentang Apache Airflow REST API, lihat Referensi API Apache Airflow REST.

Dengan menggunakan`InvokeRestApi`, Anda dapat mengakses Apache Airflow REST API menggunakan kredensil. AWS Atau, Anda juga dapat mengaksesnya dengan mendapatkan token akses server web dan kemudian menggunakan token untuk memanggilnya.

- Jika Anda menemukan kesalahan dengan pesan `Update your environment to use InvokeRestApi` saat menggunakan `InvokeRestApi` operasi, ini menunjukkan bahwa Anda perlu memperbarui lingkungan Amazon MWAA Anda. Kesalahan ini terjadi ketika lingkungan Amazon MWAA Anda tidak kompatibel dengan perubahan terbaru yang terkait dengan fitur tersebut`InvokeRestApi`. Untuk mengatasi masalah ini, perbarui lingkungan Amazon MWAA Anda untuk memasukkan perubahan yang diperlukan untuk fitur tersebut`InvokeRestApi`.

- `InvokeRestApi`Operasi ini memiliki durasi batas waktu default 10 detik. Jika operasi tidak selesai dalam jangka waktu 10 detik ini, operasi secara otomatis dihentikan, dan kesalahan muncul. Pastikan bahwa panggilan REST API Anda dirancang untuk diselesaikan dalam periode waktu tunggu ini untuk menghindari kesalahan.

> **Important**
>
> Ukuran payload respons tidak boleh melebihi 6 MB. Anda `RestApi` gagal jika batas ini terlampaui.

Gunakan contoh berikut untuk melakukan panggilan API ke Apache Airflow REST API dan memulai DAG run baru:

Topik

- Memberikan akses ke Apache Airflow REST API: airflow:InvokeRestApi
- Memanggil API REST Apache Airflow
- Membuat token sesi server web dan memanggil Apache Airflow REST API

Memberikan akses ke Apache Airflow REST API: `airflow:InvokeRestApi`

Untuk mengakses Apache Airflow REST API AWS menggunakan kredensil, Anda harus memberikan izin `airflow:InvokeRestApi` dalam kebijakan IAM Anda. Dalam contoh kebijakan berikut, tentukan `Public` peran `Admin``Op`, `User``Viewer`,,, atau `{airflow-role}` untuk menyesuaikan tingkat akses pengguna. Untuk informasi selengkapnya, lihat Peran Default dalam panduan referensi Apache Airflow.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowMwaaRestApiAccess",
            "Effect": "Allow",
```

```
            "Action": "airflow:InvokeRestApi",
            "Resource": [
            "arn:aws:airflow:us-east-1:111122223333:role/{your-environment-name}/
{airflow-role}"
            ]
        }
    ]
}
```

> **Note**
>
> Saat mengonfigurasi server web pribadi, `InvokeRestApi` tindakan tidak dapat dipanggil dari luar Virtual Private Cloud (VPC). Anda dapat menggunakan `aws:SourceVpc` kunci untuk menerapkan kontrol akses yang lebih terperinci untuk operasi ini. Untuk informasi lebih lanjut, lihat aws: SourceVpc.

Memanggil API REST Apache Airflow

Contoh skrip berikut ini mencakup cara menggunakan Apache Airflow REST API untuk mencantumkan yang DAGs tersedia di lingkungan Anda dan cara membuat variabel Apache Airflow:

```
import boto3

  env_name = "MyAirflowEnvironment"

  def list_dags(client):
    request_params = {
      "Name": env_name,
      "Path": "/dags",
      "Method": "GET",
      "QueryParameters": {
        "paused": False
      }
    }
  response = client.invoke_rest_api(
    **request_params
  )

  print("Airflow REST API response: ", response['RestApiResponse'])
```

```
def create_variable(client):
  request_params = {
    "Name": env_name,
    "Path": "/variables",
    "Method": "POST",
    "Body": {
      "key": "test-restapi-key",
      "value": "test-restapi-value",
      "description": "Test variable created by MWAA InvokeRestApi API",
    }
  }
response = client.invoke_rest_api(
  **request_params
)

print("Airflow REST API response: ", response['RestApiResponse'])

if __name__ == "__main__":
  client = boto3.client("mwaa")
  list_dags(client)
  create_variable(client)
```

Membuat token sesi server web dan memanggil Apache Airflow REST API

Untuk membuat token akses server web, gunakan fungsi Python berikut. Fungsi ini pertama-tama memanggil Amazon MWAA API untuk mendapatkan token login web. Token login web, yang kedaluwarsa setelah 60 detik, kemudian ditukar dengan token sesi web, yang memungkinkan Anda mengakses server web dan menggunakan API REST Apache Airflow. Jika Anda memerlukan lebih dari 10 transaksi per detik (TPS) kapasitas pelambatan, Anda dapat menggunakan metode ini untuk mengakses Apache Airflow REST API.

Token sesi berakhir setelah 12 jam.

> **Tip**
>
> Perubahan utama dalam contoh kode berikut dari Apache Airflow v2 ke v3 adalah:
>
> - Jalur REST API diubah dari `/api/v1` menjadi `/api/v2`
> - Jalur masuk diubah dari `/aws_maa/login` menjadi `/pluginsv2/aws_mwaa/login`
> - Respons dari login `response.cookies["_token"]` berisi informasi token yang harus Anda gunakan untuk panggilan API berikutnya

- Untuk panggilan REST API, Anda harus meneruskan `jwt_token` informasi di header sebagai:

```
headers = {
  "Authorization": f"Bearer {jwt_token}",
  "Content-Type": "application/json"
}
```

Apache Airflow v3

```
def get_token_info(region, env_name):
  logging.basicConfig(level=logging.INFO)

  try:
    # Initialize MWAA client and request a web login token
    mwaa = boto3.client('mwaa', region_name=region)
    response = mwaa.create_web_login_token(Name=env_name)

    # Extract the web server hostname and login token
    web_server_host_name = response["WebServerHostname"]
    web_token = response["WebToken"]

    # Construct the URL needed for authentication
    login_url = f"https://{web_server_host_name}/pluginsv2/aws_mwaa/login"
    login_payload = {"token": web_token}

    # Make a POST request to the MWAA login url using the login payload
    response = requests.post(
      login_url,
      data=login_payload,
      timeout=10
    )

    # Check if login was successful
    if response.status_code == 200:

    # Return the hostname and the session cookie
    return (
      web_server_host_name,
      response.cookies['_token']
    )
```

```
    else:
      # Log an error
      logging.error("Failed to log in: HTTP %d", response.status_code)
      return None
      except requests.RequestException as e:

      # Log any exceptions raised during the request to the MWAA login endpoint
      logging.error("Request failed: %s", str(e))
      return None
      except Exception as e:

      # Log any other unexpected exceptions
      logging.error("An unexpected error occurred: %s", str(e))
      return None
```

Apache Airflow v2

```
def get_session_info(region, env_name):
  logging.basicConfig(level=logging.INFO)

  try:
      # Initialize MWAA client and request a web login token
      mwaa = boto3.client('mwaa', region_name=region)
      response = mwaa.create_web_login_token(Name=env_name)

      # Extract the web server hostname and login token
      web_server_host_name = response["WebServerHostname"]
      web_token = response["WebToken"]

      # Construct the URL needed for authentication
      login_url = f"https://{web_server_host_name}/aws_mwaa/login"
      login_payload = {"token": web_token}

      # Make a POST request to the MWAA login url using the login payload
      response = requests.post(
          login_url,
          data=login_payload,
          timeout=10
      )

      # Check if login was succesfull
      if response.status_code == 200:
```

```
            # Return the hostname and the session cookie
            return (
                web_server_host_name,
                response.cookies["session"]
            )
        else:
            # Log an error
            logging.error("Failed to log in: HTTP %d", response.status_code)
            return None
    except requests.RequestException as e:
         # Log any exceptions raised during the request to the MWAA login endpoint
        logging.error("Request failed: %s", str(e))
        return None
    except Exception as e:
        # Log any other unexpected exceptions
        logging.error("An unexpected error occurred: %s", str(e))
        return None
```

Setelah otentikasi selesai, Anda memiliki kredensional untuk mulai mengirim permintaan ke titik akhir API. Dalam contoh di bagian berikut, gunakan titik akhir`dags/{dag_name}/dagRuns`.

Apache Airflow v3

```
def trigger_dag(region, env_name, dag_id):
      """
      Triggers a DAG in a specified MWAA environment using the Airflow REST API.

      Args:
      region (str): AWS region where the MWAA environment is hosted.
      env_name (str): Name of the MWAA environment.
      dag_id (str): ID of the DAG to trigger.
      """

      logging.info(f"Attempting to trigger DAG {dag_id} in environment {env_name} at
 region {region}")

      # Retrieve the web server hostname and token for authentication
      try:
      web_server_host_name, jwt_token = get_token_info(region, env_name)
      if not jwt_token:
      logging.error("Authentication failed, no jwt token retrieved.")
      return
```

```
    except Exception as e:
    logging.error(f"Error retrieving token info: {str(e)}")
    return

    # Prepare headers and payload for the request
    request_headers = {
    "Authorization": f"Bearer {jwt_token}",
    "Content-Type": "application/json" # Good practice to include, even for GET
    }

    # sample request body input
    json_body = {"logical_date": "2025-09-17T14:15:00Z"}

    # Construct the URL for triggering the DAG
    url = f"https://{web_server_host_name}/api/v2/dags/{dag_id}/dagRuns"

    # Send the POST request to trigger the DAG
    try:
    response = requests.post(url, headers=request_headers, json=json_body)
    # Check the response status code to determine if the DAG was triggered
successfully
    if response.status_code == 200:
    logging.info("DAG triggered successfully.")
    else:
    logging.error(f"Failed to trigger DAG: HTTP {response.status_code} -
{response.text}")
    except requests.RequestException as e:
    logging.error(f"Request to trigger DAG failed: {str(e)}")

    if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO)

    # Check if the correct number of arguments is provided
    if len(sys.argv) != 4:
    logging.error("Incorrect usage. Proper format: python script_name.py {region}
{env_name} {dag_id}")
    sys.exit(1)

    region = sys.argv[1]
    env_name = sys.argv[2]
    dag_id = sys.argv[3]

    # Trigger the DAG with the provided arguments
```

```
      trigger_dag(region, env_name, dag_id)
```

Apache Airflow v2

```
def trigger_dag(region, env_name, dag_name):
      """
      Triggers a DAG in a specified MWAA environment using the Airflow REST API.

      Args:
      region (str): AWS region where the MWAA environment is hosted.
      env_name (str): Name of the MWAA environment.
      dag_name (str): Name of the DAG to trigger.
      """

      logging.info(f"Attempting to trigger DAG {dag_name} in environment {env_name}
 at region {region}")

      # Retrieve the web server hostname and session cookie for authentication
      try:
      web_server_host_name, session_cookie = get_session_info(region, env_name)
      if not session_cookie:
      logging.error("Authentication failed, no session cookie retrieved.")
      return
      except Exception as e:
      logging.error(f"Error retrieving session info: {str(e)}")
      return

      # Prepare headers and payload for the request
      cookies = {"session": session_cookie}
      json_body = {"conf": {}}

      # Construct the URL for triggering the DAG
      url = f"https://{web_server_host_name}/api/v1/dags/{dag_id}/dagRuns"

      # Send the POST request to trigger the DAG
      try:
      response = requests.post(url, cookies=cookies, json=json_body)
      # Check the response status code to determine if the DAG was triggered
 successfully
      if response.status_code == 200:
      logging.info("DAG triggered successfully.")
      else:
```

```
    logging.error(f"Failed to trigger DAG: HTTP {response.status_code} -
{response.text}")
    except requests.RequestException as e:
    logging.error(f"Request to trigger DAG failed: {str(e)}")

    if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO)

    # Check if the correct number of arguments is provided
    if len(sys.argv) != 4:
    logging.error("Incorrect usage. Proper format: python script_name.py {region}
{env_name} {dag_name}")
    sys.exit(1)

    region = sys.argv[1]
    env_name = sys.argv[2]
    dag_name = sys.argv[3]

    # Trigger the DAG with the provided arguments
    trigger_dag(region, env_name, dag_name)
```

Referensi perintah CLI Apache Airflow

Topik ini menjelaskan perintah Apache Airflow CLI yang didukung dan tidak didukung di Alur Kerja Terkelola Amazon untuk Apache Airflow.

> **Tip**
>
> REST API lebih modern daripada CLI dan dirancang untuk integrasi terprogram dengan sistem eksternal. REST adalah cara yang disukai untuk berinteraksi dengan Apache Airflow.

Daftar Isi

- Prasyarat
  - Akses
  - AWS CLI
- Apa yang berubah?
- Perintah CLI yang didukung

  - Perintah yang Didukung
  - Menggunakan perintah yang mengurai DAG
- Kode sampel
  - Mengatur, mendapatkan, atau menghapus variabel Apache Airflow v2
  - Tambahkan konfigurasi saat memicu DAG
  - Jalankan perintah CLI pada terowongan SSH ke host bastion

Prasyarat

Bagian berikut menjelaskan langkah-langkah awal yang diperlukan untuk menggunakan perintah dan skrip di halaman ini.

Akses

- Akun AWS akses di AWS Identity and Access Management (IAM) ke kebijakan izin Amazon MWAA di. Kebijakan akses Apache Airflow UI: AmazonMWAAWebServerAccess
- Akun AWS akses di AWS Identity and Access Management (IAM) ke kebijakan izin Amazon MWAA. Kebijakan akses API dan konsol lengkap: AmazonMWAAFullApiAccess

AWS CLI

The AWS Command Line Interface (AWS CLI) adalah alat open source yang dapat Anda gunakan untuk berinteraksi dengan AWS layanan menggunakan perintah di shell baris perintah Anda. Untuk menyelesaikan langkah-langkah di halaman ini, Anda memerlukan yang berikut:

- AWS CLI — Instal versi 2.
- AWS CLI - Konfigurasi cepat dengan `aws configure`.

Apa yang berubah?

- v3: Arsitektur aliran udara. Apache Airflow v3 memperkenalkan perubahan arsitektur yang melanggar untuk memberikan peningkatan keamanan dan skalabilitas, dan untuk membuat pemeliharaan lebih mudah. Untuk mempelajari lebih lanjut, lihat Upgrade To Airflow 3.
- v2: Struktur perintah CLI aliran udara. Apache Airflow v2 CLI diatur sehingga perintah terkait dikelompokkan bersama sebagai subperintah, yang berarti Anda perlu memperbarui skrip

Apache Airflow v1 jika Anda ingin meningkatkan ke Apache Airflow v2. Misalnya, `unpause` di Apache Airflow v1 ada di `dags unpause` Apache Airflow v2. Untuk mempelajari lebih lanjut, lihat perubahan CLI Aliran Udara di 2.0.

Perintah CLI yang didukung

Bagian berikut mencantumkan perintah Apache Airflow CLI yang tersedia di Amazon MWAA.

Perintah yang Didukung

Apache Airflow v3

Versi minor	Perintah
v3.0.6, v3.2.1	rincian aset
v3.0.6, v3.2.1	daftar aset
v3.0.6, v3.2.1	aset terwujud
v3.0.6, v3.2.1	isi ulang buat
v3.0.6, v3.2.1	lembar contekan
v3.0.6, v3.2.1	koneksi menambahkan
v3.0.6, v3.2.1	koneksi hapus
v3.0.6, v3.2.1	hari hapus
v3.0.6, v3.2.1	daftar dags
v3.0.6, v3.2.1	dags daftar-lowongan

Versi minor	Perintah
v3.0.6, v3.2.1	dags daftar-impor-kesalahan
v3.0.6, v3.2.1	dags daftar-berjalan
v3.0.6, v3.2.1	dags eksekusi berikutnya
v3.0.6, v3.2.1	dags jeda
v3.0.6, v3.2.1	laporan dags
v3.0.6, v3.2.1	dags reserialize
v3.0.6, v3.2.1	pertunjukan dags
v3.0.6, v3.2.1	negara bagian dags
v3.0.6, v3.2.1	tes dags
v3.0.6, v3.2.1	pemicu dags
v3.0.6, v3.2.1	dags unpause
v3.0.6, v3.2.1	db bersih
v3.0.6, v3.2.1	perilaku penyedia
v3.0.6, v3.2.1	penyedia mendapatkan

Versi minor	Perintah
v3.0.6, v3.2.1	penyedia kait
v3.0.6, v3.2.1	tautan penyedia
v3.0.6, v3.2.1	daftar penyedia
v3.0.6, v3.2.1	pemberitahuan penyedia
v3.0.6, v3.2.1	rahasia penyedia
v3.0.6, v3.2.1	penyedia pemicu
v3.0.6, v3.2.1	widget penyedia
v3.0.6, v3.2.1	peran tambahan perms
v3.0.6, v3.2.1	peran del-perms
v3.0.6, v3.2.1	peran membuat
v3.0.6, v3.2.1	daftar peran
v3.0.6, v3.2.1	tugas yang jelas
v3.0.6, v3.2.1	tugas gagal-deps
v3.0.6, v3.2.1	daftar tugas

Versi minor	Perintah
v3.0.6, v3.2.1	tugas render
v3.0.6, v3.2.1	status tugas
v3.0.6, v3.2.1	tugas states-for-dag-run
v3.0.6, v3.2.1	tes tugas
v3.0.6, v3.2.1	variabel menghapus
v3.0.6, v3.2.1	variabel mendapatkan
v3.0.6, v3.2.1	variabel yang ditetapkan
v3.0.6, v3.2.1	daftar variabel
v3.0.6, v3.2.1	versi

Apache Airflow v2

Versi minor	Perintah
v2.0+	lembar contekan
v2.0+	koneksi menambahkan
v2.0+	koneksi hapus
v2.2+ (catatan)	isi ulang hari

Versi minor	Perintah
v2.0+	hari hapus
v2.2+ (catatan)	daftar dags
v2.0+	dags daftar-lowongan
v2.6+	dags daftar-impor-kesalahan
v2.2+ (catatan)	dags daftar-berjalan
v2.2+ (catatan)	dags eksekusi berikutnya
v2.0+	dags jeda
v2.0+	laporan dags
v2.4+	dags reserialize
v2.0+	pertunjukan dags
v2.0+	negara bagian dags
v2.0+	tes dags
v2.0+	pemicu dags
v2.0+	dags unpause

Versi minor	Perintah
v2.4+	db bersih
v2.0+	perilaku penyedia
v2.0+	penyedia mendapatkan
v2.0+	penyedia kait
v2.0+	tautan penyedia
v2.0+	daftar penyedia
v2.8+	pemberitahuan penyedia
v2.6+	rahasia penyedia
v2.7+	penyedia pemicu
v2.0+	widget penyedia
v2.6+	peran tambahan perms
v2.6+	peran del-perms
v2.6+	peran membuat
v2.0+	daftar peran

Versi minor	Perintah
v2.0+	tugas yang jelas
v2.0+	tugas gagal-deps
v2.0+	daftar tugas
v2.0+	tugas render
v2.0+	tugas dijalankan
v2.0+	status tugas
v2.0+	tugas states-for-dag-run
v2.0+	tes tugas
v2.0+	variabel menghapus
v2.0+	variabel mendapatkan
v2.0+	variabel yang ditetapkan
v2.0+	daftar variabel
v2.0+	versi

Menggunakan perintah yang mengurai DAG

Jika lingkungan Anda menjalankan Apache Airflow v2.0.2, perintah CLI yang mengurai DAG akan gagal jika DAG menggunakan plugin yang bergantung pada paket yang diinstal melalui: `requirements.txt`

Apache Airflow v2.0.2

- `dags backfill`
- `dags list`
- `dags list-runs`
- `dags next-execution`

Anda dapat menggunakan perintah CLI ini jika DAG Anda tidak menggunakan plugin yang bergantung pada paket yang diinstal melalui file. `requirements.txt`

Kode sampel

Bagian berikut berisi contoh berbagai cara untuk menggunakan Apache Airflow CLI.

Mengatur, mendapatkan, atau menghapus variabel Apache Airflow v2

Anda dapat menggunakan kode contoh berikut untuk mengatur, mendapatkan atau menghapus variabel dalam format`<script> <mwaa env name> get | set | delete <variable> <variable value> </variable> </variable>`.

```
[ $# -eq 0 ] && echo "Usage: $0 MWAA environment name " && exit

if [[ $2 == "" ]]; then
    dag="variables list"

elif  [ $2 == "get" ] ||  [ $2 == "delete" ] ||  [ $2 == "set" ]; then
    dag="variables $2 $3 $4 $5"

else
    echo "Not a valid command"
    exit 1
fi

CLI_JSON=$(aws mwaa --region $AWS_REGION create-cli-token --name $1) \
```

```
    && CLI_TOKEN=$(echo $CLI_JSON | jq -r '.CliToken') \
    && WEB_SERVER_HOSTNAME=$(echo $CLI_JSON | jq -r '.WebServerHostname') \
    && CLI_RESULTS=$(curl --request POST "https://$WEB_SERVER_HOSTNAME/aws_mwaa/cli" \
    --header "Authorization: Bearer $CLI_TOKEN" \
    --header "Content-Type: text/plain" \
    --data-raw "$dag" ) \
    && echo "Output:" \
    && echo $CLI_RESULTS | jq -r '.stdout' | base64 --decode \
    && echo "Errors:" \
    && echo $CLI_RESULTS | jq -r '.stderr' | base64 --decode
```

Tambahkan konfigurasi saat memicu DAG

Anda dapat menggunakan kode contoh berikut dengan Apache Airflow v2 untuk menambahkan konfigurasi saat memicu DAG, seperti. `airflow trigger_dag 'dag_name' –conf '{"key":"value"}'`

```
import boto3
import json
import requests
import base64

  mwaa_env_name = 'YOUR_ENVIRONMENT_NAME'
  dag_name = 'YOUR_DAG_NAME'
  key = "YOUR_KEY"
  value = "YOUR_VALUE"
  conf = "{\"" + key + "\":\"" + value + "\"}"

  client = boto3.client('mwaa')

  mwaa_cli_token = client.create_cli_token(
    Name=mwaa_env_name
  )

  mwaa_auth_token = 'Bearer ' + mwaa_cli_token['CliToken']
  mwaa_webserver_hostname = 'https://{0}/aws_mwaa/
cli'.format(mwaa_cli_token['WebServerHostname'])
  raw_data = "trigger_dag {0} -c '{1}'".format(dag_name, conf)

  mwaa_response = requests.post(
    mwaa_webserver_hostname,
    headers={
      'Authorization': mwaa_auth_token,
```

```
      'Content-Type': 'text/plain'
    },
    data=raw_data
  )

  mwaa_std_err_message = base64.b64decode(mwaa_response.json()
['stderr']).decode('utf8')
  mwaa_std_out_message = base64.b64decode(mwaa_response.json()
['stdout']).decode('utf8')

  print(mwaa_response.status_code)
  print(mwaa_std_err_message)
  print(mwaa_std_out_message)
```

Jalankan perintah CLI pada terowongan SSH ke host bastion

Gunakan contoh berikut untuk menjalankan perintah Airflow CLI menggunakan proxy terowongan SSH ke Linux Bastion Host.

Menggunakan curl

1. ```
   ssh -D 8080 -f -C -q -N YOUR_USER@YOUR_BASTION_HOST
   ```
2. ```
   curl -x socks5h://0:8080 --request POST https://YOUR_HOST_NAME/aws_mwaa/cli --
   header YOUR_HEADERS --data-raw YOUR_CLI_COMMAND
   ```

Mengelola koneksi ke Apache Airflow

Bab ini menjelaskan cara mengonfigurasi koneksi Apache Airflow untuk lingkungan Alur Kerja Terkelola Amazon untuk Apache Airflow.

Topik

- Ikhtisar variabel dan koneksi Apache Airflow
- Paket penyedia Apache Airflow diinstal di lingkungan Amazon MWAA
- Ikhtisar jenis koneksi
- Mengkonfigurasi koneksi Apache Airflow menggunakan rahasia AWS Secrets Manager

Ikhtisar variabel dan koneksi Apache Airflow

Dalam beberapa kasus, Anda mungkin ingin menentukan koneksi atau variabel tambahan untuk lingkungan, seperti AWS profil, atau menambahkan peran eksekusi Anda dalam objek koneksi di metastore Apache Airflow, lalu merujuk ke koneksi dari dalam DAG.

- Apache Airflow yang dikelola sendiri. Pada instalasi Apache Airflow yang dikelola sendiri, Anda mengatur opsi konfigurasi Apache Airflow. `airflow.cfg`

  ```
  [secrets]
  backend = airflow.providers.amazon.aws.secrets.secrets_manager.SecretsManagerBackend
  backend_kwargs = {"connections_prefix" : "airflow/connections", "variables_prefix" :
   "airflow/variables"}
  ```

- Apache Airflow di Amazon MWAA. Di Amazon MWAA, Anda perlu menambahkan pengaturan konfigurasi ini sebagai opsi konfigurasi Apache Airflow di konsol Amazon MWAA. Opsi konfigurasi Apache Airflow ditulis sebagai variabel lingkungan ke lingkungan Anda dan mengganti semua konfigurasi lain yang ada untuk pengaturan yang sama.

Paket penyedia Apache Airflow diinstal di lingkungan Amazon MWAA

Halaman ini mencantumkan paket penyedia Apache Airflow yang diinstal oleh Amazon MWAA untuk semua lingkungan Apache Airflow yang didukung. Untuk informasi lebih lanjut tentang paket-paket ini, lihat referensi Apache Airflow untuk paket ekstra.

> **Note**
>
> Untuk memastikan bahwa kompatibilitas dengan CloudWatch logging tidak diganti oleh instalasi pustaka Python lainnya, Amazon MWAA menginstal Watchtower versi 2.0.1 setelah melakukan. `pip3 install -r requirements.txt`

Topik

- Kendala berkas
- Version-specific paket penyedia

Kendala berkas

Dimulai dengan Apache Airflow v2.7.2, file persyaratan Anda harus menyertakan pernyataan. `--constraint` Jika Anda tidak memberikan batasan, Amazon MWAA akan menentukan satu untuk Anda untuk memastikan paket yang tercantum dalam persyaratan Anda kompatibel dengan versi Apache Airflow yang Anda gunakan.

File batasan Apache Airflow menentukan versi penyedia yang tersedia pada saat rilis Apache Airflow. Namun, dalam banyak kasus, penyedia yang lebih baru kompatibel dengan versi Apache Airflow. Karena Anda harus menggunakan batasan, untuk menentukan versi paket penyedia yang lebih baru, Anda dapat memodifikasi file kendala untuk versi penyedia tertentu:

1. Unduh file batasan khusus versi dari GitHub, misalnya https://raw.githubusercontent.com/apache/airflow/constraints-2.7.2/constraints-3.11.txt(ganti '2.7.2' dengan versi yang ingin Anda gunakan).
2. Simpan file kendala yang dimodifikasi ke folder Amazon S3 dags di lingkungan Amazon MWAA Anda, misalnya, sebagai. `constraints-3.11-updated.txt`
3. Tentukan persyaratan Anda seperti yang tercantum dalam berikut ini.

```
--constraint "/usr/local/airflow/dags/constraints-3.11-updated.txt"
apache-airflow-providers-amazon==version-number
```

> **Note**
>
> Jika Anda menggunakan server web pribadi, kami sarankan Anda mengemas pustaka yang diperlukan sebagai file WHL dengan menggunakan aws-mwaa-docker-images.

Version-specific paket penyedia

Menginstal paket penyedia yang dapat Anda gunakan untuk mengakses jenis koneksi di Apache Airflow UI. Ini juga berarti Anda tidak perlu menentukan paket-paket ini sebagai ketergantungan Python dalam file Anda. `requirements.txt` Halaman ini mencantumkan paket penyedia Apache Airflow yang diinstal oleh Amazon MWAA untuk semua lingkungan Apache Airflow yang didukung.

> **Note**
>
> Untuk Apache Airflow v2 dan yang lebih baru, Amazon MWAA menginstal Watchtower versi 2.0.1 setelah `pip3 install -r requirements.txt` melakukan, untuk memastikan kompatibilitas dengan CloudWatch logging tidak diganti oleh instalasi pustaka Python lainnya.

Anda dapat menentukan versi terbaru yang didukung `apache-airflow-providers-amazon` untuk meningkatkan penyedia ini.

Versi Apache Airflow yang didukung:

v3.2.1

Tipe koneksi	Paket
AWS Koneksi	apache-penyedia-aliran udara-amazon [aiobotocore] ==9.25.0
Koneksi Postgres	

Tipe koneksi	Paket
	apache-penyedia-aliran udara-postgres==6.6.3
Koneksi FTP	apache-penyedia-aliran udara-ftp = 3.14.3
Koneksi Fab	apache-penyedia-aliran udara-fab==3.6.1
Koneksi Seledri	apache-penyedia-aliran udara-seledri = 3.18.0
Koneksi HTTP	apache-penyedia-aliran udara-http = 6.0.2
Koneksi IMAP	apache-penyedia-aliran udara-imap = 3.11.2
SQL umum	apache-airflow-providers-common-sql = 1.34.0
Koneksi SQLite	apache-airflow-providers-sqlite = 4.3.2

v3.0.6

Tipe koneksi	Paket
AWS Koneksi	apache-penyedia-aliran udara-amazon [aiobotocore] ==9.9.0
Koneksi Postgres	apache-penyedia-aliran udara-postgres==6.2.1
Koneksi FTP	apache-penyedia-aliran udara-ftp = 3.13.1

Tipe koneksi	Paket
Koneksi Fab	apache-penyedia-aliran udara-fab==2.3.0
Koneksi Seledri	apache-penyedia-aliran udara-seledri = 3.12.1
Koneksi HTTP	apache-penyedia-aliran udara-http = 5.3.2
Koneksi IMAP	apache-penyedia-aliran udara-imap = 3.9.1
SQL umum	apache-airflow-providers-common-sql = 1.27.3
Koneksi SQLite	apache-airflow-providers-sqlite = 4.1.1

v2.11.0

Tipe koneksi	Paket
AWS Koneksi	apache-penyedia-aliran udara-amazon [aiobotocore] ==9.8.0
Koneksi Postgres	apache-penyedia-aliran udara-postgres== 6.2.0
Koneksi FTP	apache-penyedia-aliran udara-ftp = 3.13.0
Koneksi Fab	apache-penyedia-aliran udara-fab==1.5.3
Koneksi Seledri	apache-penyedia-aliran udara-seledri = 3.11.0

Tipe koneksi	Paket
Koneksi HTTP	apache-penyedia-aliran udara-http = 5.3.0
Koneksi IMAP	apache-penyedia-aliran udara-imap = 3.9.0
SQL umum	apache-airflow-providers-common-sql = 1.27.1
Koneksi SQLite	apache-airflow-providers-sqlite = 4.1.0
Koneksi SMTP	apache-penyedia-aliran udara-smtp = 2.1.0

v2.10.3

Tipe koneksi	Paket
AWS Koneksi	apache-penyedia-aliran udara-amazon [aiobotocore] ==9.0.0
Koneksi Postgres	apache-penyedia-aliran udara-postgres== 5.13.1
Koneksi FTP	apache-penyedia-aliran udara-ftp = 3.11.1
Koneksi Fab	apache-penyedia-aliran udara-fab==1.5.0
Koneksi Seledri	apache-penyedia-aliran udara-seledri = 3.8.3
Koneksi HTTP	apache-penyedia-aliran udara-http = 4.13.2
Koneksi IMAP	apache-penyedia-aliran udara-imap = 3.7.0

Tipe koneksi	Paket
SQL umum	apache-airflow-providers-common-sql = 1.19.0
Koneksi SQLite	apache-airflow-providers-sqlite = 3.9.0
Koneksi SMTP	apache-penyedia-aliran udara-smtp = 1.8.0

v2.10.1

Tipe koneksi	Paket
AWS Koneksi	apache-penyedia-aliran udara-amazon [aiobotocore] ==8.28.0
Koneksi Postgres	apache-penyedia-aliran udara-postgres== 5.12.0
Koneksi FTP	apache-penyedia-aliran udara-ftp = 3.11.0
Koneksi Fab	apache-penyedia-aliran udara-fab==1.3.0
Koneksi Seledri	apache-penyedia-aliran udara-seledri = 3.8.1
Koneksi HTTP	apache-penyedia-aliran udara-http = 4.13.0
Koneksi IMAP	apache-penyedia-aliran udara-imap = 3.7.0
SQL umum	apache-airflow-providers-common-sql = 1.16.0

Tipe koneksi	Paket
Koneksi SQLite	apache-airflow-providers-sqlite = 3.9.0
Koneksi SMTP	apache-penyedia-aliran udara-smtp = 1.8.0

v2.9.2

Tipe koneksi	Paket
AWS Koneksi	apache-penyedia-aliran udara-amazon [aiobotocore] ==8.24.0
Koneksi Postgres	apache-penyedia-aliran udara-postgres== 5.11.1
Koneksi FTP	apache-penyedia-aliran udara-ftp = 3.9.1
Koneksi Fab	apache-penyedia-aliran udara-fab==1.1.1
Koneksi Seledri	apache-penyedia-aliran udara-seledri = 3.7.2
Koneksi HTTP	apache-penyedia-aliran udara-http = 4.11.1
Koneksi IMAP	apache-penyedia-aliran udara-imap = 3.6.1
SQL umum	apache-airflow-providers-common-sql = 1.14.0
Koneksi SQLite	apache-airflow-providers-sqlite = 3.8.1
Koneksi SMTP	apache-penyedia-aliran udara-smtp = 1.7.1

v2.8.1

Tipe koneksi	Paket
AWS Koneksi	apache-penyedia-aliran udara-amazon [aiobotocore] ==8.16.0
Koneksi Postgres	apache-penyedia-aliran udara-postgres== 5.10.0
Koneksi FTP	apache-penyedia-aliran udara-ftp = 3.7.0
Koneksi Seledri	apache-penyedia-aliran udara-seledri = 3.5.1
Koneksi HTTP	apache-penyedia-aliran udara-http = 4.8.0
Koneksi IMAP	apache-penyedia-aliran udara-imap = 3.5.0
SQL umum	apache-airflow-providers-common-sql = 1.10.0
Koneksi SQLite	apache-airflow-providers-sqlite = 3.7.0

v2.7.2

Tipe koneksi	Paket
AWS Koneksi	apache-penyedia-aliran udara-amazon [aiobotocore] ==8.7.1
Koneksi Postgres	apache-penyedia-aliran udara-postgres== 5.6.1

Tipe koneksi	Paket
Koneksi FTP	apache-penyedia-aliran udara-ftp = 3.5.2
Koneksi Seledri	apache-penyedia-aliran udara-seledri = 3.3.4
Koneksi HTTP	apache-penyedia-aliran udara-http = 4.5.2
Koneksi IMAP	apache-penyedia-aliran udara-imap = 3.3.2
SQL umum	apache-airflow-providers-common-sql = 1.7.2
Koneksi SQLite	apache-airflow-providers-sqlite = 3.4.3

v2.6.3

Tipe koneksi	Paket
AWS Koneksi	apache-penyedia-aliran udara-amazon [aiobotocore] ==8.2.0
Koneksi Postgres	apache-penyedia-aliran udara-postgres== 5.5.1
Koneksi FTP	apache-penyedia-aliran udara-ftp = 3.4.2
Koneksi Seledri	apache-penyedia-aliran udara-seledri = 3.2.1
Koneksi HTTP	apache-penyedia-aliran udara-http = 4.4.2
Koneksi IMAP	apache-penyedia-aliran udara-imap = 3.2.2

Tipe koneksi	Paket
SQL umum	apache-airflow-providers-common-sql = 1.5.2
Koneksi SQLite	apache-airflow-providers-sqlite = 3.4.2

v2.5.1

Tipe koneksi	Paket
AWS Koneksi	apache-penyedia-aliran udara-amazon== 7.1.0
Koneksi Postgres	apache-penyedia-aliran udara-postgres== 5.4.0
Koneksi FTP	apache-penyedia-aliran udara-ftp = 3.3.0
Koneksi Seledri	apache-penyedia-aliran udara-seledri = 3.1.0
Koneksi HTTP	apache-penyedia-aliran udara-http = 4.1.1
Koneksi IMAP	apache-penyedia-aliran udara-imap = 3.1.1
SQL umum	apache-airflow-providers-common-sql = 1.3.3
Koneksi SQLite	apache-airflow-providers-sqlite = 3.3.1

v2.4.3

Tipe koneksi	Paket
AWS Koneksi	apache-penyedia-aliran udara-amazon==6.0.0
Koneksi Postgres	apache-penyedia-aliran udara-postgres==5.2.2
Koneksi FTP	apache-penyedia-aliran udara-ftp = 3.1.0
Koneksi Seledri	apache-penyedia-aliran udara-seledri = 3.0.0
Koneksi HTTP	apache-penyedia-aliran udara-http = 4.0.0
Koneksi IMAP	apache-penyedia-aliran udara-imap = 3.0.0
SQL umum	apache-airflow-providers-common-sql = 1.2.0
Koneksi SQLite	apache-airflow-providers-sqlite = 3.2.1

Ikhtisar jenis koneksi

Apache Airflow menyimpan koneksi sebagai string URI koneksi. Ini menyediakan template koneksi di Apache Airflow UI untuk menghasilkan string URI koneksi, terlepas dari jenis koneksi. Jika template koneksi tidak tersedia di Apache Airflow UI, template koneksi alternatif dapat digunakan untuk menghasilkan string URI koneksi ini, seperti menggunakan template koneksi HTTP. Perbedaan utama adalah awalan URI, sepertimy-conn-type://, yang biasanya diabaikan oleh penyedia Apache Airflow untuk koneksi. Halaman ini menjelaskan cara menggunakan templat koneksi di Apache Airflow UI secara bergantian untuk berbagai jenis koneksi.

> **Warning**
>
> Jangan menimpa `aws_default`koneksi di Amazon MWAA. Amazon MWAA menggunakan koneksi ini untuk melakukan berbagai tugas penting, seperti mengumpulkan log tugas. Menimpa koneksi ini dapat mengakibatkan hilangnya data dan gangguan pada ketersediaan lingkungan Anda.

Topik

- Contoh koneksi URI string
- Contoh template koneksi
- Contoh menggunakan template koneksi HTTP untuk koneksi Jdbc

Contoh koneksi URI string

Contoh berikut menyajikan string URI koneksi untuk jenis koneksi MySQL.

```
'mysql://288888a0-50a0-888-9a88-1a111aaa0000.a1.us-east-1.airflow.amazonaws.com
%2Fhome?role_arn=arn%3Aaws%3Aiam%3A%3A001122332255%3Arole%2Fservice-role%2FAmazonMWAA-
MyAirflowEnvironment-iAaaaA&region_name=us-east-1'
```

Contoh template koneksi

Contoh berikut menggambarkan template koneksi HTTP di Apache Airflow UI.

Apache Airflow v3

Apache Airflow v2

Contoh menggunakan template koneksi HTTP untuk koneksi Jdbc

Gunakan contoh berikut untuk menerapkan template koneksi HTTP untuk jenis koneksi Jdbc di Apache Airflow UI.

Apache Airflow v3

Contoh berikut menampilkan string URI koneksi yang dihasilkan oleh Apache Airflow untuk contoh di bagian ini.

```
http://myconnectionurl/some/path&login=mylogin&extra__jdbc__dry__path=usr/local/
airflow/dags/classpath/redshif-
jdbc42-2.0.0.1.jar&extra__jdbc__dry__clsname=redshift-jdbc42-2.0.0.1
```

Gunakan contoh berikut untuk menerapkan template koneksi HTTP untuk koneksi Jdbc untuk Apache Airflow v3 di Apache Airflow UI.

Apache Airflow v2

Contoh berikut menampilkan string URI koneksi yang dihasilkan oleh Apache Airflow untuk contoh di bagian ini.

```
http://myconnectionurl/some/path&login=mylogin&extra__jdbc__dry__path=usr/local/
airflow/dags/classpath/redshif-
jdbc42-2.0.0.1.jar&extra__jdbc__dry__clsname=redshift-jdbc42-2.0.0.1
```

Gunakan contoh berikut untuk menerapkan template koneksi HTTP untuk koneksi Jdbc untuk Apache Airflow v2 di Apache Airflow UI.

Mengkonfigurasi koneksi Apache Airflow menggunakan rahasia AWS Secrets Manager

AWS Secrets Manager adalah backend Apache Airflow alternatif yang didukung pada Alur Kerja Terkelola Amazon untuk lingkungan Apache Airflow. Topik ini menjelaskan cara menggunakan AWS Secrets Manager untuk menyimpan rahasia dengan aman untuk variabel Apache Airflow dan koneksi Apache Airflow di Amazon Managed Workflows untuk Apache Airflow.

> **Note**
>
> - Anda dikenakan biaya untuk rahasia yang Anda buat. Untuk informasi selengkapnya tentang harga Secrets Manager, lihat AWS Harga.

- AWS Systems Manager Parameter Store juga didukung sebagai backend rahasia di Amazon MWAA. Untuk informasi selengkapnya, lihat dokumentasi Amazon Provider Package.

Daftar Isi

- Langkah satu: Berikan Amazon MWAA izin untuk mengakses kunci rahasia Secrets Manager
- Langkah kedua: Buat backend Secrets Manager sebagai opsi konfigurasi Apache Airflow
- Langkah ketiga: Hasilkan string URI koneksi Apache AWS Airflow
- Langkah empat: Tambahkan variabel di Secrets Manager
- Langkah lima: Tambahkan koneksi di Secrets Manager
- Kode sampel
- Sumber Daya
- Apa selanjutnya?

Langkah satu: Berikan Amazon MWAA izin untuk mengakses kunci rahasia Secrets Manager

Peran eksekusi untuk lingkungan Amazon MWAA Anda memerlukan akses baca ke kunci rahasia. AWS Secrets Manager Kebijakan IAM berikut memungkinkan akses baca-tulis menggunakan kebijakan -managed. AWSSecretsManagerReadWrite

Untuk melampirkan kebijakan ke peran eksekusi Anda

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.
3. Pilih peran eksekusi Anda di panel Izin.
4. Pilih Lampirkan kebijakan.
5. Ketik `SecretsManagerReadWrite` di bidang teks Filter kebijakan.
6. Pilih Lampirkan kebijakan.

Jika Anda tidak ingin menggunakan kebijakan izin yang AWS dikelola, Anda dapat langsung memperbarui peran eksekusi lingkungan Anda untuk mengizinkan tingkat akses apa pun ke sumber

daya Secrets Manager Anda. Misalnya, pernyataan kebijakan berikut memberikan akses baca ke semua rahasia yang Anda buat Wilayah AWS di Secrets Manager tertentu.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetResourcePolicy",
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "secretsmanager:ListSecretVersionIds"
            ],
            "Resource": "arn:aws:secretsmanager:us-east-1:111122223333:secret:*"
        },
        {
            "Effect": "Allow",
            "Action": "secretsmanager:ListSecrets",
            "Resource": "*"
        }
    ]
}
```

Langkah kedua: Buat backend Secrets Manager sebagai opsi konfigurasi Apache Airflow

Bagian berikut menjelaskan cara membuat opsi konfigurasi Apache Airflow di konsol Amazon MWAA untuk backend. AWS Secrets Manager Jika Anda menggunakan pengaturan konfigurasi dengan nama yang sama`airflow.cfg`, konfigurasi yang Anda buat dalam langkah-langkah berikut akan diutamakan dan mengganti pengaturan konfigurasi.

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.
3. Pilih Edit.
4. Pilih Berikutnya.

5. Pilih Tambahkan konfigurasi khusus di panel Opsi konfigurasi Aliran Udara. Tambahkan pasangan kunci-nilai berikut:

   a. **`secrets.backend`**: **`airflow.providers.amazon.aws.secrets.secrets_manager.SecretsManagerBacken`**

   b. **`secrets.backend_kwargs`**: **`{"connections_prefix" : "airflow/connections", "variables_prefix" : "airflow/variables"}`** Ini mengkonfigurasi Apache Airflow untuk mencari string koneksi dan variabel di dan jalur. `airflow/connections/*` `airflow/variables/*`

   Anda dapat menggunakan pola pencarian untuk mengurangi jumlah panggilan API yang dilakukan Amazon MWAA ke Secrets Manager atas nama Anda. Jika Anda tidak menentukan pola pencarian, Apache Airflow akan mencari semua koneksi dan variabel di backend yang dikonfigurasi. Dengan menentukan pola, Anda mempersempit kemungkinan jalur yang dicari Apache Airflow. Ini menurunkan biaya Anda saat menggunakan Secrets Manager dengan Amazon MWAA.

   Untuk menentukan pola pencarian, tentukan `variables_lookup_pattern` parameter `connections_lookup_pattern` dan. Parameter ini menerima RegEx string sebagai input. Misalnya, untuk mencari rahasia yang dimulai`test`, masukkan yang berikut ini untuk`secrets.backend_kwargs`:

   ```
   {
     "connections_prefix": "airflow/connections",
     "connections_lookup_pattern": "^test",
     "variables_prefix" : "airflow/variables",
     "variables_lookup_pattern": "^test"
   }
   ```

   > **Note**
   >
   > Untuk menggunakan `connections_lookup_pattern` dan`variables_lookup_pattern`, Anda harus menginstal `apache-airflow-providers-amazon` versi 7.3.0 atau lebih tinggi. Untuk informasi lebih lanjut tentang memperbarui paket provder ke versi yang lebih baru, lihat. Kendala berkas

6. Pilih Simpan.

Langkah ketiga: Hasilkan string URI koneksi Apache AWS Airflow

Untuk membuat string koneksi, gunakan tombol “tab” pada keyboard Anda untuk membuat indentasi pasangan kunci-nilai di objek Connection. Kami juga merekomendasikan membuat variabel untuk `extra` objek dalam sesi shell Anda. Bagian berikut memandu Anda melalui langkah-langkah untuk menghasilkan string URI koneksi Apache Airflow untuk lingkungan Amazon MWAA menggunakan Apache Airflow atau skrip Python.

Apache Airflow CLI

Sesi shell berikut menggunakan CLI Airflow lokal Anda untuk menghasilkan string koneksi. Jika Anda tidak menginstal CLI, kami sarankan menggunakan skrip Python.

1. Buka sesi shell Python:

```
python3
```

2. Masukkan perintah berikut:

```
>>> import json
```

3. Masukkan perintah berikut:

```
>>> from airflow.models.connection import Connection
```

4. Buat variabel dalam sesi shell Anda untuk `extra` objek. Gantikan nilai sampel *YOUR_EXECUTION_ROLE_ARN* dengan peran eksekusi ARN, dan wilayah di *us-east-1* (sepertius-east-1).

```
>>> extra=json.dumps({'role_arn': 'YOUR_EXECUTION_ROLE_ARN', 'region_name': 'us-east-1'})
```

5. Buat objek koneksi. Gantikan nilai sampel `myconn` dengan nama koneksi Apache Airflow.

```
>>> myconn = Connection(
```

6. Gunakan tombol “tab” pada keyboard Anda untuk indentasi setiap pasangan kunci-nilai berikut di objek koneksi Anda. Gantikan nilai sampel di*red*.

   a. Tentukan jenis AWS koneksi:

```
... conn_id='aws',
```

b. Tentukan opsi database Apache Airflow:

```
... conn_type='mysql',
```

c. Tentukan URL UI Apache Airflow di Amazon MWAA:

```
... host='288888a0-50a0-888-9a88-1a111aaa0000.a1.us-
east-1.airflow.amazonaws.com/home',
```

d. Tentukan ID kunci AWS akses (nama pengguna) untuk masuk ke Amazon MWAA:

```
... login='YOUR_AWS_ACCESS_KEY_ID',
```

e. Tentukan kunci akses AWS rahasia (kata sandi) untuk masuk ke Amazon MWAA:

```
... password='YOUR_AWS_SECRET_ACCESS_KEY',
```

f. Tentukan variabel sesi `extra` shell:

```
... extra=extra
```

g. Tutup objek koneksi.

```
... )
```

7. Cetak string URI koneksi:

```
>>> myconn.get_uri()
```

Lihat string URI koneksi dalam respons:

```
'mysql://288888a0-50a0-888-9a88-1a111aaa0000.a1.us-east-1.airflow.amazonaws.com
%2Fhome?role_arn=arn%3Aaws%3Aiam%3A%3A001122332255%3Arole%2Fservice-role
%2FAmazonMWAA-MyAirflowEnvironment-iAaaaA&region_name=us-east-1'
```

Python script

Script Python berikut tidak memerlukan Apache Airflow CLI.

1. Salin isi contoh kode berikut dan simpan secara lokal sebagai`mwaa_connection.py`.

```
import urllib.parse

        conn_type = 'YOUR_DB_OPTION'
        host = 'YOUR_MWAA_AIRFLOW_UI_URL'
        port = 'YOUR_PORT'
        login = 'YOUR_AWS_ACCESS_KEY_ID'
        password = 'YOUR_AWS_SECRET_ACCESS_KEY'
        role_arn = urllib.parse.quote_plus('YOUR_EXECUTION_ROLE_ARN')
        region_name = 'us-east-1'

        conn_string = '{0}://{1}:{2}@{3}:{4}?
role_arn={5}&region_name={6}'.format(conn_type, login, password, host, port,
 role_arn, region_name)
        print(conn_string)
```

2. Gantikan placeholder di. *red*
3. Jalankan skrip berikut untuk menghasilkan string koneksi.

```
python3 mwaa_connection.py
```

Langkah empat: Tambahkan variabel di Secrets Manager

Bagian berikut menjelaskan cara membuat rahasia untuk variabel di Secrets Manager.

Untuk membuat rahasia

1. Buka konsol AWS Secrets Manager.
2. Pilih Simpan rahasia baru.
3. Pilih jenis rahasia lainnya.
4. Pada Tentukan key/value pasangan yang akan disimpan di panel rahasia ini, pilih Plaintext.
5. Tambahkan nilai variabel sebagai Plaintext dalam format berikut.

```
"YOUR_VARIABLE_VALUE"
```

Misalnya, untuk menentukan bilangan bulat:

```
14
```

Misalnya, untuk menentukan string:

```
"mystring"
```

6. Untuk kunci Enkripsi, pilih opsi AWS KMS kunci dari daftar dropdown.
7. Masukkan nama di bidang teks untuk nama Rahasia dalam format berikut.

```
airflow/variables/YOUR_VARIABLE_NAME
```

Contoh:

```
airflow/variables/test-variable
```

8. Pilih Berikutnya.
9. Pada halaman Konfigurasi rahasia, pada nama rahasia dan deskripsi panel, lakukan hal berikut.
   a. Untuk nama Rahasia, berikan nama untuk rahasia Anda.
   b. (Opsional) Untuk Deskripsi, berikan deskripsi untuk rahasia Anda.

   Pilih Berikutnya.
10. Pada rotasi Konfigurasi - opsional tinggalkan opsi default dan pilih Berikutnya.
11. Ulangi langkah-langkah ini di Secrets Manager untuk setiap variabel tambahan yang ingin Anda tambahkan.
12. Pada halaman Review, tinjau rahasia Anda, lalu pilih Store.

Langkah lima: Tambahkan koneksi di Secrets Manager

Bagian berikut menjelaskan cara membuat rahasia untuk URI string koneksi Anda di Secrets Manager.

Untuk membuat rahasia

1. Buka konsol AWS Secrets Manager.
2. Pilih Simpan rahasia baru.
3. Pilih jenis rahasia lainnya.
4. Pada Tentukan key/value pasangan yang akan disimpan di panel rahasia ini, pilih Plaintext.
5. Tambahkan string URI koneksi sebagai Plaintext dalam format berikut.

   ```
   YOUR_CONNECTION_URI_STRING
   ```

   Contoh:

   ```
   mysql://288888a0-50a0-888-9a88-1a111aaa0000.a1.us-east-1.airflow.amazonaws.com
   %2Fhome?role_arn=arn%3Aaws%3Aiam%3A%3A001122332255%3Arole%2Fservice-role
   %2FAmazonMWAA-MyAirflowEnvironment-iAaaaA&region_name=us-east-1
   ```

   > **Warning**
   >
   > Apache Airflow mem-parsing setiap nilai dalam string koneksi. Anda tidak boleh menggunakan tanda kutip tunggal atau ganda, atau mem-parsing koneksi sebagai string tunggal.

6. Untuk kunci Enkripsi, pilih opsi AWS KMS kunci dari daftar dropdown.
7. Masukkan nama di bidang teks untuk nama Rahasia dalam format berikut.

   ```
   airflow/connections/YOUR_CONNECTION_NAME
   ```

   Contoh:

   ```
   airflow/connections/myconn
   ```

8. Pilih Berikutnya.
9. Pada halaman Konfigurasi rahasia, pada nama rahasia dan deskripsi panel, lakukan hal berikut.

   a. Untuk nama Rahasia, berikan nama untuk rahasia Anda.

   b. (Opsional) Untuk Deskripsi, berikan deskripsi untuk rahasia Anda.

Pilih Berikutnya.

10. Pada rotasi Konfigurasi - opsional tinggalkan opsi default dan pilih Berikutnya.
11. Ulangi langkah-langkah ini di Secrets Manager untuk setiap variabel tambahan yang ingin Anda tambahkan.
12. Pada halaman Review, tinjau rahasia Anda, lalu pilih Store.

Kode sampel

- Pelajari cara menggunakan kunci rahasia untuk koneksi Apache Airflow `myconn` () di halaman ini menggunakan kode contoh di. Menggunakan kunci rahasia AWS Secrets Manager untuk koneksi Apache Airflow
- Pelajari cara menggunakan kunci rahasia untuk variabel Apache Airflow `test-variable` () di halaman ini menggunakan kode sampel di. Menggunakan kunci rahasia AWS Secrets Manager untuk variabel Apache Airflow

Sumber Daya

- Untuk informasi selengkapnya tentang mengonfigurasi rahasia Secrets Manager menggunakan konsol dan AWS CLI, lihat Buat rahasia di Panduan AWS Secrets Manager Pengguna.
- Gunakan skrip Python untuk memigrasikan volume besar variabel Apache Airflow dan koneksi ke Secrets Manager di Pindahkan koneksi dan variabel Apache Airflow Anda. AWS Secrets Manager

Apa selanjutnya?

- Pelajari cara membuat token untuk mengakses Apache Airflow UI di. Mengakses Apache Airflow

Mengelola lingkungan Amazon MWAA

Konsol Alur Kerja Terkelola Amazon untuk Apache Airflow berisi opsi bawaan untuk mengonfigurasi akses pribadi atau publik ke UI Apache Airflow. Ini juga berisi opsi bawaan untuk mengonfigurasi ukuran lingkungan, kapan harus menskalakan pekerja, dan opsi konfigurasi Apache Airflow yang dapat Anda gunakan untuk mengganti konfigurasi Apache Airflow yang biasanya hanya dapat diakses. `airflow.cfg` Bab ini menjelaskan cara menggunakan konfigurasi ini di konsol Amazon MWAA.

Topik

- Mengkonfigurasi kelas lingkungan Amazon MWAA
- Mengkonfigurasi penskalaan otomatis pekerja Amazon MWAA
- Mengkonfigurasi penskalaan otomatis server web Amazon MWAA
- Menggunakan opsi konfigurasi Apache Airflow di Amazon MWAA
- Perbarui lingkungan Amazon MWAA
- Mengubah versi Apache Airflow
- Menggunakan skrip startup dengan Amazon MWAA

Mengkonfigurasi kelas lingkungan Amazon MWAA

Kelas lingkungan yang Anda pilih untuk lingkungan Amazon MWAA menentukan ukuran AWS Fargate container yang AWS dikelola tempat Celery Executor berjalan, dan database metadata Amazon AWS Aurora PostgreSQL yang dikelola tempat penjadwal Apache Airflow membuat instance tugas. Topik ini menjelaskan setiap kelas lingkungan Amazon MWAA, dan cara memperbarui kelas lingkungan di konsol Amazon MWAA.

Bagian-bagian

- Kemampuan lingkungan
- Penjadwal Apache Airflow

Kemampuan lingkungan

Bagian berikut berisi tugas Apache Airflow bersamaan default, Random Access Memory (RAM), dan virtual centralized processing unit (vCPUs) untuk setiap kelas lingkungan. Tugas bersamaan yang

tercantum mengasumsikan bahwa konkurensi tugas tidak melebihi kapasitas pekerja Apache Airflow di lingkungan.

Dalam tabel berikut, kapasitas DAG mengacu pada definisi DAG, bukan eksekusi, dan mengasumsikan bahwa DAG Anda dinamis dalam satu file Python dan ditulis dengan praktik terbaik Apache Airflow.

Eksekusi tugas bergantung pada berapa banyak yang dijadwalkan secara bersamaan, dan mengasumsikan bahwa jumlah DAG run yang ditetapkan untuk memulai pada saat yang sama tidak melebihi default `max_dagruns_per_loop_to_schedule`, serta ukuran dan jumlah pekerja sebagaimana dirinci dalam topik ini.

mw1.micro

- Kapasitas hingga 25 DAG
- 3 tugas bersamaan (secara default)
- Komponen:
  - Server Web: 1 vCPU, RAM 3GB
  - Pekerja dan penjadwal: 1 vCPU, 3GB RAM
  - Basis data: 2 vCPU, 4GB RAM

> **Note**
>
> mw1.micro tidak mendukung auto-scaling.

mw1.small

- Kapasitas hingga 50 DAG
- 5 tugas bersamaan (secara default)
- Komponen:
  - Server web: 1 vCPU, RAM 2GB masing-masing
  - Pekerja: 1 vCPU, RAM 2GB masing-masing
  - Penjadwal: 1 vCPU, RAM 2GB masing-masing
  - Basis data: 2 vCPU, 4GB RAM

mw1.medium

- Kapasitas hingga 250 DAG
- 10 tugas bersamaan (secara default)
- Komponen:
  - Server web: 1 vCPU 2GB RAM masing-masing
  - Pekerja: 2 vCPU 4GB RAM masing-masing
  - Penjadwal: 2 vCPU 4GB RAM masing-masing
  - Basis data: 2 vCPU 8GB RAM

mw1.large

- Kapasitas hingga 1000 DAG
- 20 tugas bersamaan (secara default)
- Komponen:
  - Server web: 2 vCPU 4GB RAM masing-masing
  - Pekerja: 4 vCPU 8GB RAM masing-masing
  - Penjadwal: 4 vCPU 8GB RAM masing-masing
  - Basis data: 2 vCPU 8GB RAM

mw1.xlarge

- Kapasitas hingga 2000 DAG
- 40 tugas bersamaan (secara default)
- Komponen:
  - Server web: 4 vCPU 12GB RAM masing-masing
  - Pekerja: 8 vCPU 24GB RAM masing-masing
  - Penjadwal: 8 vCPU 24GB RAM masing-masing
  - Basis Data: 4 vCPU 32GB RAM

mw1.2xlarge

- Kapasitas hingga 4000 DAG

- 80 tugas bersamaan (secara default)
- Komponen:
  - Server web: 8 vCPU 24GB RAM masing-masing
  - Pekerja: 16 vCPU 48GB RAM masing-masing
  - Penjadwal: 16 vCPU 48GB RAM masing-masing
  - Basis data: 8 vCPU 64GB RAM

Anda dapat menggunakan `celery.worker_autoscale` untuk meningkatkan tugas per pekerja. Untuk informasi lebih lanjut, lihatthe section called “Contoh kasus penggunaan kinerja tinggi”.

Penjadwal Apache Airflow

Bagian berikut berisi opsi penjadwal Apache Airflow yang tersedia di Amazon MWAA, dan bagaimana jumlah penjadwal memengaruhi jumlah pemicu.

Di Apache Airflow, pemicu mengelola tugas yang ditolaknya sampai kondisi tertentu yang ditentukan menggunakan pemicu telah terpenuhi. Di Amazon MWAA, pemicu berjalan bersama penjadwal pada tugas Fargate yang sama. Meningkatkan jumlah penjadwal juga meningkatkan jumlah pemicu yang tersedia, mengoptimalkan cara lingkungan mengelola tugas yang ditangguhkan. Ini memastikan penanganan tugas yang efisien, segera menjadwalkannya untuk dijalankan ketika kondisi terpenuhi.

Apache Airflow v3

- v3 - Untuk lingkungan yang lebih besar dari mw1.micro, menerima nilai dari ke. 2 5 Default untuk 2 semua ukuran lingkungan kecuali mw1.micro, yang defaultnya. 1

Apache Airflow v2

- v2 - Untuk lingkungan yang lebih besar dari mw1.micro, menerima nilai dari ke. 2 5 Default untuk 2 semua ukuran lingkungan kecuali mw1.micro, yang defaultnya. 1

Mengkonfigurasi penskalaan otomatis pekerja Amazon MWAA

Mekanisme penskalaan otomatis secara otomatis meningkatkan jumlah pekerja Apache Airflow sebagai respons terhadap tugas yang sedang berjalan dan antri di Alur Kerja Terkelola Amazon Anda untuk lingkungan Apache Airflow dan membuang pekerja tambahan saat tidak ada lagi tugas yang

diantri atau dijalankan. Topik ini menjelaskan bagaimana Anda dapat mengonfigurasi penskalaan otomatis dengan menentukan jumlah maksimum pekerja Apache Airflow yang berjalan di lingkungan Anda menggunakan konsol Amazon MWAA.

> **Note**
>
> Amazon MWAA menggunakan metrik Apache Airflow untuk menentukan kapan pekerja Celery Executor tambahan diperlukan, dan jika diperlukan meningkatkan jumlah pekerja Fargate hingga nilai yang ditentukan oleh. `max-workers` Ketika pekerja tambahan menyelesaikan pekerjaan dan beban kerja berkurang, Amazon MWAA menghapusnya, sehingga menurunkan skala kembali ke nilai yang ditetapkan oleh. `min-workers` Jika pekerja mengambil tugas baru saat menurunkan skala, Amazon MWAA menyimpan sumber daya Fargate dan tidak menghapus pekerja. Untuk informasi selengkapnya, lihat Cara kerja penskalaan otomatis Amazon MWAA.

Bagian-bagian

- Cara kerja penskalaan pekerja
- Menggunakan konsol Amazon MWAA
- Contoh kasus penggunaan kinerja tinggi
- Memecahkan masalah tugas yang macet dalam status berjalan
- Apa selanjutnya?

Cara kerja penskalaan pekerja

Amazon MWAA menggunakan `RunningTasks` dan `QueuedTasks` metrik, di mana (tugas yang menjalankan+tugas diantrian)/(tugas per pekerja) = (pekerja wajib). Jika jumlah pekerja yang dibutuhkan lebih besar dari jumlah pekerja saat ini, Amazon MWAA akan menambahkan wadah pekerja Fargate ke nilai tersebut, hingga nilai maksimum yang ditentukan oleh. `max-workers`

Saat beban kerja berkurang dan jumlah `QueuedTasks` metrik berkurang, Amazon MWAA meminta Fargate untuk menurunkan pekerja untuk lingkungan. `RunningTasks` Setiap pekerja yang masih menyelesaikan pekerjaan tetap terlindungi selama downscaling sampai mereka menyelesaikan pekerjaan mereka. Tergantung pada beban kerja, tugas mungkin antri sementara pekerja menurunkan skala.

Menggunakan konsol Amazon MWAA

Anda dapat memilih jumlah maksimum pekerja yang dapat berjalan di lingkungan Anda secara bersamaan di konsol Amazon MWAA. Secara default, Anda dapat menentukan nilai maksimum hingga 25.

Untuk mengkonfigurasi jumlah pekerja

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.
3. Pilih Edit.
4. Pilih Berikutnya.
5. Pada panel kelas Lingkungan, masukkan nilai dalam Jumlah pekerja maksimum.
6. Pilih Simpan.

> **Note**
>
> Diperlukan beberapa menit sebelum perubahan berlaku pada lingkungan Anda.

Contoh kasus penggunaan kinerja tinggi

Bagian berikut menjelaskan jenis konfigurasi yang dapat Anda gunakan untuk mengaktifkan kinerja tinggi dan paralelisme pada lingkungan.

On-premise Apache Airflow

Biasanya, di platform Apache Airflow lokal, Anda mengonfigurasi paralelisme tugas, penskalaan otomatis, dan pengaturan konkurensi dalam file Anda: `airflow.cfg`

- `core.parallelism`— Jumlah maksimum instance tugas yang dapat berjalan secara bersamaan per penjadwal.
- `core.dag_concurrency`— Konkurensi maksimum untuk DAG (bukan pekerja).
- `celery.worker_autoscale`— Jumlah tugas maksimum dan minimum yang dapat dijalankan secara bersamaan pada pekerja mana pun.

Misalnya, jika `core.parallelism` disetel ke `100` dan `core.dag_concurrency` disetel ke7, Anda hanya dapat menjalankan total `14` tugas secara bersamaan jika Anda memiliki 2 DAG. Mengingat, setiap DAG diatur untuk menjalankan hanya tujuh tugas secara bersamaan (in`core.dag_concurrency`), meskipun paralelisme keseluruhan diatur ke `100` (in). `core.parallelism`

> **Note**
>
> `core.dag_concurrency`tidak tersedia di Apache Airflow v3.

Di lingkungan Amazon MWAA

Di lingkungan Amazon MWAA, Anda dapat mengonfigurasi pengaturan ini secara langsung di konsol Amazon MWAA menggunakan Menggunakan opsi konfigurasi Apache Airflow di Amazon MWAAMengkonfigurasi kelas lingkungan Amazon MWAA, dan mekanisme penskalaan otomatis jumlah pekerja maksimum. Meskipun tidak `core.dag_concurrency` tersedia dalam daftar drop-down sebagai opsi konfigurasi Apache Airflow di konsol Amazon MWAA, Anda dapat menambahkannya sebagai opsi konfigurasi Apache Airflow khusus.

Katakanlah, ketika Anda membuat lingkungan Anda, Anda memilih pengaturan berikut:

1. Kelas lingkungan mw1.small yang mengontrol jumlah maksimum tugas bersamaan yang dapat dijalankan setiap pekerja secara default dan vCPU kontainer.
2. Pengaturan default `10` pekerja dalam jumlah pekerja Maksimum.
3. Opsi konfigurasi Apache Airflow `celery.worker_autoscale` untuk tugas per pekerja`5,5`.

Ini berarti Anda dapat menjalankan 50 tugas bersamaan di lingkungan Anda. Setiap tugas di atas 50 diantrian, dan menunggu tugas yang sedang berjalan selesai.

Jalankan lebih banyak tugas bersamaan. Anda dapat memodifikasi lingkungan untuk menjalankan lebih banyak tugas secara bersamaan menggunakan konfigurasi berikut:

1. Tingkatkan jumlah maksimum tugas bersamaan yang dapat dijalankan setiap pekerja secara default dan vCPU kontainer dengan memilih kelas lingkungan `mw1.medium` (10 tugas bersamaan secara default).
2. Tambahkan `celery.worker_autoscale` sebagai opsi konfigurasi Apache Airflow.

3. Tingkatkan jumlah pekerja maksimum. Dalam contoh ini, meningkatkan pekerja maksimum dari `10` `20` menggandakan jumlah tugas bersamaan yang dapat dijalankan lingkungan.

Tentukan Pekerja minimum. Anda juga dapat menentukan jumlah minimum dan maksimum pekerja Apache Airflow yang berjalan di lingkungan Anda menggunakan AWS Command Line Interface ().AWS CLI Contoh:

```
aws mwaa update-environment --max-workers 10 --min-workers 10 --
name YOUR_ENVIRONMENT_NAME
```

Untuk mempelajari lebih lanjut, lihat perintah update-environment di. AWS CLI

Memecahkan masalah tugas yang macet dalam status berjalan

Dalam kasus yang jarang terjadi, Apache Airflow mungkin berpikir ada tugas yang masih berjalan. Untuk mengatasi masalah ini, Anda perlu menghapus tugas yang terdampar di UI Apache Airflow Anda. Untuk informasi selengkapnya, lihat topik Memecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow pemecahan masalah.

Apa selanjutnya?

- Pelajari lebih lanjut tentang praktik terbaik yang kami rekomendasikan untuk menyesuaikan kinerja lingkungan AndaPenyetelan kinerja untuk Apache Airflow di Amazon MWAA.

Mengkonfigurasi penskalaan otomatis server web Amazon MWAA

Untuk lingkungan yang menjalankan Apache Airflow v2.2.2 dan yang lebih baru, Amazon MWAA secara dinamis menskalakan server web Anda untuk menangani beban kerja yang berfluktuasi, yang pada gilirannya mencegah masalah kinerja selama beban puncak. Dengan secara otomatis menskalakan jumlah server web berdasarkan pemanfaatan CPU dan jumlah koneksi aktif, Amazon MWAA memastikan bahwa lingkungan Apache Airflow Anda dapat dengan mulus mengakomodasi peningkatan permintaan, baik dari permintaan REST API, penggunaan CLI, atau pengguna antarmuka pengguna Apache Airflow yang lebih bersamaan.

Bagian-bagian

- Cara kerja penskalaan server web
- Menggunakan konsol Amazon MWAA

Cara kerja penskalaan server web

Amazon MWAA menggunakan metrik kontainer `CPUUtilization`,, dan metrik penyeimbang beban `ActiveConnectionCount`, untuk menentukan apakah penskalaan server web diperlukan berdasarkan jumlah lalu lintas. Jika `CPUUtilization` lebih besar dari 70 atau `ActiveConnectionCount` lebih besar dari 15, Amazon MWAA akan menambahkan wadah server web Fargate tambahan hingga nilai maksimum yang ditentukan oleh. `MaxWebservers`

Saat lalu lintas menurun dan `ActiveConnectionCount` nilai `CPUUtilization` dan berkurang, Amazon MWAA meminta Fargate untuk menurunkan kontainer server web untuk lingkungan ke nilai minimum yang ditetapkan oleh. `MinimumWebservers`

Menggunakan konsol Amazon MWAA

Anda dapat memilih jumlah server web yang dapat berjalan di lingkungan Anda secara bersamaan di konsol Amazon MWAA. Secara default, jumlah minimum server web adalah dua, dan jumlah maksimum server web adalah lima.

Untuk mengkonfigurasi jumlah server web

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.
3. Pilih Edit.
4. Pilih Berikutnya.
5. Pada panel kelas Lingkungan, masukkan nilai dalam jumlah server web Maksimum.
6. Selanjutnya, masukkan nilai dalam jumlah server web Minimum.
7. Pilih Simpan.

> **Note**
>
> Ini bisa memakan waktu beberapa menit sebelum perubahan berlaku pada lingkungan Anda.

Menggunakan opsi konfigurasi Apache Airflow di Amazon MWAA

Opsi konfigurasi Apache Airflow dapat dilampirkan ke Alur Kerja Terkelola Amazon Anda untuk lingkungan Apache Airflow sebagai variabel lingkungan. Anda dapat memilih dari daftar dropdown

yang disarankan atau menentukan opsi konfigurasi khusus untuk versi Apache Airflow Anda di konsol Amazon MWAA. Topik ini menjelaskan opsi konfigurasi Apache Airflow yang tersedia dan cara menggunakan opsi ini untuk mengganti pengaturan konfigurasi Apache Airflow di lingkungan Anda.

Daftar Isi

- Prasyarat
- Cara kerjanya
- Menggunakan opsi konfigurasi untuk memuat plugin
- Ikhtisar opsi konfigurasi
  - Opsi konfigurasi Apache Airflow
  - Referensi Apache Airflow
  - Menggunakan konsol Amazon MWAA
- Referensi konfigurasi
  - Konfigurasi email
  - Konfigurasi tugas
  - Konfigurasi penjadwal
  - Konfigurasi pekerja
  - Konfigurasi server web
  - Konfigurasi pemicu
- Konfigurasi yang tidak didukung
- Contoh dan kode sampel
  - Contoh DAG
  - Contoh pengaturan pemberitahuan email
- Apa selanjutnya?

Prasyarat

Anda akan memerlukan yang berikut ini sebelum Anda dapat menyelesaikan langkah-langkah di halaman ini.

- Izin — Anda Akun AWS harus telah diberikan akses oleh administrator Anda ke kebijakan kontrol AmazonMWAAFullConsoleAccessakses untuk lingkungan Anda. Selain itu, lingkungan Amazon

MWAA Anda harus diizinkan oleh peran eksekusi Anda untuk mengakses AWS sumber daya yang digunakan oleh lingkungan Anda.

- Akses — Jika Anda memerlukan akses ke repositori publik untuk menginstal dependensi langsung di server web, lingkungan Anda harus dikonfigurasi dengan akses server web jaringan publik. Untuk informasi lebih lanjut, lihatthe section called “Mode akses Apache Airflow”.
- Konfigurasi Amazon S3 - Bucket Amazon S3 yang digunakan untuk menyimpan DAG, plugin `plugins.zip` khusus, dan `requirements.txt` dependensi Python harus dikonfigurasi dengan Akses Publik Diblokir dan Diaktifkan Versi.

Cara kerjanya

Saat Anda membuat lingkungan, Amazon MWAA melampirkan pengaturan konfigurasi yang Anda tentukan di konsol Amazon MWAA dalam opsi konfigurasi Airflow sebagai variabel lingkungan ke wadah untuk lingkungan Anda. AWS Fargate Jika Anda menggunakan setelan dengan nama yang sama`airflow.cfg`, opsi yang Anda tentukan di konsol Amazon MWAA akan mengganti nilainya. `airflow.cfg`

Meskipun kami tidak mengekspos `airflow.cfg` di Apache Airflow UI dari lingkungan Amazon MWAA secara default, Anda dapat mengubah opsi konfigurasi Apache Airflow langsung di konsol Amazon MWAA, termasuk pengaturan untuk mengekspos konfigurasi. `webserver.expose_config`

Menggunakan opsi konfigurasi untuk memuat plugin

Secara default di Apache Airflow v2 dan yang lebih baru, plugin dikonfigurasi untuk dimuat “malas” menggunakan pengaturan. `core.lazy_load_plugins : True` Jika Anda menggunakan plugin khusus, Anda harus menambahkan `core.lazy_load_plugins : False` sebagai opsi konfigurasi Apache Airflow untuk memuat plugin di awal setiap proses Airflow untuk mengganti pengaturan default.

Ikhtisar opsi konfigurasi

Saat Anda menambahkan konfigurasi di konsol Amazon MWAA, Amazon MWAA menulis konfigurasi sebagai variabel lingkungan.

- Opsi yang terdaftar. Anda dapat memilih dari salah satu pengaturan konfigurasi yang tersedia untuk versi Apache Airflow Anda di daftar dropdown. Misalnya, `dag_concurrency`

: 16. Pengaturan konfigurasi diterjemahkan ke wadah Fargate lingkungan Anda sebagai `AIRFLOW__CORE__DAG_CONCURRENCY : 16`

- Opsi kustom. Anda juga dapat menentukan opsi konfigurasi Airflow yang tidak terdaftar untuk versi Apache Airflow Anda di daftar dropdown. Misalnya, `foo.user` : `YOUR_USER_NAME`. Pengaturan konfigurasi diterjemahkan ke wadah Fargate lingkungan Anda sebagai `AIRFLOW__FOO__USER : YOUR_USER_NAME`

Opsi konfigurasi Apache Airflow

Gambar berikut menggambarkan di mana Anda dapat menyesuaikan opsi konfigurasi Apache Airflow di konsol Amazon MWAA.

Referensi Apache Airflow

Untuk daftar opsi konfigurasi yang didukung oleh Apache Airflow, lihat Referensi Konfigurasi dalam panduan referensi Apache Airflow. Untuk mengakses opsi untuk versi Apache Airflow yang Anda jalankan di Amazon MWAA, pilih versi dari daftar drop-down.

Menggunakan konsol Amazon MWAA

Prosedur berikut memandu Anda melalui langkah-langkah menambahkan opsi konfigurasi Aliran Udara ke lingkungan Anda.

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.
3. Pilih Edit.
4. Pilih Berikutnya.
5. Pilih Tambahkan konfigurasi khusus di panel Opsi konfigurasi Aliran Udara.
6. Pilih konfigurasi dari daftar dropdown dan masukkan nilai, atau masukkan konfigurasi khusus dan masukkan nilai.
7. Pilih Tambahkan konfigurasi khusus untuk setiap konfigurasi yang ingin Anda tambahkan.
8. Pilih Simpan.

Referensi konfigurasi

Bagian berikut berisi daftar konfigurasi Apache Airflow yang tersedia di daftar dropdown di konsol Amazon MWAA.

Konfigurasi email

Daftar berikut menampilkan opsi konfigurasi pemberitahuan email Airflow yang tersedia di Amazon MWAA untuk Apache Airflow v2 dan v3.

Kami merekomendasikan menggunakan port 587 untuk lalu lintas SMTP. Secara default, AWS memblokir lalu lintas SMTP keluar pada port 25 dari semua instans Amazon EC2. Jika Anda ingin mengirim lalu lintas keluar pada port 25, Anda dapat meminta pembatasan ini dihapus.

Opsi konfigurasi aliran udara	Deskripsi	Nilai contoh
email.email_backend	Utilitas Apache Airflow digunakan untuk pemberitahuan email di email_backend.	airflow.utils.email.send_email_smtp
smtp.smtp_host	Nama server keluar yang digunakan untuk alamat email di smtp_host.	localhost
smtp.smtp_starttls	Transport Layer Security (TLS) digunakan untuk mengenkripsi email melalui internet di smtp_starttls.	False
smtp.smtp_ssl	Secure Sockets Layer (SSL) digunakan untuk menghubungkan server dan klien email di smtp_ssl.	True
smtp.smtp_port		587

Opsi konfigurasi aliran udara	Deskripsi	Nilai contoh
	Port Transmission Control Protocol (TCP) yang ditunjuk ke server di smtp_port.	
smtp.smtp_mail_from	Alamat email keluar di smtp_mail_from.	myemail@domain.com

Konfigurasi tugas

Daftar berikut menampilkan konfigurasi yang tersedia di daftar dropdown untuk tugas Airflow di Amazon MWAA untuk Apache Airflow v2 dan v3.

Opsi konfigurasi aliran udara	Deskripsi	Nilai contoh
core.default_task_retries	Berapa kali untuk mencoba kembali tugas Apache Airflow di default_task_retries.	3
inti.paralelisme	Jumlah maksimum instance tugas yang dapat berjalan secara bersamaan di seluruh lingkungan secara paralel (paralelisme).	40

Konfigurasi penjadwal

Daftar berikut menampilkan konfigurasi penjadwal Apache Airflow yang tersedia di daftar dropdown di Amazon MWAA untuk Apache Airflow v2 dan v3.

Opsi konfigurasi aliran udara	Deskripsi	Nilai contoh
scheduler.catchup_by_default		False

Opsi konfigurasi aliran udara	Deskripsi	Nilai contoh
	Memberitahu scheduler untuk membuat DAG run untuk “catch up” ke interval waktu tertentu di catchup_by_default.	
scheduler.scheduler_zombie_task_threshold **Note** Tidak tersedia di Apache Airflow v3.	Memberi tahu penjadwal apakah akan menandai instance tugas sebagai gagal dan menjadwal ulang tugas di scheduler_zombie_task_threshold.	300

Konfigurasi pekerja

Daftar berikut menampilkan konfigurasi pekerja Airflow yang tersedia di daftar dropdown di Amazon MWAA untuk Apache Airflow v2 dan v3.

Opsi konfigurasi aliran udara	Deskripsi	Nilai contoh
seledry.worker_autoscale	Jumlah tugas maksimum dan minimum yang dapat berjalan secara bersamaan pada pekerja mana pun yang menggunakan Celery Executor di worker_autoscale. Nilai harus dipisahkan koma dalam urutan sebagai berikut:. `max_concurrency,min_concurrency`	16,12

Konfigurasi server web

Daftar berikut menampilkan konfigurasi server web Apache Airflow yang tersedia di daftar dropdown di Amazon MWAA untuk Apache Airflow v2 dan v3.

Opsi konfigurasi aliran udara	Deskripsi	Nilai contoh
webserver.default_ui_timezone **Note** Tidak tersedia di Apache Airflow v3.	Pengaturan datetime Apache Airflow UI default di default_ui_timezone. **Note** Pengaturan `default_ui_timezone` opsi tidak mengubah zona waktu di mana DAG Anda dijadwalkan untuk berjalan. Untuk mengubah zona waktu DAG Anda, Anda dapat menggunakan plugin khusus. Untuk informasi lebih lanjut, lihatthe section called "Mengubah zona waktu DAG".	America/New_York

Konfigurasi pemicu

Daftar berikut menampilkan konfigurasi pemicu Apache Airflow yang tersedia di Amazon MWAA untuk Apache Airflow v2 dan v3.

Opsi konfigurasi aliran udara	Deskripsi	Nilai contoh

Opsi konfigurasi aliran udara	Deskripsi	Nilai contoh
mwaa.triggerer_enabled	Digunakan untuk mengaktifkan dan menonaktifkan pemicu di Amazon MWAA. Secara default, nilai ini diatur ke `True`. Jika disetel ke`False`, Amazon MWAA tidak akan memulai proses pemicu apa pun pada penjadwal.	True
triggerer.default_capacity (dalam v2) triggerer.capacity (dalam v3)	Mendefinisikan pemicu angka setiap pemicu dapat berjalan secara paralel. Di Amazon MWAA, kapasitas ini diatur per setiap pemicu dan per setiap penjadwal karena kedua komponen berjalan berdampingan satu sama lain. Default per scheduler diatur ke`60`,,`125`, `250`,`500`, dan `1000` untuk instance kecil, sedang dan besar, xlarge, dan 2xlarge, masing-masing.	125

Konfigurasi yang tidak didukung

Opsi konfigurasi Apache Airflow berikut tidak tersedia di Amazon MWAA. Anda tidak dapat menyetel atau mengganti opsi ini menggunakan konsol atau API Amazon MWAA.

Opsi konfigurasi aliran udara	Versi Apache Airflow	Nilai default	Alasan
core.multi_team	3.2	False	Amazon MWAA saat ini tidak mendukung mode multi-tim.

Opsi konfigurasi aliran udara	Versi Apache Airflow	Nilai default	Alasan
			Mengaktifkan fitur ini tidak kompatibel dengan otentikasi Amazon MWAA, manajemen rahasia tingkat lingkungan CeleryExecutor, dan lingkungan.
triggerer.queues_enabled	3.2	False	Amazon MWAA tidak mendukung penetapan antrian pemicu. Mengaktifkan opsi ini menyebabkan tugas yang ditangguhkan menggantung tanpa batas waktu.

Contoh dan kode sampel

Contoh DAG

Anda dapat menggunakan DAG berikut untuk mencetak opsi konfigurasi `email_backend` Apache Airflow Anda. Untuk menjalankan sebagai respons terhadap peristiwa Amazon MWAA, salin kode ke folder DAG lingkungan Anda di bucket penyimpanan Amazon S3 Anda.

```
from airflow.decorators import dag
    from datetime import datetime

    def print_var(**kwargs):
    email_backend = kwargs['conf'].get(section='email', key='email_backend')
    print("email_backend")
    return email_backend

    @dag(
    dag_id="print_env_variable_example",
    schedule_interval=None,
    start_date=datetime(yyyy, m, d),
    catchup=False,
    )
    def print_variable_dag():
    email_backend_test = PythonOperator(
    task_id="email_backend_test",
    python_callable=print_var,
```

```
    provide_context=True
    )

    print_variable_test = print_variable_dag()
```

Contoh pengaturan pemberitahuan email

Opsi konfigurasi Apache Airflow berikut dapat digunakan untuk akun email menggunakan Gmail.com kata sandi aplikasi. Untuk informasi selengkapnya, lihat Masuk menggunakan kata sandi aplikasi di panduan referensi Bantuan Gmail.

Apa selanjutnya?

- Pelajari cara mengunggah folder DAG Anda ke bucket Amazon S3 Anda. Menambahkan atau memperbarui DAGs

Perbarui lingkungan Amazon MWAA

> **Note**
>
> Pembaruan Amazon MWAA yang anggun belum didukung di wilayah Kanada Barat (Calgary) dan Asia Pasifik (Malaysia).

Pembaruan lingkungan Amazon MWAA menerapkan perubahan terbaru dan tambalan keamanan. Anda juga dapat mengedit konfigurasi yang ada dan meningkatkan versi Apache Airflow. Panduan ini menjelaskan langkah-langkah untuk memperbarui lingkungan Amazon MWAA.

Daftar Isi

- Sebelum Anda mulai
- Strategi penggantian pekerja
- Perbarui sumber daya lingkungan
- Perbarui lingkungan
  - Langkah satu: Tentukan detail
  - Langkah kedua: Konfigurasikan pengaturan lanjutan

- Langkah ketiga: Tinjau dan perbarui

Sebelum Anda mulai

- Jaringan VPC yang Anda tentukan untuk lingkungan Anda tidak dapat dimodifikasi setelah lingkungan dibuat.
- Anda memerlukan bucket Amazon S3 yang dikonfigurasi untuk Memblokir semua akses publik, dengan Pembuatan Versi Bucket diaktifkan.
- Anda memerlukan izin Akun AWS dengan untuk menggunakan Amazon MWAA, dan izin di AWS Identity and Access Management (IAM) untuk membuat peran IAM. Jika Anda memilih mode akses jaringan Pribadi untuk server web Apache Airflow, yang membatasi akses Apache Airflow dalam VPC Amazon Anda, Anda memerlukan izin di IAM untuk membuat titik akhir Amazon VPC.
- Untuk mengaktifkan pembaruan lingkungan Anggun, Anda perlu memutakhirkan Ke Apache Airflow versi 2.4.3 atau lebih tinggi. Untuk memutakhirkan versi Airflow, lihat. Mengubah versi Apache Airflow

Strategi penggantian pekerja

Anda dapat memilih strategi penggantian pekerja untuk mengontrol cara Amazon MWAA menangani pekerja aktif selama pembaruan lingkungan. Anda dapat memilih salah satu strategi berikut:

Pembaruan paksa

Pembaruan paksa adalah strategi penggantian pekerja default. Pembaruan paksa segera menghentikan semua pekerja aktif, sehingga menyebabkan gagalnya tugas yang sedang berjalan selama pembaruan.

Pembaruan yang anggun

Pembaruan mudah memungkinkan pekerja terus menjalankan tugas hingga 12 jam sebelum dimatikan. Ini mencegah tugas gagal karena gangguan pembaruan, selama mereka selesai di bawah 12 jam. Tugas baru dialihkan ke pekerja yang sudah diperbarui.

Untuk mengaktifkan pembaruan Anggun pada lingkungan yang ada, Anda harus menyelesaikannya Pembaruan paksa dan memastikan lingkungan berada di Apache Airflow versi 2.4.3 atau lebih tinggi.

> **Note**
>
> Jika Anda melakukan pembaruan saat lingkungan Anda dalam `MAINTENANCE` status, strategi penggantian pekerja untuk setiap pembaruan lingkungan yang sedang berlangsung akan beralih dari `GRACEFUL` ke`FORCED`. Pembaruan Anda dilakukan setelah pemeliharaan selesai.

Perbarui sumber daya lingkungan

Pembaruan lingkungan Amazon MWAA menggunakan konfigurasi lingkungan yang ada secara default. Untuk memperbarui lingkungan tanpa mengubah konfigurasi Anda saat ini:

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Dari daftar Lingkungan, pilih lingkungan yang ingin Anda perbarui.
3. Pada halaman lingkungan, pilih Edit untuk mengedit lingkungan.
4. Pilih Berikutnya sampai Anda berada di halaman Review dan save.
5. Pada halaman Tinjau dan simpan, tinjau perubahan Anda, lalu pilih Simpan.

Perbarui lingkungan

Bagian berikut menjelaskan langkah-langkah untuk memperbarui lingkungan Amazon MWAA.

Langkah satu: Tentukan detail

Untuk menentukan detail untuk lingkungan

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Dari daftar Lingkungan, pilih lingkungan yang ingin Anda perbarui.
3. Pada halaman lingkungan, pilih Edit untuk mengedit lingkungan.
4. Di bagian Detail lingkungan, untuk versi Airflow, pilih nomor versi Apache Airflow baru yang ingin Anda tingkatkan lingkungan dari daftar dropdown.

> **Note**
>
> Sebelum Anda meningkatkan, pastikan bahwa DAG dan sumber daya alur kerja lainnya kompatibel dengan versi Apache Airflow yang baru. Untuk informasi lebih lanjut, lihatMengubah versi Apache Airflow.

5. Di bawah kode DAG di Amazon S3 tentukan yang berikut:
   a. Ember S3. Pilih Jelajahi S3 dan pilih bucket Amazon S3 Anda, atau masukkan URI Amazon S3.
   b. Folder DAGs. Pilih Jelajahi S3 dan pilih `dags` folder di bucket Amazon S3 Anda, atau masukkan URI Amazon S3.
   c. File plugin - opsional. Pilih Jelajahi S3 dan pilih `plugins.zip` file di bucket Amazon S3 Anda, atau masukkan URI Amazon S3.
   d. File persyaratan - opsional. Pilih Jelajahi S3 dan pilih `requirements.txt` file di bucket Amazon S3 Anda, atau masukkan URI Amazon S3.
   e. File skrip startup - opsional, Pilih Browse S3 dan pilih file skrip di bucket Amazon S3 Anda, atau masukkan URI Amazon S3.
6. Pilih Berikutnya.

Langkah kedua: Konfigurasikan pengaturan lanjutan

Untuk mengonfigurasi pengaturan lanjutan

1. Di bawah akses server web, pilih mode akses Apache Airflow pilihan Anda:
   a. Jaringan pribadi. Ini membatasi akses UI Apache Airflow ke pengguna dalam VPC Amazon Anda yang telah diberikan akses ke kebijakan IAM untuk lingkungan Anda. Anda memerlukan izin untuk membuat titik akhir Amazon VPC untuk langkah ini.

> **Note**
>
> Pilih opsi Jaringan pribadi jika UI Apache Airflow Anda hanya diakses dalam jaringan perusahaan, dan Anda tidak memerlukan akses ke repositori publik untuk instalasi persyaratan server web. Jika Anda memilih opsi mode akses ini, Anda perlu membuat mekanisme untuk mengakses server web Apache Airflow Anda di Amazon

VPC Anda. Untuk informasi lebih lanjut, lihatMengakses titik akhir VPC untuk server web Apache Airflow Anda (akses jaringan pribadi).

b. Jaringan publik. Hal ini memungkinkan Apache Airflow UI untuk diakses melalui internet oleh pengguna yang diberikan akses ke kebijakan IAM untuk lingkungan Anda.

c. Baik akses jaringan publik maupun pribadi. Tersedia untuk Apache Airflow versi 3.2.1 dan yang lebih baru. Hal ini memungkinkan Apache Airflow UI untuk diakses melalui internet, sementara pekerja berkomunikasi dengan server web melalui titik akhir VPC pribadi. Pilih opsi ini jika VPC Amazon yang menghosting lingkungan Anda tidak memiliki akses internet.

2. Di bawah Grup keamanan, pilih grup keamanan yang digunakan untuk mengamankan VPC Amazon Anda:

a. Secara default, Amazon MWAA membuat grup keamanan di VPC Amazon Anda dengan aturan masuk dan keluar tertentu di Buat grup keamanan baru.

b. Opsional. Hapus centang kotak di Buat grup keamanan baru untuk memilih hingga 5 grup keamanan.

> **Note**
>
> Grup keamanan Amazon VPC yang ada harus dikonfigurasi dengan aturan masuk dan keluar tertentu untuk memungkinkan lalu lintas jaringan. Untuk mempelajari lebih lanjut, lihatKeamanan di VPC Anda di Amazon MWAA.

3. Di bawah kelas Lingkungan, pilih kelas lingkungan.

   Sebaiknya pilih ukuran terkecil yang diperlukan untuk mendukung beban kerja Anda. Anda dapat mengubah kelas lingkungan kapan saja.

4. Untuk jumlah pekerja Maksimum, tentukan jumlah maksimum pekerja Apache Airflow yang akan dijalankan di lingkungan.

   Untuk informasi lebih lanjut, lihatContoh kasus penggunaan kinerja tinggi.

5. Tentukan jumlah server web maksimum dan Jumlah server web minimum untuk mengonfigurasi cara Amazon MWAA menskalakan server web Apache Airflow di lingkungan Anda.

   Untuk informasi lebih lanjut tentang penskalaan otomatis server web, lihat. the section called “Mengkonfigurasi penskalaan otomatis server web”

6. Di bawah Enkripsi, pilih opsi enkripsi data:

a. Secara default, Amazon MWAA menggunakan kunci yang AWS dimiliki untuk mengenkripsi data Anda.

b. Opsional. Pilih Sesuaikan pengaturan enkripsi (lanjutan) untuk memilih AWS KMS kunci yang berbeda. Jika Anda memilih untuk menentukan Customer-managed kunci dalam langkah ini, Anda harus menentukan ID AWS KMS kunci atau ARN. AWS KMS alias dan kunci multi-wilayah tidak didukung oleh Amazon MWAA. Jika Anda menentukan kunci Amazon S3 untuk enkripsi sisi server di bucket Amazon S3, Anda harus menentukan kunci yang sama untuk lingkungan Amazon MWAA Anda.

> **Note**
>
> Anda harus memiliki izin untuk kunci untuk memilihnya di konsol Amazon MWAA. Anda juga harus memberikan izin kepada Amazon MWAA untuk menggunakan kunci dengan melampirkan kebijakan yang dijelaskan di dalamnya. Lampirkan kebijakan kunci

7. Direkomendasikan. Di bawah Monitoring, pilih satu atau beberapa kategori log untuk konfigurasi pencatatan Aliran Udara untuk mengirim log Apache Airflow ke Log: CloudWatch

   a. Log tugas aliran udara. Pilih jenis log tugas Apache Airflow untuk dikirim CloudWatch ke tingkat Log di Log.

   b. Log server web aliran udara. Pilih jenis log server web Apache Airflow untuk dikirim CloudWatch ke tingkat Log di Log.

   c. Log penjadwal aliran udara. Pilih jenis log penjadwal Apache Airflow untuk dikirim CloudWatch ke tingkat Log di Log.

   d. Log pekerja aliran udara. Pilih jenis log pekerja Apache Airflow untuk dikirim CloudWatch ke tingkat Log di Log.

   e. Log pemrosesan DAG aliran udara. Pilih jenis log pemrosesan Apache Airflow DAG untuk dikirim CloudWatch ke tingkat Log di Log.

8. Opsional. Untuk opsi konfigurasi Aliran Udara, pilih opsi Tambahkan konfigurasi khusus.

   Anda dapat memilih dari daftar dropdown yang disarankan dari opsi konfigurasi Apache Airflow untuk versi Apache Airflow Anda, atau menentukan opsi konfigurasi khusus. Misalnya, `core.default_task_retries : 3`.

9. Di bawah Izin, pilih peran eksekusi:

a. Secara default, Amazon MWAA membuat peran eksekusi di Buat peran baru. Anda harus memiliki izin untuk membuat peran IAM untuk menggunakan opsi ini.

b. Opsional. Pilih Masukkan peran ARN untuk memasukkan Nama Sumber Daya Amazon (ARN) dari peran eksekusi yang ada.

10. Di bawah Spesifikasi pembaruan, pilih Strategi penggantian pekerja untuk mengontrol cara penanganan pekerja aktif selama pembaruan.
11. Pilih Berikutnya.

Langkah ketiga: Tinjau dan perbarui

Untuk meninjau ringkasan lingkungan

- Tinjau ringkasan lingkungan, pilih Simpan.

> **Note**
>
> Dibutuhkan sekitar dua puluh hingga tiga puluh menit untuk memperbarui lingkungan menggunakan pembaruan paksa. Pembaruan lingkungan yang anggun mungkin memakan waktu hingga dua belas jam untuk diselesaikan, karena menunggu tugas Anda yang sedang berlangsung selesai.

Mengubah versi Apache Airflow

Amazon MWAA mendukung peningkatan dan penurunan versi versi minor. Ini berarti Anda dapat memperbarui lingkungan Anda dari versi x**.4.**z ke x**.5.**z atau dari x**.5.**z kex**.4.**z. Untuk melakukan upgrade versi utama, misalnya dari versi **1.**y**.**z ke**2.**y**.**z, Anda harus membuat lingkungan baru dan memigrasikan sumber daya Anda. Untuk informasi selengkapnya tentang meningkatkan ke versi utama baru Apache Airflow, lihat Migrasi ke lingkungan Amazon MWAA baru di Panduan Migrasi Amazon MWAA.

Selama proses upgrade atau downgrade, Amazon MWAA menangkap snapshot metadata lingkungan Anda, upgrade atau downgrade pekerja, penjadwal, server web ke versi Apache Airflow yang baru, dan akhirnya mengembalikan database metadata menggunakan snapshot.

Sebelum Anda meningkatkan atau menurunkan versi, pastikan sumber daya alur kerja Anda DAGs dan lainnya kompatibel dengan versi Apache Airflow baru yang Anda tingkatkan. Jika Anda

menggunakan `requirements.txt` untuk mengelola dependensi, Anda juga harus memastikan bahwa dependensi yang Anda tentukan dalam persyaratan Anda kompatibel dengan versi baru.

Topik

- Tingkatkan atau turunkan sumber daya alur kerja Anda
- Tentukan versi baru

Tingkatkan atau turunkan sumber daya alur kerja Anda

Setiap kali Anda mengubah versi Apache Airflow, pastikan Anda mereferensikan URL yang `--constraint` benar di situs Anda. `requirements.txt`

> **Warning**
>
> Menentukan persyaratan yang tidak sesuai dengan versi Apache Airflow target Anda selama upgrade atau downgrade dapat mengakibatkan proses rollback yang panjang ke versi Apache Airflow sebelumnya dengan versi persyaratan sebelumnya.

Migrasikan sumber daya alur kerja Anda

1. Buat fork aws-mwaa-docker-imagesrepositori, dan kloning salinan pelari lokal Amazon MWAA.
2. Checkout ke cabang aws-mwaa-docker-images repositori yang cocok dengan versi yang Anda upgrade atau downgrade.
3. Untuk memperbarui`requirements.txt`, ikuti praktik terbaik yang kami rekomendasikan dalam Mengelola dependensi Python, di Panduan Pengguna Amazon MWAA.
4. (Opsional) Untuk mempercepat proses upgrade atau downgrade, bersihkan database metadata lingkungan. Lingkungan dengan sejumlah besar metadata dapat memakan waktu lebih lama untuk ditingkatkan.
5. Setelah berhasil menguji sumber daya alur kerja, salin DAGs`requirements.txt`, dan plugin ke bucket Amazon S3 lingkungan Anda.

Anda sekarang siap untuk mengedit lingkungan, menentukan versi Apache Airflow baru, dan memulai prosedur pembaruan.

Tentukan versi baru

Setelah Anda selesai memperbarui sumber daya alur kerja Anda untuk memastikan kompatibilitas dengan versi Apache Airflow yang baru, lakukan hal berikut untuk mengedit detail lingkungan dan menentukan versi Apache Airflow yang ingin Anda tingkatkan.

> **Note**
>
> Saat Anda melakukan upgrade atau downgrade, semua tugas yang saat ini berjalan di lingkungan dihentikan selama prosedur. Prosedur pembaruan dapat memakan waktu hingga dua jam, selama waktu itu lingkungan Anda tidak akan tersedia.

Tentukan versi baru menggunakan konsol

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Dari daftar Lingkungan, pilih lingkungan yang ingin Anda upgrade atau downgrade.
3. Pada halaman lingkungan, pilih Edit untuk mengedit lingkungan.
4. Di bagian Environment details, untuk versi Airflow, pilih nomor versi Apache Airflow yang ingin Anda upgrade atau downgrade lingkungan dari daftar dropdown.
5. Pilih Berikutnya sampai Anda berada di halaman Review dan save.
6. Pada halaman Tinjau dan simpan, tinjau perubahan Anda, lalu pilih Simpan.

Ketika Anda menerapkan perubahan, lingkungan Anda memulai prosedur upgrade atau downgrade. Selama periode ini, status lingkungan Anda menunjukkan tindakan apa yang diambil Amazon MWAA, dan apakah prosedurnya berhasil.

Dalam skenario peningkatan atau penurunan versi yang berhasil, statusnya akan menjadi`UPDATING`, lalu `CREATING_SNAPSHOT` saat Amazon MWAA menangkap cadangan metadata Anda. Akhirnya, status akan kembali dulu ke`UPDATING`, lalu ke `AVAILABLE` saat prosedur selesai.

Jika lingkungan gagal meningkatkan atau menurunkan versi, status lingkungan Anda akan menjadi. `ROLLING_BACK` Jika rollback berhasil, status akan muncul terlebih dahulu`UPDATE_FAILED`, menunjukkan bahwa pembaruan gagal tetapi lingkungan tersedia. Jika rollback gagal, statusnya akan`UNAVAILABLE`, menunjukkan bahwa Anda tidak dapat mengakses lingkungan.

Menggunakan skrip startup dengan Amazon MWAA

Skrip startup adalah skrip shell (`.sh`) yang Anda host di bucket Amazon S3 lingkungan Anda yang mirip dengan Anda DAGs, persyaratan, dan plugin. Amazon MWAA menjalankan skrip ini selama startup pada setiap komponen Apache Airflow individu (pekerja, penjadwal, dan server web) sebelum menginstal persyaratan dan menginisialisasi proses Apache Airflow. Gunakan skrip startup untuk melakukan hal berikut:

- Instal runtime — Instal runtime Linux yang diperlukan oleh alur kerja dan koneksi Anda.
- Konfigurasi variabel lingkungan - Tetapkan variabel lingkungan untuk setiap komponen Apache Airflow. Timpa variabel umum sepertiPATH,PYTHONPATH, danLD_LIBRARY_PATH.
- Kelola kunci dan token — Berikan token akses untuk repositori khusus ke `requirements.txt` dan konfigurasikan kunci keamanan.

Topik berikut menjelaskan cara mengonfigurasi skrip startup untuk menginstal runtime Linux, mengatur variabel lingkungan, dan memecahkan masalah terkait menggunakan Log. CloudWatch

Topik

- Konfigurasikan skrip startup
- Instal runtime Linux menggunakan skrip startup
- Mengatur variabel lingkungan menggunakan skrip startup

Konfigurasikan skrip startup

Untuk menggunakan skrip startup dengan lingkungan Amazon MWAA yang ada, unggah `.sh` file ke bucket Amazon S3 lingkungan Anda. Kemudian, untuk mengaitkan skrip dengan lingkungan, tentukan hal berikut di detail lingkungan Anda:

- Jalur URL Amazon S3 ke skrip — Jalur relatif ke skrip yang dihosting di bucket Anda, misalnya, `s3://mwaa-environment/startup.sh`
- ID versi Amazon S3 dari skrip — Versi skrip shell startup di bucket Amazon S3 Anda. Anda harus menentukan ID versi yang ditetapkan Amazon S3 ke file setiap kali Anda memperbarui skrip. Versi IDs Unicode, UTF-8 dikodekan, URL-ready, string buram yang panjangnya tidak lebih dari 1.024 byte, misalnya,. `3sL4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY+MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo`

Untuk menyelesaikan langkah-langkah di bagian ini, gunakan contoh skrip berikut. Skrip mengeluarkan nilai yang ditetapkan untuk`MWAA_AIRFLOW_COMPONENT`. Variabel lingkungan ini mengidentifikasi setiap komponen Apache Airflow tempat skrip berjalan.

Salin kode dan simpan secara lokal sebagai`startup.sh`.

```
#!/bin/sh

echo "Printing Apache Airflow component"
echo $MWAA_AIRFLOW_COMPONENT
```

Selanjutnya, unggah skrip ke bucket Amazon S3 Anda.

Konsol Manajemen AWS

Untuk mengunggah skrip shell (konsol)

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon S3 di. https://console.aws.amazon.com/s3/
2. Dari daftar Bucket, pilih nama bucket yang terkait dengan lingkungan Anda.
3. Di tab Objek, pilih Unggah.
4. Pada halaman Unggah, seret dan lepas skrip shell yang Anda buat.
5. Pilih Unggah.

Script tercantum dalam daftar Objek. Amazon S3 membuat ID versi baru untuk file tersebut. Jika Anda memperbarui skrip dan mengunggahnya lagi menggunakan nama file yang sama, ID versi baru ditetapkan ke file tersebut.

AWS CLI

Untuk membuat dan mengunggah skrip shell (CLI)

1. Buka prompt perintah baru, dan jalankan `ls` perintah Amazon S3 untuk membuat daftar dan mengidentifikasi bucket yang terkait dengan lingkungan Anda.

   ```
   aws s3 ls
   ```

2. Arahkan ke folder tempat Anda menyimpan skrip shell. Gunakan `cp` di jendela prompt baru untuk mengunggah skrip ke bucket Anda. Ganti *amzn-s3-demo-bucket* dengan informasi Anda.

   ```
   aws s3 cp startup.sh s3://amzn-s3-demo-bucket/startup.sh
   ```

   Jika berhasil, Amazon S3 mengeluarkan jalur URL ke objek:

   ```
   upload: ./startup.sh to s3://amzn-s3-demo-bucket/startup.sh
   ```

3. Gunakan perintah berikut untuk mengambil ID versi terbaru untuk skrip.

   ```
   aws s3api list-object-versions --bucket amzn-s3-demo-bucket --prefix startup --
   query 'Versions[?IsLatest].[VersionId]' --output text
   ```

   ```
   BbdVMmBRjtestta1EsVnbybZp1Wqh1J4
   ```

   Anda menentukan ID versi ini saat Anda mengaitkan skrip dengan lingkungan.

Sekarang, kaitkan skrip dengan lingkungan Anda.

Konsol Manajemen AWS

Untuk mengaitkan skrip dengan lingkungan (konsol)

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih baris untuk lingkungan yang ingin Anda perbarui, lalu pilih Edit.
3. Pada halaman Tentukan detail, untuk file skrip Startup - opsional, masukkan URL Amazon S3 untuk skrip, misalnya:. `s3://amzn-s3-demo-bucket/startup-sh.`
4. Pilih versi terbaru dari daftar drop-down, atau Jelajahi S3 untuk menemukan skrip.
5. Pilih Berikutnya, lalu lanjutkan ke halaman Tinjau dan simpan.
6. Tinjau perubahan, lalu pilih Simpan.

Pembaruan lingkungan dapat memakan waktu antara 10 hingga 30 menit. Amazon MWAA menjalankan skrip startup karena setiap komponen di lingkungan Anda dimulai ulang.

AWS CLI

Untuk mengaitkan skrip dengan lingkungan (CLI)

- Buka prompt perintah dan gunakan `update-environment` untuk menentukan URL Amazon S3 dan ID versi untuk skrip.

```
aws mwaa update-environment \
--name your-mwaa-environment \
--startup-script-s3-path startup.sh \
--startup-script-s3-object-version BbdVMmBRjtestta1EsVnbybZp1Wqh1J4
```

Jika berhasil, Amazon MWAA mengembalikan Amazon Resource Name (ARN) untuk lingkungan:

```
arn:aws:airflow:us-west-2:123456789012:environment/your-mwaa-environment
```

Pembaruan lingkungan dapat memakan waktu antara 10 hingga 30 menit. Amazon MWAA menjalankan skrip startup karena setiap komponen di lingkungan Anda dimulai ulang.

Terakhir, ambil peristiwa log untuk memverifikasi bahwa skrip berfungsi seperti yang diharapkan. Saat Anda mengaktifkan pencatatan untuk setiap komponen Apache Airflow, Amazon MWAA membuat grup log dan aliran log baru. Untuk informasi selengkapnya, lihat jenis log Apache Airflow.

Konsol Manajemen AWS

Untuk mendapatkan aliran log Apache Airflow (konsol)

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan Anda.
3. Di panel Pemantauan, pilih grup log yang ingin Anda akses log, misalnya, grup log penjadwal aliran udara.
4. Di CloudWatch konsol, dari daftar aliran Log, pilih aliran dengan awalan berikut:. `startup_script_execution_ip`
5. Pada panel Log peristiwa, Anda akan merujuk ke output dari perintah mencetak nilai untuk`MWAA_AIRFLOW_COMPONENT`. Misalnya, untuk log penjadwal, Anda akan yang berikut ini:

```
Printing Apache Airflow component
        scheduler
        Finished running startup script. Execution time: 0.004s.
        Running verification
        Verification completed
```

Anda dapat mengulangi langkah-langkah sebelumnya untuk mengakses log pekerja dan server web.

Instal runtime Linux menggunakan skrip startup

Gunakan skrip startup untuk memperbarui sistem operasi komponen Apache Airflow, dan instal pustaka runtime tambahan untuk digunakan dengan alur kerja Anda. Misalnya, skrip berikut berjalan `yum update` untuk memperbarui sistem operasi.

Saat menjalankan `yum update` skrip startup, Anda harus mengecualikan Python menggunakan `--exclude=python*` seperti yang tercantum dalam contoh. Agar lingkungan Anda berjalan, Amazon MWAA menginstal versi Python tertentu yang kompatibel dengan lingkungan Anda. Oleh karena itu, Anda tidak dapat memperbarui versi Python lingkungan menggunakan skrip startup.

```
#!/bin/sh

echo "Updating operating system"
sudo yum update -y --exclude=python*
```

Untuk menginstal runtime pada komponen Apache Airflow tertentu, `MWAA_AIRFLOW_COMPONENT` gunakan `if` dan dan pernyataan bersyarat. `fi` Contoh ini menjalankan satu perintah untuk menginstal `libaio` perpustakaan pada penjadwal dan pekerja, tetapi tidak di server web.

> **Important**
>
> - Jika Anda telah mengonfigurasi server web pribadi, Anda harus menggunakan kondisi berikut atau menyediakan semua file instalasi secara lokal untuk menghindari batas waktu instalasi.
> - Gunakan `sudo` untuk menjalankan operasi yang membutuhkan hak administratif.

```
#!/bin/sh

if [[ "${MWAA_AIRFLOW_COMPONENT}" != "webserver" ]]
  then
    sudo yum -y install libaio

fi
```

Anda dapat menggunakan skrip startup untuk mendapatkan versi Python.

```
#!/bin/sh

export PYTHON_VERSION_CHECK=`python -c 'import sys; version=sys.version_info[:3];
 print("{0}.{1}.{2}".format(*version))'`
echo "Python version is $PYTHON_VERSION_CHECK"
```

Amazon MWAA tidak mendukung penggantian versi Python default, karena hal ini dapat menyebabkan ketidakcocokan dengan pustaka Apache Airflow yang diinstal.

Mengatur variabel lingkungan menggunakan skrip startup

Gunakan skrip startup untuk mengatur variabel lingkungan dan memodifikasi konfigurasi Apache Airflow. Berikut ini mendefinisikan variabel baru,`ENVIRONMENT_STAGE`. Anda dapat mereferensikan variabel ini dalam DAG atau dalam modul kustom Anda.

```
#!/bin/sh

export ENVIRONMENT_STAGE="development"
echo "$ENVIRONMENT_STAGE"
```

Gunakan skrip startup untuk menimpa Apache Airflow atau variabel sistem yang umum. Misalnya, Anda mengatur `LD_LIBRARY_PATH` untuk menginstruksikan Python untuk mencari binari di jalur yang Anda tentukan. Ini memungkinkan Anda menyediakan binari khusus untuk alur kerja Anda menggunakan plugin:

```
#!/bin/sh

export LD_LIBRARY_PATH=/usr/local/airflow/plugins/your-custom-binary
```

Variabel lingkungan yang tersimpan

Amazon MWAA menyimpan satu set variabel lingkungan kritis. Jika Anda menimpa variabel cadangan, Amazon MWAA mengembalikannya ke default. Berikut ini mencantumkan variabel yang dicadangkan:

- `MWAA__AIRFLOW__COMPONENT`— Digunakan untuk mengidentifikasi komponen Apache Airflow dengan salah satu nilai berikut`scheduler`:`worker`,, atau. `webserver`
- `AIRFLOW__WEBSERVER__SECRET_KEY`— Kunci rahasia yang digunakan untuk menandatangani cookie sesi dengan aman di server web Apache Airflow.
- `AIRFLOW__CORE__FERNET_KEY`— Kunci yang digunakan untuk enkripsi dan dekripsi data sensitif yang disimpan dalam database metadata, misalnya, kata sandi koneksi.
- `AIRFLOW_HOME`— Jalur ke direktori home Apache Airflow tempat file konfigurasi dan file DAG disimpan secara lokal.
- `AIRFLOW__CELERY__BROKER_URL`— URL broker pesan yang digunakan untuk komunikasi antara penjadwal Apache Airflow dan node pekerja Celery.
- `AIRFLOW__CELERY__RESULT_BACKEND`— URL database yang digunakan untuk menyimpan hasil tugas Seledri.
- `AIRFLOW__CORE__EXECUTOR`— Kelas eksekutor yang harus digunakan Apache Airflow. Di Amazon MWAA ini adalah `CeleryExecutor`
- `AIRFLOW__CORE__LOAD_EXAMPLES`— Digunakan untuk mengaktifkan, atau menonaktifkan, pemuatan contoh DAGs.
- `AIRFLOW__METRICS__METRICS_BLOCK_LIST`— Digunakan untuk mengelola metrik Apache Airflow mana yang dipancarkan dan ditangkap oleh Amazon MWAA di. CloudWatch
- `SQL_ALCHEMY_CONN`— String koneksi untuk database RDS untuk PostgreSQL yang digunakan untuk menyimpan metadata Apache Airflow di Amazon MWAA.
- `AIRFLOW__CORE__SQL_ALCHEMY_CONN`— Digunakan untuk tujuan yang sama sepertiSQL_ALCHEMY_CONN, tetapi mengikuti konvensi penamaan Apache Airflow yang baru.
- `AIRFLOW__CELERY__DEFAULT_QUEUE`— Antrian default untuk tugas Seledri di Apache Airflow.
- `AIRFLOW__OPERATORS__DEFAULT_QUEUE`— Antrian default untuk tugas menggunakan operator Apache Airflow tertentu.
- `AIRFLOW_VERSION`- Versi Apache Airflow diinstal di lingkungan Amazon MWAA.
- `AIRFLOW_CONN_AWS_DEFAULT`— AWS Kredensi default yang digunakan untuk berintegrasi dengan AWS layanan lain di.

- AWS_DEFAULT_REGION— Menetapkan default yang Wilayah AWS digunakan dengan kredensi default untuk diintegrasikan dengan layanan lain AWS .
- AWS_REGION— Jika didefinisikan, variabel lingkungan ini mengesampingkan nilai dalam variabel lingkungan AWS_DEFAULT_REGION dan wilayah pengaturan profil.
- PYTHONUNBUFFERED— Digunakan untuk mengirim `stdout` dan `stderr` mengalirkan ke log kontainer.
- AIRFLOW__METRICS__STATSD_ALLOW_LIST— Digunakan untuk mengonfigurasi daftar izinkan awalan dipisahkan koma untuk mengirim metrik yang dimulai dengan elemen daftar.
- AIRFLOW__METRICS__STATSD_ON— Mengaktifkan metrik pengiriman ke. `StatsD`
- AIRFLOW__METRICS__STATSD_HOST— Digunakan untuk terhubung ke `StatSD` daemon.
- AIRFLOW__METRICS__STATSD_PORT— Digunakan untuk terhubung ke `StatSD` daemon.
- AIRFLOW__METRICS__STATSD_PREFIX— Digunakan untuk terhubung ke `StatSD` daemon.
- AIRFLOW__CELERY__WORKER_AUTOSCALE— Menetapkan konkurensi maksimum dan minimum.
- AIRFLOW__CORE__DAG_CONCURRENCY— Menetapkan jumlah instance tugas yang dapat dijalankan secara bersamaan oleh penjadwal dalam satu DAG.
- AIRFLOW__CORE__MAX_ACTIVE_TASKS_PER_DAG— Menetapkan jumlah maksimum tugas aktif per DAG.
- AIRFLOW__CORE__PARALLELISM— Mendefinisikan jumlah maksimum instance tugas yang dapat secara bersamaan.
- AIRFLOW__SCHEDULER__PARSING_PROCESSES— Menetapkan jumlah maksimum proses yang diuraikan oleh penjadwal untuk menjadwalkan. DAGs
- AIRFLOW__CELERY_BROKER_TRANSPORT_OPTIONS__VISIBILITY_TIMEOUT— Mendefinisikan jumlah detik pekerja menunggu untuk mengakui tugas sebelum pesan dikirim kembali ke pekerja lain.
- AIRFLOW__CELERY_BROKER_TRANSPORT_OPTIONS__REGION— Menetapkan Wilayah AWS untuk transportasi Seledri yang mendasarinya.
- AIRFLOW__CELERY_BROKER_TRANSPORT_OPTIONS__PREDEFINED_QUEUES— Mengatur antrian untuk transportasi Seledri yang mendasarinya.
- AIRFLOW_SCHEDULER_ALLOWED_RUN_ID_PATTERN— Digunakan untuk memverifikasi validitas input Anda untuk `run_id` parameter saat memicu DAG.
- AIRFLOW__WEBSERVER__BASE_URL— URL server web yang digunakan untuk meng-host Apache Airflow UI.

- `PYTHONPATH`(hanya untuk Apache Airflow v2.9 dan yang lebih baru) - Dicadangkan oleh Amazon MWAA untuk memastikan bahwa semua fungsi lingkungan dasar beroperasi dengan benar.

> **Note**
>
> Untuk Apache Airflow versi lebih awal dari v2.9*PYTHONPATH*, adalah variabel lingkungan tanpa syarat.

Variabel lingkungan yang tidak dicadangkan

Anda dapat menggunakan skrip startup untuk menimpa variabel lingkungan yang tidak dilindungi. Berikut ini mencantumkan beberapa variabel umum ini:

- `PATH`— Menentukan daftar direktori tempat sistem operasi mencari file dan skrip yang dapat dieksekusi. Ketika perintah berjalan di baris perintah, sistem memeriksa direktori `PATH` untuk menemukan dan menjalankan perintah. Saat Anda membuat operator atau tugas khusus di Apache Airflow, Anda mungkin perlu mengandalkan skrip eksternal atau executable. Jika direktori yang berisi file-file ini tidak dalam yang ditentukan dalam `PATH` variabel, tugas gagal dijalankan ketika sistem tidak dapat menemukannya. Dengan menambahkan direktori yang sesuai`PATH`, tugas Apache Airflow dapat menemukan dan menjalankan executable yang diperlukan.
- `PYTHONPATH`— Digunakan oleh penerjemah Python untuk menentukan direktori mana yang akan mencari modul dan paket yang diimpor. Ini adalah daftar direktori yang dapat Anda tambahkan ke jalur pencarian default. Ini memungkinkan penerjemah menemukan dan memuat pustaka Python yang tidak termasuk dalam pustaka standar, atau diinstal dalam direktori sistem. Gunakan variabel ini untuk menambahkan modul Anda dan paket Python kustom dan menggunakannya dengan Anda. DAGs

> **Note**
>
> Untuk Apache Airflow v2.9 dan yang lebih baru*PYTHONPATH*, adalah variabel lingkungan yang dicadangkan.

- `LD_LIBRARY_PATH`- Variabel lingkungan yang digunakan oleh dynamic linker dan loader di Linux untuk menemukan dan memuat pustaka bersama. Ini menentukan daftar direktori yang berisi pustaka bersama, yang dicari sebelum direktori perpustakaan sistem default. Gunakan variabel ini untuk menentukan binari kustom Anda.

- `CLASSPATH`— Digunakan oleh Java Runtime Environment (JRE) dan Java Development Kit (JDK) untuk mencari dan memuat kelas Java, pustaka, dan sumber daya saat runtime. Ini adalah daftar direktori, file JAR, dan arsip ZIP yang berisi kode Java yang dikompilasi.

Bekerja dengan DAGs di Amazon MWAA

Untuk menjalankan Directed Acyclic Graphs () DAGs di Amazon Managed Workflows untuk lingkungan Apache Airflow, Anda menyalin file ke bucket penyimpanan Amazon S3 yang dilampirkan ke lingkungan Anda, lalu beri tahu Amazon MWAA di mana DAGs file Anda dan pendukung berada di konsol Amazon MWAA. Amazon MWAA menangani sinkronisasi DAGs antara pekerja, penjadwal, dan server web. Panduan ini menjelaskan cara menambahkan atau memperbarui DAGs dan menginstal plugin kustom dan dependensi Python di lingkungan Amazon MWAA.

Topik

- Ikhtisar ember Amazon S3
- Menambahkan atau memperbarui DAGs
- Menginstal plugin kustom
- Menginstal dependensi Python
- Menghapus file di Amazon S3

Ikhtisar ember Amazon S3

Bucket Amazon S3 untuk lingkungan Amazon MWAA harus memiliki Akses Publik Diblokir. Secara default, semua sumber daya Amazon S3—bucket, objek, dan sub-sumber daya terkait (misalnya, konfigurasi siklus hidup) —bersifat pribadi.

- Hanya pemilik sumber daya, Akun AWS yang membuat ember, yang dapat mengakses sumber daya. Pemilik sumber daya (misalnya, administrator Anda) dapat memberikan izin akses kepada orang lain dengan menulis kebijakan kontrol akses.
- Kebijakan akses yang Anda atur harus memiliki izin untuk menambahkan DAGs, plugin khususplugins.zip, dan dependensi Python ke bucket Amazon S3 `requirements.txt` Anda. Untuk contoh kebijakan yang berisi izin yang diperlukan, lihat Amazon MWAAFull ConsoleAccess.

Bucket Amazon S3 untuk lingkungan Amazon MWAA harus mengaktifkan Versioning. Saat pembuatan versi bucket Amazon S3 diaktifkan, kapan pun versi baru dibuat, salinan baru dibuat.

- Pembuatan versi diaktifkan untuk plugin khusus dalam dependensi, `plugins.zip` dan Python di bucket Amazon S3 Anda. `requirements.txt`

- Anda harus menentukan versi`plugins.zip`, dan `requirements.txt` di konsol Amazon MWAA setiap kali file ini diperbarui di bucket Amazon S3 Anda.

Menambahkan atau memperbarui DAGs

Directed Acyclic Graphs (DAGs) didefinisikan dalam file Python yang mendefinisikan struktur DAG sebagai kode. Anda dapat menggunakan AWS CLI atau konsol Amazon S3 untuk mengunggah DAGs ke lingkungan Anda. Topik ini menjelaskan langkah-langkah untuk menambah atau memperbarui Apache DAGs Airflow di lingkungan Alur Kerja Terkelola Amazon untuk Apache Airflow menggunakan folder di bucket Amazon S3 Anda. `dags`

Bagian-bagian

- Prasyarat
- Cara kerjanya
- Apa yang berubah?
- Pengujian DAGs menggunakan utilitas Amazon MWAA CLI
- Mengunggah kode DAG ke Amazon S3
- Menentukan jalur ke DAGs folder Anda di konsol Amazon MWAA (pertama kali)
- Mengakses perubahan pada Apache Airflow UI Anda
- Apa selanjutnya?

Prasyarat

Anda memerlukan yang berikut ini sebelum dapat menyelesaikan langkah-langkah di halaman ini.

- Izin — Anda Akun AWS harus telah diberikan akses oleh administrator Anda ke kebijakan kontrol MWAAFull ConsoleAccess akses Amazon untuk lingkungan Anda. Selain itu, lingkungan Amazon MWAA Anda harus diizinkan oleh peran eksekusi Anda untuk mengakses AWS sumber daya yang digunakan oleh lingkungan Anda.
- Akses — Jika Anda memerlukan akses ke repositori publik untuk menginstal dependensi langsung di server web, lingkungan Anda harus dikonfigurasi dengan akses server web jaringan publik. Untuk informasi lebih lanjut, lihatthe section called “Mode akses Apache Airflow”.
- Konfigurasi Amazon S3 - Bucket Amazon S3 yang digunakan untuk menyimpan plugin kustom DAGs Anda`plugins.zip`, dan dependensi Python harus dikonfigurasi dengan Akses Publik Diblokir dan `requirements.txt` Diaktifkan Versi.

Cara kerjanya

Grafik Asiklik Terarah (DAG) didefinisikan dalam satu file Python yang mendefinisikan struktur DAG sebagai kode. Ini terdiri dari yang berikut:

- Definisi DAG.
- Operator yang menjelaskan cara menjalankan DAG dan tugas yang harus dijalankan.
- Hubungan operator yang menggambarkan urutan untuk menjalankan tugas.

Untuk menjalankan platform Apache Airflow di lingkungan Amazon MWAA, Anda perlu menyalin definisi DAG ke folder `dags` di bucket penyimpanan Anda. Misalnya, folder DAG di bucket penyimpanan Anda harus menyerupai:

Example Folder DAG

```
dags/
# dag_def.py
```

Amazon MWAA secara otomatis menyinkronkan objek baru dan yang diubah dari bucket Amazon S3 Anda ke folder penjadwal Amazon MWAA dan `/usr/local/airflow/dags` container pekerja setiap 30 detik, menjaga hierarki file sumber Amazon S3, terlepas dari jenis file. Waktu yang DAGs dibutuhkan baru untuk dicantumkan di UI Apache Airflow Anda dikendalikan oleh. `scheduler.dag_dir_list_interval` Perubahan yang ada DAGs akan diambil pada loop pemrosesan DAG berikutnya.

> **Note**
>
> Anda tidak perlu menyertakan file `airflow.cfg` konfigurasi di folder DAG Anda. Anda dapat mengganti konfigurasi Apache Airflow default dari konsol Amazon MWAA. Untuk informasi lebih lanjut, lihatMenggunakan opsi konfigurasi Apache Airflow di Amazon MWAA.

Apa yang berubah?

Untuk meninjau perubahan rilis Apache Airflow tertentu, lihat halaman Catatan Rilis.

- Konfigurasi Apache Airflow v3: Referensi Konfigurasi
- Apache Airflow v2 informasi antarmuka publik: Antarmuka Publik Aliran Udara

Pengujian DAGs menggunakan utilitas Amazon MWAA CLI

- Utilitas antarmuka baris perintah (CLI) mereplikasi Alur Kerja Terkelola Amazon untuk lingkungan Apache Airflow secara lokal.
- CLI membangun image container Docker secara lokal yang mirip dengan image produksi Amazon MWAA. Anda dapat menggunakan ini untuk menjalankan lingkungan Apache Airflow lokal untuk mengembangkan dan DAGs menguji, plugin khusus, dan dependensi sebelum menerapkan ke Amazon MWAA.
- Untuk menjalankan CLI, lihat pada. aws-mwaa-docker-images GitHub

Mengunggah kode DAG ke Amazon S3

Anda dapat menggunakan konsol Amazon S3 atau AWS Command Line Interface (AWS CLI) untuk mengunggah kode DAG ke bucket Amazon S3 Anda. Langkah-langkah berikut mengasumsikan Anda mengunggah kode (`.py`) ke folder bernama `dags` di bucket Amazon S3 Anda.

Menggunakan AWS CLI

The AWS Command Line Interface (AWS CLI) adalah alat open source yang dapat Anda gunakan untuk berinteraksi dengan AWS layanan menggunakan perintah di shell baris perintah Anda. Untuk menyelesaikan langkah-langkah di halaman ini, Anda memerlukan yang berikut:

- AWS CLI — Instal versi 2.
- AWS CLI - Konfigurasi cepat dengan `aws configure`.

Untuk mengunggah menggunakan AWS CLI

1. Gunakan perintah berikut untuk membuat daftar semua bucket Amazon S3 Anda.

```
aws s3 ls
```

2. Gunakan perintah berikut untuk mencantumkan file dan folder di bucket Amazon S3 untuk lingkungan Anda.

```
aws s3 ls s3://YOUR_S3_BUCKET_NAME
```

3. Perintah berikut mengunggah `dag_def.py` file ke `dags` folder.

```
aws s3 cp dag_def.py s3://amzn-s3-demo-bucket/dags/
```

Jika folder bernama `dags` belum ada di bucket Amazon S3 Anda, perintah ini membuat `dags` folder dan mengunggah file bernama `dag_def.py` ke folder baru.

Menggunakan konsol Amazon S3

Konsol Amazon S3 adalah antarmuka pengguna berbasis web yang dapat Anda gunakan untuk membuat dan mengelola sumber daya di bucket Amazon S3 Anda. Langkah-langkah berikut menganggap Anda memiliki DAGs folder bernama`dags`.

Untuk mengunggah menggunakan konsol Amazon S3

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.
3. Pilih tautan bucket S3 di kode DAG di panel S3 untuk membuka bucket penyimpanan Anda di konsol.
4. Pilih `dags` folder.
5. Pilih Unggah.
6. Pilih Tambahkan file.
7. Pilih salinan lokal Anda`dag_def.py`, pilih Unggah.

Menentukan jalur ke DAGs folder Anda di konsol Amazon MWAA (pertama kali)

Langkah-langkah berikut mengasumsikan Anda menentukan jalur ke folder di bucket Amazon S3 Anda bernama. `dags`

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan tempat Anda ingin berlari DAGs.
3. Pilih Edit.
4. Pada kode DAG di panel Amazon S3, pilih Jelajahi S3 yang berdekatan dengan bidang folder DAG.
5. Pilih `dags` folder Anda.

6. Pilih Tutup.
7. Pilih Berikutnya, Perbarui lingkungan.

Mengakses perubahan pada Apache Airflow UI Anda

Anda memerlukan Kebijakan akses Apache Airflow UI: AmazonMWAAWebServerAccess izin untuk Akun AWS in AWS Identity and Access Management (IAM) untuk mengakses UI Apache Airflow Anda.

Untuk mengakses UI Apache Airflow Anda

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.
3. Pilih Buka UI Aliran Udara.

Apa selanjutnya?

Uji plugin khusus DAGs, dan dependensi Python Anda secara lokal menggunakan on. aws-mwaa-docker-images GitHub

Menginstal plugin kustom

Alur Kerja Terkelola Amazon untuk Apache Airflow mendukung pengelola plugin bawaan Apache Airflow, memungkinkan Anda menggunakan operator, kait, sensor, atau antarmuka Apache Airflow khusus. Halaman ini menjelaskan langkah-langkah untuk menginstal plugin kustom Apache Airflow di lingkungan Amazon MWAA Anda menggunakan file. `plugins.zip`

Daftar Isi

- Prasyarat
- Cara kerjanya
- Kapan menggunakan plugin
- Ikhtisar plugin kustom
  - Direktori plugin kustom dan batas ukuran
- Contoh plugin kustom
  - Contoh menggunakan struktur direktori datar di plugins.zip

  - Contoh menggunakan struktur direktori bersarang di plugins.zip
- Membuat file plugins.zip
  - Langkah satu: Uji plugin khusus menggunakan utilitas Amazon MWAA CLI
  - Langkah kedua: Buat file plugins.zip
- Mengunggah plugins.zip ke Amazon S3
  - Menggunakan AWS CLI
  - Menggunakan konsol Amazon S3
- Menginstal plugin khusus di lingkungan Anda
  - Menentukan jalur ke plugins.zip konsol Amazon MWAA (pertama kali)
  - Menentukan plugins.zip versi di konsol Amazon MWAA
- Contoh kasus penggunaan untuk plugins.zip
- Apa selanjutnya?

Prasyarat

Anda akan memerlukan yang berikut ini sebelum Anda dapat menyelesaikan langkah-langkah di halaman ini.

- Izin — Anda Akun AWS harus telah diberikan akses oleh administrator Anda ke kebijakan kontrol MWAAFull ConsoleAccess akses Amazon untuk lingkungan Anda. Selain itu, lingkungan Amazon MWAA Anda harus diizinkan oleh peran eksekusi Anda untuk mengakses AWS sumber daya yang digunakan oleh lingkungan Anda.
- Akses — Jika Anda memerlukan akses ke repositori publik untuk menginstal dependensi langsung di server web, lingkungan Anda harus dikonfigurasi dengan akses server web jaringan publik. Untuk informasi lebih lanjut, lihatthe section called “Mode akses Apache Airflow”.
- Konfigurasi Amazon S3 - Bucket Amazon S3 yang digunakan untuk menyimpan plugin kustom DAGs Andaplugins.zip, dan dependensi Python harus dikonfigurasi dengan Akses Publik Diblokir dan `requirements.txt` Diaktifkan Versi.

Cara kerjanya

Untuk menjalankan plugin khusus di lingkungan Anda, Anda harus melakukan tiga hal:

1. Buat `plugins.zip` file secara lokal.

2. Unggah `plugins.zip` file lokal ke bucket Amazon S3 Anda.
3. Tentukan versi file ini di bidang file Plugins di konsol Amazon MWAA.

> **Note**
>
> Jika ini adalah pertama kalinya Anda mengunggah `plugins.zip` ke bucket Amazon S3 Anda, Anda juga perlu menentukan jalur ke file di konsol Amazon MWAA. Anda hanya perlu menyelesaikan langkah ini sekali.

Kapan menggunakan plugin

Plugin hanya diperlukan untuk memperluas antarmuka pengguna Apache Airflow, seperti yang diuraikan dalam dokumentasi Apache Airflow. Operator kustom dapat ditempatkan langsung di `/dags` folder di samping `DAG` kode Anda.

Jika Anda perlu membuat integrasi Anda sendiri dengan sistem eksternal, letakkan di `dags` folder/ atau subfolder di dalamnya, tetapi tidak di `plugins.zip` folder. Di Apache Airflow 2.x, plugin terutama digunakan untuk memperluas UI.

Demikian pula, dependensi lain tidak dapat ditempatkan di. `plugins.zip` Sebagai gantinya, mereka dapat disimpan di lokasi di `/dags` folder Amazon S3, di mana mereka akan disinkronkan ke setiap wadah Amazon MWAA sebelum Apache Airflow dimulai.

> **Note**
>
> File apa pun di `/dags` folder atau `plugins.zip` yang tidak secara eksplisit mendefinisikan objek Apache Airflow DAG harus terdaftar dalam file. `.airflowignore`

Ikhtisar plugin kustom

Manajer plugin bawaan Apache Airflow dapat mengintegrasikan fitur eksternal ke intinya hanya dengan menjatuhkan file ke dalam folder. `$AIRFLOW_HOME/plugins` Ini dapat Anda gunakan untuk menggunakan operator, kait, sensor, atau antarmuka Apache Airflow khusus. Bagian berikut memberikan contoh struktur direktori datar dan bersarang di lingkungan pengembangan lokal dan pernyataan impor yang dihasilkan, yang menentukan struktur direktori dalam plugins.zip.

Direktori plugin kustom dan batas ukuran

Penjadwal Apache Airflow dan pekerja mencari plugin khusus selama startup pada wadah Fargate yang AWS dikelola untuk lingkungan Anda di. `/usr/local/airflow/plugins/*`

- Struktur direktori. Struktur direktori (at`/*`) didasarkan pada isi `plugins.zip` file Anda. Misalnya, jika Anda `plugins.zip` berisi `operators` direktori sebagai direktori tingkat utama, maka direktori akan diekstraksi ke lingkungan Anda`/usr/local/airflow/plugins/operators`.
- Batas ukuran. Kami merekomendasikan `plugins.zip` file kurang dari 1 GB. Semakin besar ukuran `plugins.zip` file, semakin lama waktu startup pada suatu lingkungan. Meskipun Amazon MWAA tidak membatasi ukuran `plugins.zip` file secara eksplisit, jika dependensi tidak dapat diinstal dalam sepuluh menit, layanan Fargate akan habis waktu dan mencoba mengembalikan lingkungan ke keadaan stabil.

> **Note**
>
> Untuk lingkungan yang menggunakan Apache Airflow v2.0.2, Amazon MWAA membatasi lalu lintas keluar di server web Apache Airflow, dan tidak mengizinkan pemasangan plugin atau dependensi Python langsung di server web. Dimulai dengan Apache Airflow v2.2.2, Amazon MWAA dapat menginstal plugin dan dependensi langsung di server web.

Contoh plugin kustom

Bagian berikut menggunakan kode contoh dalam panduan referensi Apache Airflow untuk menjelaskan cara menyusun lingkungan pengembangan lokal Anda.

Contoh menggunakan struktur direktori datar di plugins.zip

Apache Airflow v3

Contoh berikut menyajikan `plugins.zip` file dengan struktur direktori datar untuk Apache Airflow v3.

Example direktori datar dengan **`PythonVirtualenvOperator`** plugins.zip

Contoh berikut menampilkan pohon tingkat utama dari file plugins.zip untuk plugin `PythonVirtualenvOperator` kustom di. Membuat plugin khusus untuk Apache Airflow PythonVirtualenvOperator

```
### virtual_python_plugin.py
```

Example plugins/virtual_python_plugin.py

Contoh berikut menampilkan plugin `PythonVirtualenvOperator` kustom.

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
from airflow.plugins_manager import AirflowPlugin
import airflow.utils.python_virtualenv
from typing import List

def _generate_virtualenv_cmd(tmp_dir: str, python_bin: str, system_site_packages:
 bool) -> List[str]:
    cmd = ['python3','/usr/local/airflow/.local/lib/python3.7/site-packages/
virtualenv', tmp_dir]
    if system_site_packages:
        cmd.append('--system-site-packages')
    if python_bin is not None:
        cmd.append(f'--python={python_bin}')
    return cmd

airflow.utils.python_virtualenv._generate_virtualenv_cmd=_generate_virtualenv_cmd

class VirtualPythonPlugin(AirflowPlugin):
    name = 'virtual_python_plugin'
```

Apache Airflow v2

Contoh berikut menyajikan `plugins.zip` file dengan struktur direktori datar untuk Apache Airflow v2.

Example direktori datar dengan **`PythonVirtualenvOperator`** plugins.zip

Contoh berikut menampilkan pohon tingkat utama dari file plugins.zip untuk plugin `PythonVirtualenvOperator` kustom di. Membuat plugin khusus untuk Apache Airflow PythonVirtualenvOperator

```
### virtual_python_plugin.py
```

Example plugins/virtual_python_plugin.py

Contoh berikut menampilkan plugin `PythonVirtualenvOperator` kustom.

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
from airflow.plugins_manager import AirflowPlugin
import airflow.utils.python_virtualenv
from typing import List

def _generate_virtualenv_cmd(tmp_dir: str, python_bin: str, system_site_packages:
 bool) -> List[str]:
    cmd = ['python3','/usr/local/airflow/.local/lib/python3.7/site-packages/
virtualenv', tmp_dir]
    if system_site_packages:
        cmd.append('--system-site-packages')
```

```
    if python_bin is not None:
        cmd.append(f'--python={python_bin}')
    return cmd

airflow.utils.python_virtualenv._generate_virtualenv_cmd=_generate_virtualenv_cmd

class VirtualPythonPlugin(AirflowPlugin):
    name = 'virtual_python_plugin'
```

Contoh menggunakan struktur direktori bersarang di plugins.zip

Apache Airflow v3

Contoh berikut menyajikan `plugins.zip` file dengan direktori terpisah untuk`hooks`,`operators`, dan `sensors` direktori.

Example plugins.zip

```
__init__.py
my_airflow_plugin.py
 hooks/
  |-- __init__.py
  |-- my_airflow_hook.py
 operators/
  |-- __init__.py
  |-- my_airflow_operator.py
  |-- hello_operator.py
 sensors/
  |-- __init__.py
  |-- my_airflow_sensor.py
```

Contoh berikut menampilkan pernyataan impor di DAG (DAGs folder) yang menggunakan plugin kustom.

Example dags/your_dag.py

```
from airflow import DAG
from datetime import datetime, timedelta
from operators.my_airflow_operator import MyOperator
from sensors.my_airflow_sensor import MySensor
from operators.hello_operator import HelloOperator
```

```
default_args = {
 'owner': 'airflow',
 'depends_on_past': False,
 'start_date': datetime(2018, 1, 1),
 'email_on_failure': False,
 'email_on_retry': False,
 'retries': 1,
 'retry_delay': timedelta(minutes=5),
}


with DAG('customdag',
   max_active_runs=3,
   schedule_interval='@once',
   default_args=default_args) as dag:

 sens = MySensor(
  task_id='taskA'
 )

 op = MyOperator(
  task_id='taskB',
  my_field='some text'
 )

 hello_task = HelloOperator(task_id='sample-task', name='foo_bar')



 sens >> op >> hello_task
```

Example plugins/my_airflow_plugin.py

```
from airflow.plugins_manager import AirflowPlugin
from hooks.my_airflow_hook import *
from operators.my_airflow_operator import *

class PluginName(AirflowPlugin):

    name = 'my_airflow_plugin'

    hooks = [MyHook]
```

```
    operators = [MyOperator]
    sensors = [MySensor]
```

Contoh berikut menyajikan setiap pernyataan impor yang diperlukan dalam file plugin kustom.

Example hooks/my_airflow_hook.py

```
from airflow.hooks.base import BaseHook


class MyHook(BaseHook):

    def my_method(self):
        print("Hello World")
```

Example sensors/my_airflow_sensor.py

```
from airflow.sensors.base import BaseSensorOperator
from airflow.utils.decorators import apply_defaults


class MySensor(BaseSensorOperator):

    @apply_defaults
    def __init__(self,
                 *args,
                 **kwargs):
        super(MySensor, self).__init__(*args, **kwargs)

    def poke(self, context):
        return True
```

Example operators/my_airflow_operator.py

```
from airflow.operators.bash import BaseOperator
from airflow.utils.decorators import apply_defaults
from hooks.my_airflow_hook import MyHook


class MyOperator(BaseOperator):

    @apply_defaults
    def __init__(self,
```

```
                my_field,
                *args,
                **kwargs):
        super(MyOperator, self).__init__(*args, **kwargs)
        self.my_field = my_field

    def execute(self, context):
        hook = MyHook('my_conn')
        hook.my_method()
```

Example operators/hello_operator.py

```
from airflow.models.baseoperator import BaseOperator
from airflow.utils.decorators import apply_defaults

class HelloOperator(BaseOperator):

    @apply_defaults
    def __init__(
            self,
            name: str,
            **kwargs) -> None:
        super().__init__(**kwargs)
        self.name = name

    def execute(self, context):
        message = "Hello {}".format(self.name)
        print(message)
        return message
```

Ikuti langkah-langkah dalam Menguji plugin kustom menggunakan utilitas Amazon MWAA CLI, dan kemudian Membuat file plugins.zip untuk zip konten dalam direktori Anda. `plugins` Misalnya, `cd plugins`.

Apache Airflow v2

Contoh berikut menyajikan `plugins.zip` file dengan direktori terpisah untuk`hooks`,`operators`, dan `sensors` direktori.

Example plugins.zip

```
__init__.py
 my_airflow_plugin.py
```

```
 hooks/
  |-- __init__.py
  |-- my_airflow_hook.py
 operators/
  |-- __init__.py
  |-- my_airflow_operator.py
  |-- hello_operator.py
 sensors/
  |-- __init__.py
  |-- my_airflow_sensor.py
```

Contoh berikut menampilkan pernyataan impor di DAG (DAGs folder) yang menggunakan plugin kustom.

Example dags/your_dag.py

```
from airflow import DAG
from datetime import datetime, timedelta
from operators.my_airflow_operator import MyOperator
from sensors.my_airflow_sensor import MySensor
from operators.hello_operator import HelloOperator

default_args = {
 'owner': 'airflow',
 'depends_on_past': False,
 'start_date': datetime(2018, 1, 1),
 'email_on_failure': False,
 'email_on_retry': False,
 'retries': 1,
 'retry_delay': timedelta(minutes=5),
}


with DAG('customdag',
   max_active_runs=3,
   schedule_interval='@once',
   default_args=default_args) as dag:

 sens = MySensor(
  task_id='taskA'
 )

 op = MyOperator(
```

```
 task_id='taskB',
 my_field='some text'
)

hello_task = HelloOperator(task_id='sample-task', name='foo_bar')



sens >> op >> hello_task
```

Example plugins/my_airflow_plugin.py

```
from airflow.plugins_manager import AirflowPlugin
from hooks.my_airflow_hook import *
from operators.my_airflow_operator import *

class PluginName(AirflowPlugin):

    name = 'my_airflow_plugin'

    hooks = [MyHook]
    operators = [MyOperator]
    sensors = [MySensor]
```

Contoh berikut menyajikan setiap pernyataan impor yang diperlukan dalam file plugin kustom.

Example hooks/my_airflow_hook.py

```
from airflow.hooks.base import BaseHook


class MyHook(BaseHook):

    def my_method(self):
        print("Hello World")
```

Example sensors/my_airflow_sensor.py

```
from airflow.sensors.base import BaseSensorOperator
from airflow.utils.decorators import apply_defaults
```

```
class MySensor(BaseSensorOperator):

    @apply_defaults
    def __init__(self,
                 *args,
                 **kwargs):
        super(MySensor, self).__init__(*args, **kwargs)

    def poke(self, context):
        return True
```

Example operators/my_airflow_operator.py

```
from airflow.operators.bash import BaseOperator
from airflow.utils.decorators import apply_defaults
from hooks.my_airflow_hook import MyHook


class MyOperator(BaseOperator):

    @apply_defaults
    def __init__(self,
                 my_field,
                 *args,
                 **kwargs):
        super(MyOperator, self).__init__(*args, **kwargs)
        self.my_field = my_field

    def execute(self, context):
        hook = MyHook('my_conn')
        hook.my_method()
```

Example operators/hello_operator.py

```
from airflow.models.baseoperator import BaseOperator
from airflow.utils.decorators import apply_defaults

class HelloOperator(BaseOperator):

    @apply_defaults
    def __init__(
            self,
            name: str,
```

```
            **kwargs) -> None:
        super().__init__(**kwargs)
        self.name = name

    def execute(self, context):
        message = "Hello {}".format(self.name)
        print(message)
        return message
```

Ikuti langkah-langkah dalam Menguji plugin kustom menggunakan utilitas Amazon MWAA CLI, dan kemudian Membuat file plugins.zip untuk zip konten dalam direktori Anda. `plugins` Misalnya, `cd plugins`.

Membuat file plugins.zip

Langkah-langkah berikut menjelaskan langkah-langkah yang kami sarankan untuk membuat file plugins.zip secara lokal.

Langkah satu: Uji plugin khusus menggunakan utilitas Amazon MWAA CLI

- Utilitas antarmuka baris perintah (CLI) mereplikasi Alur Kerja Terkelola Amazon untuk lingkungan Apache Airflow secara lokal.
- CLI membangun image container Docker secara lokal yang mirip dengan image produksi Amazon MWAA. Anda dapat menggunakan ini untuk menjalankan lingkungan Apache Airflow lokal untuk mengembangkan dan DAGs menguji, plugin kustom, dan dependensi sebelum menerapkan ke Amazon MWAA.
- Untuk menjalankan CLI, lihat pada. aws-mwaa-docker-images GitHub

Langkah kedua: Buat file plugins.zip

Anda dapat menggunakan utilitas arsip ZIP bawaan, atau utilitas ZIP lainnya (seperti 7zip) untuk membuat file.zip.

> **Note**
>
> Utilitas zip bawaan untuk OS Windows dapat menambahkan subfolder saat Anda membuat file.zip. Sebaiknya verifikasi konten file plugins.zip sebelum mengunggah ke bucket Amazon S3 Anda untuk memastikan tidak ada direktori tambahan yang ditambahkan.

1. Ubah direktori ke direktori plugin Airflow lokal Anda. Contoh:

```
myproject$ cd plugins
```

2. Jalankan perintah berikut untuk memastikan bahwa konten memiliki izin yang dapat dieksekusi (hanya macOS dan Linux).

```
plugins$ chmod -R 755 .
```

3. Zip konten di dalam `plugins` folder Anda.

```
plugins$ zip -r plugins.zip .
```

Mengunggah **`plugins.zip`** ke Amazon S3

Anda dapat menggunakan konsol Amazon S3 atau AWS Command Line Interface (AWS CLI) untuk mengunggah `plugins.zip` file ke bucket Amazon S3 Anda.

Menggunakan AWS CLI

The AWS Command Line Interface (AWS CLI) adalah alat open source yang dapat Anda gunakan untuk berinteraksi dengan AWS layanan menggunakan perintah di shell baris perintah Anda. Untuk menyelesaikan langkah-langkah di halaman ini, Anda memerlukan yang berikut:

- AWS CLI — Instal versi 2.
- AWS CLI - Konfigurasi cepat dengan `aws configure`.

Untuk mengunggah menggunakan AWS CLI

1. Di prompt perintah Anda, arahkan ke direktori tempat `plugins.zip` file Anda disimpan. Contoh:

```
cd plugins
```

2. Gunakan perintah berikut untuk membuat daftar semua bucket Amazon S3 Anda.

```
aws s3 ls
```

3. Gunakan perintah berikut untuk mencantumkan file dan folder di bucket Amazon S3 untuk lingkungan Anda.

```
aws s3 ls s3://YOUR_S3_BUCKET_NAME
```

4. Gunakan perintah berikut untuk mengunggah `plugins.zip` file ke bucket Amazon S3 untuk lingkungan Anda.

```
aws s3 cp plugins.zip s3://amzn-s3-demo-bucket/plugins.zip
```

Menggunakan konsol Amazon S3

Konsol Amazon S3 adalah antarmuka pengguna berbasis web yang dapat Anda gunakan untuk membuat dan mengelola sumber daya di bucket Amazon S3 Anda.

Untuk mengunggah menggunakan konsol Amazon S3

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.
3. Pilih tautan bucket S3 di kode DAG di panel S3 untuk membuka bucket penyimpanan Anda di konsol.
4. Pilih Unggah.
5. Pilih Tambahkan file.
6. Pilih salinan lokal Anda`plugins.zip`, pilih Unggah.

Menginstal plugin khusus di lingkungan Anda

Bagian ini menjelaskan cara menginstal plugin khusus yang Anda unggah ke bucket Amazon S3 Anda dengan menentukan jalur ke file plugins.zip, dan menentukan versi file plugins.zip setiap kali file zip diperbarui.

Menentukan jalur ke **`plugins.zip`** konsol Amazon MWAA (pertama kali)

Jika ini adalah pertama kalinya Anda mengunggah `plugins.zip` ke bucket Amazon S3 Anda, Anda juga perlu menentukan jalur ke file di konsol Amazon MWAA. Anda hanya perlu menyelesaikan langkah ini sekali.

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.

3. Pilih Edit.
4. Pada kode DAG di panel Amazon S3, pilih Jelajahi S3 yang berdekatan dengan file Plugins - bidang opsional.
5. Pilih `plugins.zip` file di bucket Amazon S3 Anda.
6. Pilih Tutup.
7. Pilih Berikutnya, Perbarui lingkungan.

Menentukan **`plugins.zip`** versi di konsol Amazon MWAA

Anda perlu menentukan versi `plugins.zip` file Anda di konsol Amazon MWAA setiap kali Anda mengunggah versi baru Anda `plugins.zip` di bucket Amazon S3 Anda.

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.
3. Pilih Edit.
4. Pada kode DAG di panel Amazon S3, pilih `plugins.zip` versi dalam daftar tarik-turun.
5. Pilih Berikutnya.

Contoh kasus penggunaan untuk plugins.zip

- Pelajari cara membuat plugin khusus diPlugin kustom dengan Apache Hive dan Hadoop.
- Pelajari cara membuat plugin khusus diPlugin kustom untuk menambal PythonVirtualenvOperator .
- Pelajari cara membuat plugin khusus diPlugin kustom dengan Oracle.
- Pelajari cara membuat plugin khusus dithe section called “Mengubah zona waktu DAG”.

Apa selanjutnya?

Uji DAGs, plugin kustom, dan dependensi Python Anda secara lokal menggunakan on. aws-mwaa-docker-images GitHub

Menginstal dependensi Python

Ketergantungan Python adalah paket atau distribusi apa pun yang tidak termasuk dalam instalasi dasar Apache Airflow untuk versi Apache Airflow Anda di Amazon Managed Workflows untuk

lingkungan Apache Airflow. Topik ini menjelaskan langkah-langkah untuk menginstal dependensi Apache Airflow Python di lingkungan Amazon MWAA Anda menggunakan `requirements.txt` file di bucket Amazon S3 Anda.

Daftar Isi

- Prasyarat
- Cara kerjanya
- Ikhtisar dependensi Python
  - Ketergantungan Python lokasi dan batas ukuran
- Membuat file requirements.txt
  - Langkah satu: Uji dependensi Python menggunakan utilitas Amazon MWAA CLI
  - Langkah kedua: Buat requirements.txt
- Mengunggah requirements.txt ke Amazon S3
  - Menggunakan AWS CLI
  - Menggunakan konsol Amazon S3
- Menginstal dependensi Python di lingkungan Anda
  - Menentukan jalur ke requirements.txt konsol Amazon MWAA (pertama kali)
  - Menentukan requirements.txt versi di konsol Amazon MWAA
- Mengakses log untuk requirements.txt
- Apa selanjutnya?

Prasyarat

Anda akan memerlukan yang berikut ini sebelum Anda dapat menyelesaikan langkah-langkah di halaman ini.

- Izin — Anda Akun AWS harus telah diberikan akses oleh administrator Anda ke kebijakan kontrol MWAAFull ConsoleAccess akses Amazon untuk lingkungan Anda. Selain itu, lingkungan Amazon MWAA Anda harus diizinkan oleh peran eksekusi Anda untuk mengakses AWS sumber daya yang digunakan oleh lingkungan Anda.
- Akses — Jika Anda memerlukan akses ke repositori publik untuk menginstal dependensi langsung di server web, lingkungan Anda harus dikonfigurasi dengan akses server web jaringan publik. Untuk informasi lebih lanjut, lihatthe section called “Mode akses Apache Airflow”.

- Konfigurasi Amazon S3 - Bucket Amazon S3 yang digunakan untuk menyimpan plugin kustom DAGs Andaplugins.zip, dan dependensi Python harus dikonfigurasi dengan Akses Publik Diblokir dan requirements.txt Diaktifkan Versi.

Cara kerjanya

Di Amazon MWAA, Anda menginstal semua dependensi Python dengan mengunggah file ke bucket Amazon S3 requirements.txt Anda, lalu menentukan versi file di konsol Amazon MWAA setiap kali Anda memperbarui file. Amazon MWAA berjalan pip3 install -r requirements.txt untuk menginstal dependensi Python pada penjadwal Apache Airflow dan masing-masing pekerja.

Untuk menjalankan dependensi Python pada lingkungan Anda, Anda harus melakukan tiga hal:

1. Buat requirements.txt file secara lokal.
2. Unggah lokal requirements.txt ke bucket Amazon S3 Anda.
3. Tentukan versi file ini di bidang file Persyaratan di konsol Amazon MWAA.

> **Note**
>
> Jika ini adalah pertama kalinya Anda membuat dan mengunggah requirements.txt ke bucket Amazon S3 Anda, Anda juga perlu menentukan jalur ke file di konsol Amazon MWAA. Anda hanya perlu menyelesaikan langkah ini sekali.

Ikhtisar dependensi Python

Anda dapat menginstal tambahan Apache Airflow dan dependensi Python lainnya dari Python Package Index PyPi (.org), roda Python (), atau dependensi Python yang dihosting pada Repo Compliant .whl /PEP-503 pribadi di lingkungan Anda. PyPi

Ketergantungan Python lokasi dan batas ukuran

Penjadwal Apache Airflow dan pekerja mencari paket dalam requirements.txt file dan paket diinstal pada lingkungan di. /usr/local/airflow/.local/bin

- Batas ukuran. Kami merekomendasikan requirements.txt file yang mereferensikan perpustakaan yang ukuran gabungannya kurang dari 1 GB. Semakin banyak perpustakaan

Amazon MWAA perlu menginstal, semakin lama waktu startup di lingkungan. Meskipun Amazon MWAA tidak membatasi ukuran pustaka yang diinstal secara eksplisit, jika dependensi tidak dapat diinstal dalam sepuluh menit, layanan Fargate akan habis waktu dan mencoba mengembalikan lingkungan ke keadaan stabil.

Membuat file requirements.txt

Langkah-langkah berikut menjelaskan langkah-langkah yang kami sarankan untuk membuat file requirements.txt secara lokal.

Langkah satu: Uji dependensi Python menggunakan utilitas Amazon MWAA CLI

- Utilitas antarmuka baris perintah (CLI) mereplikasi Alur Kerja Terkelola Amazon untuk lingkungan Apache Airflow secara lokal.
- CLI membangun image container Docker secara lokal yang mirip dengan image produksi Amazon MWAA. Anda dapat menggunakan ini untuk menjalankan lingkungan Apache Airflow lokal untuk mengembangkan dan DAGs menguji, plugin kustom, dan dependensi sebelum menerapkan ke Amazon MWAA.
- Untuk menjalankan CLI, lihat pada. aws-mwaa-docker-images GitHub

Langkah kedua: Buat **`requirements.txt`**

Bagian berikut menjelaskan cara menentukan dependensi Python dari Indeks Paket Python dalam sebuah file. `requirements.txt`

Apache Airflow v3

1. Uji secara lokal. Tambahkan pustaka tambahan secara iteratif untuk menemukan kombinasi paket dan versinya yang tepat, sebelum membuat `requirements.txt` file. Untuk menjalankan utilitas Amazon MWAA CLI, lihat pada. aws-mwaa-docker-images GitHub
2. Tinjau paket tambahan Apache Airflow. Untuk mengakses daftar paket yang diinstal untuk Apache Airflow v3 di Amazon MWAA, lihat di situs web. aws-mwaa-docker-images `requirements.txt` GitHub
3. Tambahkan pernyataan kendala. Tambahkan file kendala untuk lingkungan Apache Airflow v3 Anda di bagian atas file Anda. `requirements.txt` File batasan Apache Airflow menentukan versi penyedia yang tersedia pada saat rilis Apache Airflow.

Dalam contoh berikut, ganti *{environment-version}* dengan nomor versi lingkungan Anda, dan *{Python-version}* dengan versi Python yang kompatibel dengan lingkungan Anda.

Untuk informasi tentang versi Python yang kompatibel dengan lingkungan Apache Airflow Anda, lihat Apache Airflow Versions.

```
--constraint "https://raw.githubusercontent.com/apache/airflow/
constraints-{Airflow-version}/constraints-{Python-version}.txt"
```

Jika file kendala menentukan bahwa `xyz==1.0` paket tidak kompatibel dengan paket lain di lingkungan Anda, `pip3 install` akan gagal mencegah pustaka yang tidak kompatibel diinstal ke lingkungan Anda. Jika instalasi gagal untuk paket apa pun, Anda dapat mengakses log kesalahan untuk setiap komponen Apache Airflow (penjadwal, pekerja, dan server web) di aliran log yang sesuai di Log. CloudWatch Untuk informasi selengkapnya tentang jenis log, lihatthe section called “Mengakses log Aliran Udara”.

4. Paket Apache Airflow. Tambahkan paket ekstra dan versi (==). Ini membantu mencegah paket dengan nama yang sama, tetapi versi yang berbeda, diinstal di lingkungan Anda.

```
apache-airflow[package-extra]==2.5.1
```

5. Pustaka Python. Tambahkan nama paket dan versi (==) di `requirements.txt` file Anda. Ini membantu mencegah pembaruan yang melanggar future PyPidari.org agar tidak diterapkan secara otomatis.

```
library == version
```

Example Boto3 dan psycopg2-biner

Contoh ini disediakan untuk tujuan demonstrasi. Pustaka biner boto dan psycopg2 disertakan dengan instalasi dasar untuk Apache Airflow v3 dan tidak perlu ditentukan dalam file. `requirements.txt`

```
boto3==1.17.54
boto==2.49.0
botocore==1.20.54
psycopg2-binary==2.8.6
```

Jika paket ditentukan tanpa versi, Amazon MWAA menginstal versi terbaru paket dari .org. PyPi Versi ini mungkin bertentangan dengan paket lain di Anda`requirements.txt`.

Apache Airflow v2

1. Uji secara lokal. Tambahkan pustaka tambahan secara iteratif untuk menemukan kombinasi paket dan versinya yang tepat, sebelum membuat `requirements.txt` file. Untuk menjalankan utilitas Amazon MWAA CLI, lihat pada. aws-mwaa-docker-images GitHub
2. Tinjau paket tambahan Apache Airflow. Untuk mengakses daftar paket yang diinstal untuk Apache Airflow v2 di Amazon MWAA, akses di situs web. aws-mwaa-docker-images `requirements.txt` GitHub
3. Tambahkan pernyataan kendala. Tambahkan file kendala untuk lingkungan Apache Airflow v2 Anda di bagian atas file Anda. `requirements.txt` File batasan Apache Airflow menentukan versi penyedia yang tersedia pada saat rilis Apache Airflow.

   Dimulai dengan Apache Airflow v2.7.2, file persyaratan Anda harus menyertakan pernyataan. `--constraint` Jika Anda tidak memberikan kendala, Amazon MWAA akan menentukan satu untuk Anda untuk memastikan paket yang tercantum dalam persyaratan Anda kompatibel dengan versi Apache Airflow yang Anda gunakan.

   Dalam contoh berikut, ganti *`{environment-version}`* dengan nomor versi lingkungan Anda, dan *`{Python-version}`* dengan versi Python yang kompatibel dengan lingkungan Anda.

   Untuk informasi tentang versi Python yang kompatibel dengan lingkungan Apache Airflow Anda, lihat Apache Airflow Versions.

   ```
   --constraint "https://raw.githubusercontent.com/apache/airflow/
   constraints-{Airflow-version}/constraints-{Python-version}.txt"
   ```

   Jika file kendala menentukan bahwa `xyz==1.0` paket tidak kompatibel dengan paket lain di lingkungan Anda, `pip3 install` akan gagal mencegah pustaka yang tidak kompatibel diinstal ke lingkungan Anda. Jika instalasi gagal untuk paket apa pun, Anda dapat mengakses log kesalahan untuk setiap komponen Apache Airflow (penjadwal, pekerja, dan server web) di aliran log yang sesuai di Log. CloudWatch Untuk informasi selengkapnya tentang jenis log, lihatthe section called “Mengakses log Aliran Udara”.

4. Paket Apache Airflow. Tambahkan paket ekstra dan versi (==). Ini membantu mencegah paket dengan nama yang sama, tetapi versi yang berbeda, diinstal di lingkungan Anda.

```
apache-airflow[package-extra]==2.5.1
```

5. Pustaka Python. Tambahkan nama paket dan versi (==) di `requirements.txt` file Anda. Ini membantu mencegah pembaruan yang melanggar future PyPidari.org agar tidak diterapkan secara otomatis.

```
library == version
```

Example Boto3 dan psycopg2-biner

Contoh ini disediakan untuk tujuan demonstrasi. Pustaka biner boto dan psycopg2 disertakan dengan instalasi dasar Apache Airflow v2 dan tidak perlu ditentukan dalam file. `requirements.txt`

```
boto3==1.17.54
boto==2.49.0
botocore==1.20.54
psycopg2-binary==2.8.6
```

Jika paket ditentukan tanpa versi, Amazon MWAA menginstal versi terbaru paket dari .org. PyPi Versi ini mungkin bertentangan dengan paket lain di Anda`requirements.txt`.

Mengunggah **`requirements.txt`** ke Amazon S3

Anda dapat menggunakan konsol Amazon S3 atau AWS Command Line Interface (AWS CLI) untuk mengunggah `requirements.txt` file ke bucket Amazon S3 Anda.

Menggunakan AWS CLI

The AWS Command Line Interface (AWS CLI) adalah alat open source yang dapat Anda gunakan untuk berinteraksi dengan AWS layanan menggunakan perintah di shell baris perintah Anda. Untuk menyelesaikan langkah-langkah di halaman ini, Anda memerlukan yang berikut:

- AWS CLI — Instal versi 2.
- AWS CLI - Konfigurasi cepat dengan `aws configure`.

Untuk mengunggah menggunakan AWS CLI

1. Gunakan perintah berikut untuk membuat daftar semua bucket Amazon S3 Anda.

```
aws s3 ls
```

2. Gunakan perintah berikut untuk mencantumkan file dan folder di bucket Amazon S3 untuk lingkungan Anda.

```
aws s3 ls s3://YOUR_S3_BUCKET_NAME
```

3. Perintah berikut mengunggah `requirements.txt` file ke bucket Amazon S3.

```
aws s3 cp requirements.txt s3://amzn-s3-demo-bucket/requirements.txt
```

Menggunakan konsol Amazon S3

Konsol Amazon S3 adalah antarmuka pengguna berbasis web yang dapat Anda gunakan untuk membuat dan mengelola sumber daya di bucket Amazon S3 Anda.

Untuk mengunggah menggunakan konsol Amazon S3

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.
3. Pilih tautan bucket S3 di kode DAG di panel S3 untuk membuka bucket penyimpanan Anda di konsol.
4. Pilih Unggah.
5. Pilih Tambahkan file.
6. Pilih salinan lokal Anda`requirements.txt`, pilih Unggah.

Menginstal dependensi Python di lingkungan Anda

Bagian ini menjelaskan cara menginstal dependensi yang Anda unggah ke bucket Amazon S3 Anda dengan menentukan jalur ke file requirements.txt, dan menentukan versi file requirements.txt setiap kali diperbarui.

Menentukan jalur ke `requirements.txt` konsol Amazon MWAA (pertama kali)

Jika ini adalah pertama kalinya Anda membuat dan mengunggah `requirements.txt` ke bucket Amazon S3 Anda, Anda juga perlu menentukan jalur ke file di konsol Amazon MWAA. Anda hanya perlu menyelesaikan langkah ini sekali.

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.
3. Pilih Edit.
4. Pada kode DAG di panel Amazon S3, pilih Jelajahi S3 yang berdekatan dengan file Persyaratan - bidang opsional.
5. Pilih `requirements.txt` file di bucket Amazon S3 Anda.
6. Pilih Tutup.
7. Pilih Berikutnya, Perbarui lingkungan.

Anda dapat mulai menggunakan paket baru segera setelah lingkungan Anda selesai memperbarui.

Menentukan `requirements.txt` versi di konsol Amazon MWAA

Anda perlu menentukan versi `requirements.txt` file Anda di konsol Amazon MWAA setiap kali Anda mengunggah versi baru Anda `requirements.txt` di bucket Amazon S3 Anda.

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.
3. Pilih Edit.
4. Pada kode DAG di panel Amazon S3, pilih `requirements.txt` versi dalam daftar tarik-turun.
5. Pilih Berikutnya, Perbarui lingkungan.

Anda dapat mulai menggunakan paket baru segera setelah lingkungan Anda selesai memperbarui.

Mengakses log untuk `requirements.txt`

Anda dapat melihat log Apache Airflow untuk penjadwal yang menjadwalkan alur kerja Anda dan mengurai folder Anda. `dags` Langkah-langkah berikut menjelaskan cara membuka grup log untuk penjadwal di konsol Amazon MWAA, dan mengakses log Apache Airflow di konsol Log. CloudWatch

Untuk mengakses log untuk **`requirements.txt`**

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.
3. Pilih grup log penjadwal aliran udara di panel Pemantauan.
4. Pilih `requirements_install_ip` log masuk Aliran log.
5. Lihat daftar paket yang diinstal pada lingkungan di`/usr/local/airflow/.local/bin`. Contoh:

```
Collecting appdirs==1.4.4 (from -r /usr/local/airflow/.local/bin (line 1))
Downloading https://files.pythonhosted.org/
packages/3b/00/2344469e2084fb28kjdsfiuyweb47389789vxbmnbjhsdgf5463acd6cf5e3db69324/
appdirs-1.4.4-py2.py3-none-any.whl
Collecting astroid==2.4.2 (from -r /usr/local/airflow/.local/bin (line 2))
```

6. Tinjau daftar paket dan apakah salah satu dari ini mengalami kesalahan selama instalasi. Jika terjadi kesalahan, Anda mendapatkan kesalahan yang mirip dengan yang berikut ini:

```
2021-03-05T14:34:42.731-07:00
No matching distribution found for LibraryName==1.0.0 (from -r /usr/local/
airflow/.local/bin (line 4))
No matching distribution found for LibraryName==1.0.0 (from -r /usr/local/
airflow/.local/bin (line 4))
```

Apa selanjutnya?

Uji DAGs, plugin kustom, dan dependensi Python Anda secara lokal menggunakan on. aws-mwaa-docker-images GitHub

Menghapus file di Amazon S3

Halaman ini menjelaskan cara kerja pembuatan versi di bucket Amazon S3 untuk lingkungan Alur Kerja Terkelola Amazon untuk Apache Airflow, dan langkah-langkah untuk menghapus DAG,, atau file. `plugins.zip` `requirements.txt`

Daftar Isi

- Prasyarat

- Ikhtisar versi
- Cara kerjanya
- Menghapus DAG di Amazon S3
- Menghapus requirements.txt “saat ini” atau plugins.zip dari lingkungan
- Menghapus versi requirements.txt atau plugins.zip “non-current” (sebelumnya)
- Menggunakan siklus hidup untuk menghapus versi “tidak saat ini” (sebelumnya) dan menghapus penanda secara otomatis
- Contoh kebijakan siklus hidup untuk menghapus versi “tidak saat ini” requirements.txt dan menghapus penanda secara otomatis
- Apa selanjutnya?

Prasyarat

Anda akan memerlukan yang berikut ini sebelum Anda dapat menyelesaikan langkah-langkah di halaman ini.

- Izin — Anda Akun AWS harus telah diberikan akses oleh administrator Anda ke kebijakan kontrol MWAAFull ConsoleAccess akses Amazon untuk lingkungan Anda. Selain itu, lingkungan Amazon MWAA Anda harus diizinkan oleh peran eksekusi Anda untuk mengakses AWS sumber daya yang digunakan oleh lingkungan Anda.
- Akses — Jika Anda memerlukan akses ke repositori publik untuk menginstal dependensi langsung di server web, lingkungan Anda harus dikonfigurasi dengan akses server web jaringan publik. Untuk informasi lebih lanjut, lihatthe section called “Mode akses Apache Airflow”.
- Konfigurasi Amazon S3 - Bucket Amazon S3 yang digunakan untuk menyimpan plugin kustom DAGs Anda`plugins.zip`, dan dependensi Python harus dikonfigurasi dengan Akses Publik Diblokir dan `requirements.txt` Diaktifkan Versi.

Ikhtisar versi

`requirements.txt`Dan `plugins.zip` di bucket Amazon S3 Anda berversi. Saat pembuatan versi bucket Amazon S3 diaktifkan untuk objek, dan artefak (misalnya, plugins.zip) dihapus dari bucket Amazon S3, file tersebut tidak akan dihapus seluruhnya. Setiap kali artefak dihapus di Amazon S3, salinan baru dari file dibuat yaitu 404 (Object not found) error/0k file yang mengatakan. `I'm not here` Amazon S3 menyebut ini penanda hapus. Penanda hapus adalah versi “null” dari file dengan nama kunci (atau kunci) dan ID versi seperti objek lainnya.

Sebaiknya hapus versi file dan hapus spidol secara berkala untuk mengurangi biaya penyimpanan bucket Amazon S3 Anda. Untuk menghapus versi file “tidak saat ini” (sebelumnya) sepenuhnya, Anda harus menghapus versi file, dan kemudian penanda hapus untuk versi tersebut.

Cara kerjanya

Amazon MWAA menjalankan operasi sinkronisasi pada bucket Amazon S3 Anda setiap tiga puluh detik. Hal ini menyebabkan penghapusan DAG apa pun di bucket Amazon S3 disinkronkan ke gambar Aliran Udara wadah Fargate Anda.

Untuk `plugins.zip` dan `requirements.txt` file, perubahan hanya terjadi setelah pembaruan lingkungan saat Amazon MWAA membuat gambar Airflow baru dari wadah Fargate Anda dengan plugin khusus dan dependensi Python. Jika Anda menghapus versi saat ini dari salah satu `plugins.zip` file `requirements.txt` atau, dan kemudian memperbarui lingkungan Anda tanpa memberikan versi baru untuk file yang dihapus, maka pembaruan akan gagal dengan pesan kesalahan, seperti,`Unable to read version {version number} of file {file name}`.

Menghapus DAG di Amazon S3

File DAG (`.py`) tidak berversi dan dapat dihapus langsung di konsol Amazon S3. Langkah-langkah berikut menjelaskan cara menghapus DAG di bucket Amazon S3 Anda.

Untuk menghapus DAG

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.
3. Pilih tautan bucket S3 di kode DAG di panel S3 untuk membuka bucket penyimpanan Anda di konsol.
4. Pilih `dags` foldernya.
5. Pilih DAG, Hapus.
6. Di bawah Hapus objek? , ketik`delete`.
7. Pilih Hapus objek.

> **Note**
>
> Apache Airflow mempertahankan sejarah DAG run. Setelah DAG dijalankan di Apache Airflow, itu tetap dalam daftar Airflow terlepas dari status file, sampai Anda menghapusnya di

Apache DAGs Airflow. Untuk menghapus DAG di Apache Airflow, pilih tombol merah “hapus” di kolom Links.

Menghapus requirements.txt “saat ini” atau plugins.zip dari lingkungan

Saat ini, tidak ada cara untuk menghapus plugins.zip atau requirements.txt dari lingkungan setelah ditambahkan, tetapi kami sedang mengerjakan masalah ini. Untuk sementara, solusinya adalah menunjuk ke teks kosong atau file zip, masing-masing.

Menghapus versi requirements.txt atau plugins.zip “non-current” (sebelumnya)

`plugins.zip`File `requirements.txt` dan di bucket Amazon S3 Anda diversi di Amazon MWAA. Jika Anda ingin menghapus file-file ini di bucket Amazon S3 Anda sepenuhnya, Anda harus mengambil versi saat ini (121212) dari objek (misalnya, plugins.zip), menghapus versi, dan kemudian menghapus penanda hapus untuk versi file.

Anda juga dapat menghapus versi file “tidak saat ini” (sebelumnya) di konsol Amazon S3; namun, Anda masih perlu menghapus penanda hapus menggunakan salah satu opsi berikut.

- Untuk mengambil versi objek, lihat Mengambil versi objek dari bucket berkemampuan versi di panduan Amazon S3.
- Untuk menghapus versi objek, lihat Menghapus versi objek dari bucket berkemampuan versi di panduan Amazon S3.
- Untuk menghapus penanda hapus, lihat Mengelola penanda hapus di panduan Amazon S3.

Menggunakan siklus hidup untuk menghapus versi “tidak saat ini” (sebelumnya) dan menghapus penanda secara otomatis

Anda dapat mengonfigurasi kebijakan siklus hidup bucket Amazon S3 untuk menghapus versi plugins.zip dan requirements.txt versi “tidak terkini” (sebelumnya) di bucket Amazon S3 setelah beberapa hari tertentu, atau untuk menghapus penanda penghapusan objek yang kedaluwarsa.

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.

3. Di bawah kode DAG di Amazon S3, pilih bucket Amazon S3 Anda.
4. Pilih Buat aturan siklus hidup.

Contoh kebijakan siklus hidup untuk menghapus versi “tidak saat ini” requirements.txt dan menghapus penanda secara otomatis

Gunakan contoh berikut untuk membuat aturan siklus hidup yang secara permanen menghapus versi “tidak terkini” dari file requirements.txt dan penanda hapusnya setelah tiga puluh hari.

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.
3. Di bawah kode DAG di Amazon S3, pilih bucket Amazon S3 Anda.
4. Pilih Buat aturan siklus hidup.
5. Dalam nama aturan Siklus Hidup, ketik. `Delete previous requirements.txt versions and delete markers after thirty days`
6. Di Awalan, persyaratan.
7. Dalam tindakan aturan Siklus Hidup, pilih Hapus versi objek sebelumnya secara permanen dan Hapus penanda hapus kedaluwarsa atau unggahan multibagian yang tidak lengkap.
8. Dalam Jumlah hari setelah objek menjadi versi sebelumnya, ketik`30`.
9. Di penanda hapus objek kedaluwarsa, pilih Hapus penanda hapus objek kedaluwarsa, objek dihapus secara permanen setelah 30 hari.

Apa selanjutnya?

- Pelajari lebih lanjut tentang penanda hapus Amazon S3 di Mengelola penanda hapus.
- Pelajari selengkapnya tentang siklus hidup Amazon S3 di objek kedaluwarsa.

Jaringan

Panduan ini menjelaskan pengaturan jaringan VPC Amazon yang Anda perlukan untuk lingkungan Amazon MWAA.

Bagian-bagian

- Tentang jaringan di Amazon MWAA
- Keamanan di VPC Anda di Amazon MWAA
- Mengelola akses ke titik akhir VPC Amazon khusus layanan di Amazon MWAA
- Membuat titik akhir layanan VPC yang diperlukan di VPC Amazon dengan perutean pribadi
- Mengelola titik akhir VPC Amazon Anda sendiri di Amazon MWAA

Tentang jaringan di Amazon MWAA

VPC Amazon adalah jaringan virtual yang terhubung dengan Anda. Akun AWS Ini memberi Anda keamanan cloud dan kemampuan untuk menskalakan secara dinamis dengan memberikan kontrol halus atas infrastruktur virtual dan segmentasi lalu lintas jaringan Anda. Halaman ini menjelaskan infrastruktur VPC Amazon dengan perutean publik, perutean pribadi, atau perutean publik dan pribadi yang diperlukan untuk mendukung Alur Kerja Terkelola Amazon untuk lingkungan Apache Airflow.

Daftar Isi

- Istilah
- Apa yang didukung
- Ikhtisar infrastruktur VPC
  - Routing publik melalui internet
  - Perutean pribadi tanpa akses internet
- Contoh kasus penggunaan untuk VPC Amazon dan mode akses Apache Airflow
  - Akses internet diizinkan - jaringan VPC Amazon baru
  - Akses internet tidak diizinkan - jaringan VPC Amazon baru
  - Akses internet tidak diizinkan - jaringan VPC Amazon yang ada

Istilah

Perutean publik

Jaringan VPC Amazon yang memiliki akses ke internet.

Perutean pribadi

Jaringan VPC Amazon tanpa akses ke internet.

Apa yang didukung

Tabel berikut menjelaskan jenis Amazon VPC Amazon yang didukung Amazon MWAA.

Jenis Amazon VPC	Didukung	
VPC Amazon yang dimiliki oleh akun yang mencoba menciptakan lingkungan.	Ya	
VPC Amazon bersama tempat beberapa orang Akun AWS membuat sumber dayanya AWS .	Ya	

Ikhtisar infrastruktur VPC

Saat Anda membuat lingkungan Amazon MWAA, Amazon MWAA membuat antara satu hingga dua titik akhir VPC untuk lingkungan Anda berdasarkan mode akses Apache Airflow yang Anda pilih untuk lingkungan Anda. Titik akhir ini ditampilkan sebagai Antarmuka Jaringan Elastis (ENI) dengan IP pribadi di VPC Amazon Anda. Setelah titik akhir ini dibuat, lalu lintas apa pun yang ditujukan ke IP ini dialihkan secara pribadi atau publik ke layanan terkait AWS yang digunakan oleh lingkungan Anda.

Bagian berikut menjelaskan infrastruktur VPC Amazon yang diperlukan untuk merutekan lalu lintas secara publik melalui internet, atau secara pribadi dalam VPC Amazon Anda.

Routing publik melalui internet

Bagian ini menjelaskan infrastruktur VPC Amazon dari lingkungan dengan perutean publik. Anda memerlukan infrastruktur VPC berikut:

- Satu grup keamanan VPC. Grup keamanan VPC bertindak sebagai firewall virtual untuk mengontrol lalu lintas jaringan masuk (masuk) dan keluar (keluar) pada sebuah instance.
  - Hingga 5 grup keamanan dapat ditentukan.
  - Kelompok keamanan harus menentukan aturan masuk referensi sendiri untuk dirinya sendiri.
  - Grup keamanan harus menentukan aturan keluar untuk semua lalu lintas (`0.0.0.0/0`; untuk IPv6, gunakan). `::/0`
  - Kelompok keamanan harus mengizinkan semua lalu lintas dalam aturan referensi diri. Misalnya, (Disarankan) Contoh semua grup keamanan referensi mandiri akses.
  - Grup keamanan secara opsional dapat membatasi lalu lintas lebih lanjut dengan menentukan rentang port untuk rentang port HTTPS `443` dan rentang port TCP. `5432` Misalnya, (Opsional) Contoh grup keamanan yang membatasi akses masuk ke port 5432 dan (Opsional) Contoh grup keamanan yang membatasi akses masuk ke port 443.
- Dua subnet publik. Subnet publik adalah subnet yang dikaitkan dengan tabel rute yang memiliki rute ke gateway internet.
  - Diperlukan dua subnet publik. Hal ini memungkinkan Amazon MWAA untuk membuat image container baru untuk lingkungan Anda di zona ketersediaan Anda yang lain, jika satu container gagal.
  - Subnet harus berada di Availability Zone yang berbeda. Misalnya,`us-east-1a`,`us-east-1b`.
  - Subnet harus merutekan ke gateway NAT (atau instance NAT) dengan Alamat IP Elastis (EIP).
  - Subnet harus memiliki tabel rute yang mengarahkan lalu lintas internet ke gateway internet.
- Dua subnet pribadi. Subnet pribadi adalah subnet yang tidak terkait dengan tabel rute yang memiliki rute ke gateway internet.
  - Diperlukan dua subnet pribadi. Hal ini memungkinkan Amazon MWAA untuk membuat image container baru untuk lingkungan Anda di zona ketersediaan Anda yang lain, jika satu container gagal.
  - Subnet harus berada di Availability Zone yang berbeda. Misalnya,`us-east-1a`,`us-east-1b`.
  - Subnet harus memiliki tabel rute ke perangkat NAT (gateway atau instance).
  - Subnet tidak boleh rute ke gateway internet.
  - Setel `assignIpV6AddressOnCreation` ke `true` untuk subnet IPv6.

  - Untuk subnet pribadi IPv6, Anda harus memiliki koneksi ke gateway internet khusus egress-only (EIGW).
- Daftar kontrol akses jaringan (ACL). NACL mengelola (dengan mengizinkan atau menolak aturan) lalu lintas masuk dan keluar di tingkat subnet.
  - NACL harus memiliki aturan masuk yang memungkinkan semua lalu lintas (`0.0.0.0/0`; untuk IPv6, gunakan). `::/0`
  - NACL harus memiliki aturan keluar yang memungkinkan semua lalu lintas (`0.0.0.0/0`; untuk IPv6, gunakan). `::/0`
  - Misalnya, (Direkomendasikan) Contoh ACLs.
- Dua gateway NAT (atau contoh NAT). Perangkat NAT meneruskan lalu lintas dari instance di subnet pribadi ke internet atau AWS layanan lain, dan kemudian mengarahkan respons kembali ke instance.
  - Perangkat NAT harus dilampirkan ke subnet publik. (Satu perangkat NAT per subnet publik.)
  - Perangkat NAT harus memiliki Elastic IPv4 Address (EIP) yang terpasang pada setiap subnet publik.
- Sebuah gateway internet. Gateway internet menghubungkan VPC Amazon ke internet dan layanan lainnya AWS .
  - Gateway internet harus dilampirkan ke VPC Amazon.

Perutean pribadi tanpa akses internet

Bagian ini menjelaskan infrastruktur VPC Amazon dari lingkungan dengan perutean pribadi. Anda memerlukan infrastruktur VPC berikut:

- Satu grup keamanan VPC. Grup keamanan VPC bertindak sebagai firewall virtual untuk mengontrol lalu lintas jaringan masuk (masuk) dan keluar (keluar) pada sebuah instance.
  - Hingga 5 grup keamanan dapat ditentukan.
  - Kelompok keamanan harus menentukan aturan masuk referensi sendiri untuk dirinya sendiri.
  - Grup keamanan harus menentukan aturan keluar untuk semua lalu lintas (`0.0.0.0/0`; untuk IPv6, gunakan). `::/0`
  - Kelompok keamanan harus mengizinkan semua lalu lintas dalam aturan referensi diri. Misalnya, (Disarankan) Contoh semua grup keamanan referensi mandiri akses.
  - Grup keamanan secara opsional dapat membatasi lalu lintas lebih lanjut dengan menentukan rentang port untuk rentang port HTTPS `443` dan rentang port TCP. `5432` Misalnya, (Opsional)

Contoh grup keamanan yang membatasi akses masuk ke port 5432 dan (Opsional) Contoh grup keamanan yang membatasi akses masuk ke port 443.

- Dua subnet pribadi. Subnet pribadi adalah subnet yang tidak terkait dengan tabel rute yang memiliki rute ke gateway internet.
  - Diperlukan dua subnet pribadi. Hal ini memungkinkan Amazon MWAA untuk membuat image container baru untuk lingkungan Anda di zona ketersediaan Anda yang lain, jika satu container gagal.
  - Subnet harus berada di Availability Zone yang berbeda. Misalnya,`us-east-1a`,`us-east-1b`.
  - Subnet harus memiliki tabel rute ke titik akhir VPC Anda.
  - Subnet harus memiliki tabel rute ke EIGW untuk diunduh dari internet sebagai bagian dari DAG.
  - Subnet tidak boleh memiliki tabel rute ke perangkat NAT (gateway atau instance), atau gateway internet.
- Daftar kontrol akses jaringan (ACL). NACL mengelola (dengan mengizinkan atau menolak aturan) lalu lintas masuk dan keluar di tingkat subnet.
  - NACL harus memiliki aturan masuk yang memungkinkan semua lalu lintas (`0.0.0.0/0`; untuk IPv6, gunakan). `::/0`
  - NACL harus memiliki aturan keluar yang menyangkal semua lalu lintas (`0.0.0.0/0`; untuk IPv6, gunakan). `::/0`
  - Misalnya, (Direkomendasikan) Contoh ACLs.
- Tabel rute lokal. Tabel rute lokal adalah rute default untuk komunikasi dalam VPC.
  - Tabel rute lokal harus dikaitkan dengan subnet pribadi Anda.
  - Tabel rute lokal harus mengaktifkan instance di VPC Anda untuk berkomunikasi dengan jaringan Anda sendiri. Misalnya, jika Anda menggunakan untuk mengakses titik akhir antarmuka VPC AWS Client VPN untuk server web Apache Airflow Anda, tabel rute harus merutekan ke titik akhir VPC.
- Titik akhir VPC untuk setiap AWS layanan yang digunakan oleh lingkungan Anda, dan titik akhir VPC Apache Airflow di VPC Amazon yang sama dan Wilayah AWS Amazon sebagai lingkungan Amazon MWAA Anda.
  - Titik akhir VPC untuk setiap AWS layanan yang digunakan oleh lingkungan dan titik akhir VPC untuk Apache Airflow. Misalnya, (Diperlukan) Titik akhir VPC.
  - Titik akhir VPC harus mengaktifkan DNS pribadi.
  - Endpoint VPC harus dikaitkan dengan dua subnet pribadi lingkungan Anda.
  - Titik akhir VPC harus dikaitkan dengan grup keamanan lingkungan Anda.

- Kebijakan titik akhir VPC untuk setiap titik akhir harus dikonfigurasi untuk memungkinkan akses ke AWS layanan yang digunakan oleh lingkungan. Misalnya, (Disarankan) Contoh kebijakan titik akhir VPC untuk mengizinkan semua akses.
- Kebijakan titik akhir VPC untuk Amazon S3 harus dikonfigurasi untuk mengizinkan akses bucket. Misalnya, (Disarankan) Contoh kebijakan titik akhir gateway Amazon S3 untuk mengizinkan akses bucket.

Contoh kasus penggunaan untuk VPC Amazon dan mode akses Apache Airflow

Bagian ini menjelaskan berbagai kasus penggunaan untuk akses jaringan di VPC Amazon Anda dan mode akses server web Apache Airflow yang dipilih di konsol Amazon MWAA.

Akses internet diizinkan - jaringan VPC Amazon baru

Jika akses internet di VPC Anda diizinkan oleh organisasi Anda, dan Anda ingin pengguna mengakses server web Apache Airflow Anda melalui internet:

1. Buat jaringan VPC Amazon dengan akses internet.
2. Buat lingkungan dengan mode akses jaringan Publik untuk server web Apache Airflow Anda.
3. Yang kami rekomendasikan: Sebaiknya gunakan templat CloudFormation mulai cepat yang membuat infrastruktur VPC Amazon, bucket Amazon S3, dan lingkungan Amazon MWAA secara bersamaan. Untuk mempelajari lebih lanjut, lihatTutorial mulai cepat untuk Alur Kerja Terkelola Amazon untuk Apache Airflow.

Jika akses internet di VPC Anda diizinkan oleh organisasi Anda, dan Anda ingin membatasi akses server web Apache Airflow ke pengguna dalam VPC Anda:

1. Buat jaringan VPC Amazon dengan akses internet.
2. Buat mekanisme untuk mengakses titik akhir antarmuka VPC untuk server web Apache Airflow Anda dari komputer Anda.
3. Buat lingkungan dengan mode akses jaringan Pribadi untuk server web Apache Airflow Anda.
4. Apa yang kami rekomendasikan:

a. Sebaiknya gunakan konsol Amazon MWAA diOpsi satu: Membuat jaringan VPC di konsol Amazon MWAA, atau CloudFormation templat di. Opsi dua: Membuat jaringan VPC Amazon dengan akses internet

b. Kami merekomendasikan untuk mengonfigurasi akses menggunakan server web AWS Client VPN Apache Airflow Anda di. Tutorial: Mengkonfigurasi akses jaringan pribadi menggunakan AWS Client VPN

Akses internet tidak diizinkan - jaringan VPC Amazon baru

Jika akses internet di VPC Anda tidak diizinkan oleh organisasi Anda, dan Anda ingin server web Apache Airflow tetap dapat diakses melalui internet:

1. Buat jaringan VPC Amazon tanpa akses internet.
2. Buat lingkungan dengan mode akses jaringan publik dan pribadi untuk server web Apache Airflow Anda. Opsi ini tersedia untuk Apache Airflow versi 3.2.1 dan yang lebih baru.
3. Apa yang kami rekomendasikan: Kami merekomendasikan menggunakan CloudFormation template untuk membuat VPC Amazon tanpa akses internet dan titik akhir VPC untuk setiap layanan yang AWS digunakan oleh Amazon MWAA di. Opsi tiga: Membuat jaringan VPC Amazon tanpa akses internet

Jika akses internet di VPC Anda tidak diizinkan oleh organisasi Anda, dan Anda ingin membatasi akses server web Apache Airflow ke pengguna dalam VPC Anda:

1. Buat jaringan VPC Amazon tanpa akses internet.
2. Buat mekanisme untuk mengakses titik akhir antarmuka VPC untuk server web Apache Airflow Anda dari komputer Anda.
3. Buat titik akhir VPC untuk setiap AWS layanan yang digunakan oleh lingkungan Anda.
4. Buat lingkungan dengan mode akses jaringan Pribadi untuk server web Apache Airflow Anda.
5. Apa yang kami rekomendasikan:

   a. Sebaiknya gunakan CloudFormation template untuk membuat VPC Amazon tanpa akses internet dan titik akhir VPC untuk setiap layanan yang AWS digunakan oleh Amazon MWAA di. Opsi tiga: Membuat jaringan VPC Amazon tanpa akses internet

b. Kami merekomendasikan untuk mengonfigurasi akses menggunakan server web AWS Client VPN Apache Airflow Anda di. Tutorial: Mengkonfigurasi akses jaringan pribadi menggunakan AWS Client VPN

Akses internet tidak diizinkan - jaringan VPC Amazon yang ada

Jika akses internet di VPC Anda tidak diizinkan oleh organisasi Anda, dan Anda sudah memiliki jaringan VPC Amazon yang diperlukan tanpa akses internet:

1. Buat titik akhir VPC untuk setiap AWS layanan yang digunakan oleh lingkungan Anda.
2. Buat titik akhir VPC untuk Apache Airflow.
3. Buat mekanisme untuk mengakses titik akhir antarmuka VPC untuk server web Apache Airflow Anda dari komputer Anda.
4. Buat lingkungan dengan mode akses jaringan Pribadi untuk server web Apache Airflow Anda.
5. Apa yang kami rekomendasikan:

   a. Kami merekomendasikan untuk membuat dan melampirkan titik akhir VPC yang diperlukan untuk AWS setiap layanan yang digunakan oleh Amazon MWAA, dan titik akhir VPC yang diperlukan untuk Apache Airflow. Membuat titik akhir layanan VPC yang diperlukan di VPC Amazon dengan perutean pribadi

   b. Kami merekomendasikan untuk mengonfigurasi akses menggunakan server web AWS Client VPN Apache Airflow Anda di. Tutorial: Mengkonfigurasi akses jaringan pribadi menggunakan AWS Client VPN

Keamanan di VPC Anda di Amazon MWAA

Halaman ini menjelaskan komponen Amazon VPC yang digunakan untuk mengamankan Alur Kerja Terkelola Amazon untuk lingkungan Apache Airflow dan konfigurasi yang diperlukan untuk komponen ini.

Daftar Isi

- Ketentuan
- Ikhtisar keamanan
- Daftar kontrol akses jaringan (ACLs)
  - (Direkomendasikan) Contoh ACLs

- Grup keamanan VPC
  - (Disarankan) Contoh semua grup keamanan referensi mandiri akses
  - (Opsional) Contoh grup keamanan yang membatasi akses masuk ke port 5432
  - (Opsional) Contoh grup keamanan yang membatasi akses masuk ke port 443
- Kebijakan titik akhir VPC (hanya perutean pribadi)
  - (Disarankan) Contoh kebijakan titik akhir VPC untuk mengizinkan semua akses
  - (Disarankan) Contoh kebijakan titik akhir gateway Amazon S3 untuk mengizinkan akses bucket

Ketentuan

Perutean publik

Jaringan VPC Amazon yang memiliki akses ke internet.

Perutean pribadi

Jaringan VPC Amazon tanpa akses ke internet.

Ikhtisar keamanan

Grup keamanan dan daftar kontrol akses (ACLs) menyediakan cara untuk mengontrol lalu lintas jaringan di seluruh subnet dan instance di VPC Amazon Anda menggunakan aturan yang Anda tentukan.

- Lalu lintas jaringan ke dan dari subnet dapat dikontrol oleh Access Control Lists (ACLs). Anda hanya perlu satu ACL, dan ACL yang sama dapat digunakan di beberapa lingkungan.
- Lalu lintas jaringan ke dan dari sebuah instans dapat dikontrol oleh grup keamanan Amazon VPC. Anda dapat menggunakan antara satu hingga lima grup keamanan per lingkungan.
- Lalu lintas jaringan ke dan dari sebuah instance juga dapat dikontrol oleh kebijakan titik akhir VPC. Jika akses internet dalam VPC Amazon Anda tidak diizinkan oleh organisasi Anda dan Anda menggunakan jaringan VPC Amazon dengan perutean pribadi, kebijakan titik akhir VPC diperlukan untuk titik akhir VPC dan titik akhir VPC Apache Airflow AWS .

Daftar kontrol akses jaringan (ACLs)

Daftar kontrol akses jaringan (ACL) dapat mengelola (dengan mengizinkan atau menolak aturan) lalu lintas masuk dan keluar di tingkat subnet. ACL adalah stateless, yang berarti bahwa aturan masuk dan keluar harus ditentukan secara terpisah dan eksplisit. Ini digunakan untuk menentukan jenis lalu lintas jaringan yang diizinkan masuk atau keluar dari instance dalam jaringan VPC.

Setiap VPC Amazon memiliki ACL default yang memungkinkan semua lalu lintas masuk dan keluar. Anda dapat mengedit aturan ACL default, atau membuat ACL kustom dan melampirkannya ke subnet Anda. Subnet hanya dapat memiliki satu ACL yang melekat padanya kapan saja, tetapi satu ACL dapat dilampirkan ke beberapa subnet.

(Direkomendasikan) Contoh ACLs

Contoh berikut menampilkan aturan ACL masuk dan keluar yang dapat digunakan untuk VPC Amazon dengan perutean publik atau perutean pribadi.

Nomor aturan	Tipe	Protokol	Rentang port	Sumber	Izinkan/Tolak
100	Semua IPv4 lalu lintas	Semua	Semua	0.0.0.0/0	Izinkan
*	Semua IPv4 lalu lintas	Semua	Semua	0.0.0.0/0	Menyangkal

Grup keamanan VPC

Grup keamanan VPC bertindak sebagai firewall virtual yang mengontrol lalu lintas jaringan pada tingkat instans. Grup keamanan bersifat stateful, yang berarti bahwa ketika koneksi masuk diizinkan, ia diizinkan untuk membalas. Ini digunakan untuk menentukan jenis lalu lintas jaringan yang diizinkan masuk dari instance dalam jaringan VPC.

Setiap Amazon VPC memiliki grup keamanan default. Secara default, ia tidak memiliki aturan masuk. Ini memiliki aturan keluar yang memungkinkan semua lalu lintas keluar. Anda dapat mengedit aturan grup keamanan default, atau membuat grup keamanan khusus dan melampirkannya ke VPC Amazon Anda. Di Amazon MWAA, Anda perlu mengonfigurasi aturan masuk dan keluar untuk mengarahkan lalu lintas di gateway NAT Anda.

(Disarankan) Contoh semua grup keamanan referensi mandiri akses

Contoh berikut menampilkan aturan grup keamanan masuk yang memungkinkan semua lalu lintas untuk VPC Amazon dengan perutean publik atau perutean pribadi. Kelompok keamanan dalam contoh ini adalah aturan referensi diri untuk dirinya sendiri.

Tipe	Protokol	Jenis Sumber	Sumber		
Semua Lalu lintas	Semua	Semua	sg-0909e8e81919/-kelompok my-mwaa-vpc-security		

Contoh berikut menampilkan aturan grup keamanan keluar.

Tipe	Protokol	Jenis Sumber	Sumber		
Semua Lalu lintas	Semua	Semua	0.0.0.0/0		

(Opsional) Contoh grup keamanan yang membatasi akses masuk ke port 5432

Contoh berikut menampilkan aturan grup keamanan masuk yang mengizinkan semua lalu lintas HTTPS di port 5432 untuk database metadata Amazon Aurora PostgreSQL (dimiliki oleh Amazon MWAA) untuk lingkungan Anda.

> **Note**
>
> Jika Anda memilih untuk membatasi lalu lintas menggunakan aturan ini, Anda harus menambahkan aturan lain untuk mengizinkan lalu lintas TCP pada port 443.

Tipe	Protokol	Rentang port	Jenis sumber	Sumber	

Tipe	Protokol	Rentang port	Jenis sumber	Sumber	
TCP Kustom	TCP	5432	Khusus	sg-0909e8e81919/-kelompok my-mwaa-vpc-security	

(Opsional) Contoh grup keamanan yang membatasi akses masuk ke port 443

Contoh berikut menampilkan aturan grup keamanan masuk yang mengizinkan semua lalu lintas TCP pada port 443 untuk server web Apache Airflow.

Tipe	Protokol	Rentang port	Jenis sumber	Sumber	
HTTPS	TCP	443	Khusus	sg-0909e8e81919/-kelompok my-mwaa-vpc-security	

Kebijakan titik akhir VPC (hanya perutean pribadi)

Kebijakan titik akhir (AWS PrivateLink) VPC mengontrol akses ke AWS layanan dari subnet pribadi Anda. Kebijakan titik akhir VPC adalah kebijakan sumber daya IAM yang Anda lampirkan ke gateway VPC atau titik akhir antarmuka. Bagian ini menjelaskan izin yang diperlukan untuk kebijakan titik akhir VPC untuk setiap titik akhir VPC.

Sebaiknya gunakan kebijakan titik akhir antarmuka VPC untuk setiap titik akhir VPC yang Anda buat yang memungkinkan akses penuh ke semua AWS layanan, dan menggunakan peran eksekusi Anda secara eksklusif untuk izin. AWS

(Disarankan) Contoh kebijakan titik akhir VPC untuk mengizinkan semua akses

Contoh berikut menyajikan kebijakan titik akhir antarmuka VPC untuk VPC Amazon dengan perutean pribadi.

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    }
  ]
}
```

(Disarankan) Contoh kebijakan titik akhir gateway Amazon S3 untuk mengizinkan akses bucket

Contoh berikut menyajikan kebijakan titik akhir gateway VPC yang menyediakan akses ke bucket Amazon S3 yang diperlukan untuk operasi ECR Amazon untuk VPC Amazon dengan perutean pribadi. Ini diperlukan agar gambar Amazon ECR Anda dapat diambil, selain bucket tempat file Anda DAGs dan file pendukung disimpan.

```
{
  "Statement": [
    {
      "Sid": "Access-to-specific-bucket-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::prod-us-east-1-starport-layer-bucket/*"]
    }
  ]
}
```

Mengelola akses ke titik akhir VPC Amazon khusus layanan di Amazon MWAA

Titik akhir VPC (AWS PrivateLink) yang dapat Anda gunakan untuk menghubungkan VPC Anda secara pribadi ke layanan yang di-host AWS tanpa memerlukan gateway internet, perangkat NAT, VPN, atau proxy firewall. Titik akhir ini adalah perangkat virtual yang dapat diskalakan secara

horizontal dan sangat tersedia yang memungkinkan komunikasi antara instance di VPC dan layanan Anda. AWS Halaman ini menjelaskan titik akhir VPC yang dibuat oleh Amazon MWAA, dan cara mengakses titik akhir VPC untuk server web Apache Airflow Anda jika Anda memilih mode akses jaringan pribadi di Alur Kerja Terkelola Amazon untuk Apache Airflow.

> **Note**
>
> Jika Anda memilih Akses jaringan publik dan pribadi, titik akhir VPC untuk server web dibuat dan dikelola secara otomatis oleh Amazon MWAA untuk konektivitas pekerja. Tidak diperlukan pengaturan titik akhir VPC tambahan, dan Anda tidak perlu mengonfigurasi akses ke titik akhir VPC untuk akses browser ke UI Apache Airflow.

Daftar Isi

- Harga
- Ikhtisar titik akhir VPC
  - Mode akses jaringan publik
  - Mode akses jaringan pribadi
- Izin untuk menggunakan yang lain AWS layanan
- Mengakses titik akhir VPC
  - Mengakses titik akhir VPC di konsol VPC Amazon
  - Mengidentifikasi alamat IP pribadi server web Apache Airflow Anda dan titik akhir VPC-nya
- Mengakses titik akhir VPC untuk server web Apache Airflow Anda (akses jaringan pribadi)
  - Menggunakan sebuah AWS Client VPN
  - Menggunakan Linux Bastion Host
  - Menggunakan Load Balancer (lanjutan)

Harga

- AWS PrivateLink Harga

Ikhtisar titik akhir VPC

Saat Anda membuat lingkungan Amazon MWAA, Amazon MWAA membuat antara satu hingga dua titik akhir VPC untuk lingkungan Anda. Titik akhir ini ditampilkan sebagai Antarmuka Jaringan Elastis (ENI) dengan IP pribadi di VPC Amazon Anda. Setelah titik akhir ini dibuat, lalu lintas apa pun yang ditujukan ke IP ini dialihkan secara pribadi atau publik ke layanan terkait AWS yang digunakan oleh lingkungan Anda.

Mode akses jaringan publik

Jika Anda memilih mode akses jaringan Publik untuk server web Apache Airflow Anda, lalu lintas jaringan dirutekan secara publik melalui internet.

- Amazon MWAA membuat titik akhir antarmuka VPC untuk database metadata Amazon Aurora PostgreSQL Anda. Titik akhir dibuat di Availability Zones yang dipetakan ke subnet pribadi Anda dan independen dari yang lain. Akun AWS
- Amazon MWAA kemudian mengikat alamat IP dari subnet pribadi Anda ke titik akhir antarmuka. Ini dirancang untuk mendukung praktik terbaik mengikat satu IP dari setiap Availability Zone dari VPC Amazon.

Mode akses jaringan pribadi

Jika Anda memilih mode akses jaringan Pribadi untuk server web Apache Airflow Anda, lalu lintas jaringan dirutekan secara pribadi dalam VPC Amazon Anda.

- Amazon MWAA membuat titik akhir antarmuka VPC untuk server web Apache Airflow Anda, dan titik akhir antarmuka untuk basis data metadata Amazon Aurora PostgreSQL Anda. Titik akhir dibuat di Availability Zones yang dipetakan ke subnet pribadi Anda dan independen dari yang lain. Akun AWS
- Amazon MWAA kemudian mengikat alamat IP dari subnet pribadi Anda ke titik akhir antarmuka. Ini dirancang untuk mendukung praktik terbaik mengikat satu IP dari setiap Availability Zone dari VPC Amazon.

Izin untuk menggunakan yang lain AWS layanan

Titik akhir antarmuka menggunakan peran eksekusi untuk lingkungan Anda di AWS Identity and Access Management (IAM) untuk mengelola izin ke AWS sumber daya yang digunakan oleh

lingkungan Anda. Karena semakin banyak AWS layanan diaktifkan untuk suatu lingkungan, setiap layanan mengharuskan Anda mengonfigurasi izin menggunakan peran eksekusi lingkungan Anda. Untuk menambahkan izin, lihat. Peran eksekusi Amazon MWAA

Jika Anda telah memilih mode akses jaringan Pribadi untuk server web Apache Airflow Anda, Anda juga harus mengizinkan izin dalam kebijakan titik akhir VPC untuk setiap titik akhir. Untuk mempelajari lebih lanjut, lihatthe section called “Kebijakan titik akhir VPC (hanya perutean pribadi)”.

Mengakses titik akhir VPC

Bagian ini menjelaskan cara mengakses titik akhir VPC yang dibuat oleh Amazon MWAA, dan cara mengidentifikasi alamat IP pribadi untuk titik akhir VPC Apache Airflow Anda.

Mengakses titik akhir VPC di konsol VPC Amazon

Bagian berikut menampilkan langkah-langkah untuk mengakses titik akhir VPC yang dibuat oleh Amazon MWAA, dan titik akhir VPC apa pun yang mungkin telah Anda buat jika Anda menggunakan perutean pribadi untuk VPC Amazon Anda.

Untuk mengakses titik akhir VPC

1. Buka halaman Endpoints di konsol VPC Amazon.
2. Pilih Anda Wilayah AWS.
3. Lihat titik akhir antarmuka VPC yang dibuat oleh Amazon MWAA, dan titik akhir VPC apa pun yang mungkin telah Anda buat jika Anda menggunakan perutean pribadi di VPC Amazon Anda.

Untuk mempelajari lebih lanjut tentang titik akhir layanan VPC yang diperlukan untuk VPC Amazon dengan perutean pribadi, lihat. Membuat titik akhir layanan VPC yang diperlukan di VPC Amazon dengan perutean pribadi

Mengidentifikasi alamat IP pribadi server web Apache Airflow Anda dan titik akhir VPC-nya

Langkah-langkah berikut menjelaskan cara mengambil nama host server web Apache Airflow Anda dan titik akhir antarmuka VPC-nya, dan alamat IP pribadinya.

1. Gunakan perintah berikut AWS Command Line Interface (AWS CLI) untuk mengambil nama host untuk server web Apache Airflow Anda.

```
aws mwaa get-environment --name YOUR_ENVIRONMENT_NAME --query
 'Environment.WebserverUrl'
```

Anda mendapatkan sesuatu yang mirip dengan respons berikut:

```
"99aa99aa-55aa-44a1-a91f-f4552cf4e2f5-vpce.c10.us-west-2.airflow.amazonaws.com"
```

2. Jalankan perintah dig pada nama host yang dikembalikan sebagai respons dari perintah sebelumnya. Contoh:

```
dig CNAME +short 99aa99aa-55aa-44a1-a91f-f4552cf4e2f5-vpce.c10.us-
west-2.airflow.amazonaws.com
```

Anda mendapatkan sesuatu yang mirip dengan respons berikut:

```
vpce-0699aa333a0a0a0-bf90xjtr.vpce-svc-00bb7c2ca2213bc37.us-
west-2.vpce.amazonaws.com.
```

3. Gunakan perintah berikut AWS Command Line Interface (AWS CLI) untuk mengambil nama DNS titik akhir VPC yang dikembalikan sebagai respons dari perintah sebelumnya. Contoh:

```
aws ec2 describe-vpc-endpoints | grep vpce-0699aa333a0a0a0-bf90xjtr.vpce-
svc-00bb7c2ca2213bc37.us-west-2.vpce.amazonaws.com.
```

Anda mendapatkan sesuatu yang mirip dengan respons berikut:

```
"DnsName": "vpce-066777a0a0a0-bf90xjtr.vpce-svc-00bb7c2ca2213bc37.us-
west-2.vpce.amazonaws.com",
```

4. Jalankan perintah nslookup atau dig pada nama host Apache Airflow Anda dan nama DNS titik akhir VPC-nya untuk mengambil alamat IP. Contoh:

```
dig +short YOUR_AIRFLOW_HOST_NAME YOUR_AIRFLOW_VPC_ENDPOINT_DNS
```

Anda mendapatkan sesuatu yang mirip dengan respons berikut:

```
192.0.5.1
192.0.6.1
```

Mengakses titik akhir VPC untuk server web Apache Airflow Anda (akses jaringan pribadi)

Jika Anda telah memilih mode akses jaringan Pribadi untuk server web Apache Airflow Anda, Anda harus membuat mekanisme untuk mengakses titik akhir antarmuka VPC untuk server web Apache Airflow Anda. Anda harus menggunakan VPC Amazon, grup keamanan VPC, dan subnet pribadi yang sama dengan lingkungan Amazon MWAA Anda untuk sumber daya ini.

Menggunakan sebuah AWS Client VPN

AWS Client VPN adalah layanan VPN berbasis klien terkelola yang dapat Anda gunakan untuk mengakses AWS sumber daya dan sumber daya Anda dengan aman di jaringan lokal Anda. Ini menyediakan koneksi TLS aman dari lokasi mana pun menggunakan klien OpenVPN.

Kami merekomendasikan mengikuti tutorial Amazon MWAA untuk mengonfigurasi Client VPN:. Tutorial: Mengkonfigurasi akses jaringan pribadi menggunakan AWS Client VPN

Menggunakan Linux Bastion Host

Sebuah bastion host adalah server yang tujuannya adalah untuk menyediakan akses ke jaringan pribadi dari jaringan eksternal, seperti melalui internet dari komputer Anda. Instans Linux berada dalam subnet publik, dan mereka diatur dengan grup keamanan yang memungkinkan akses SSH dari grup keamanan yang dilampirkan ke instans Amazon EC2 yang mendasari yang menjalankan host bastion.

Kami merekomendasikan mengikuti tutorial Amazon MWAA untuk mengonfigurasi Linux Bastion Host:. Tutorial: Mengkonfigurasi akses jaringan pribadi menggunakan Linux Bastion Host

Menggunakan Load Balancer (lanjutan)

Bagian berikut menampilkan konfigurasi yang perlu Anda terapkan ke Application Load Balancer.

1. Kelompok sasaran. Anda harus menggunakan grup target yang mengarah ke alamat IP pribadi untuk server web Apache Airflow Anda, dan titik akhir antarmuka VPC-nya. Kami merekomendasikan untuk menentukan kedua alamat IP pribadi sebagai target terdaftar Anda, karena hanya menggunakan satu dapat mengurangi ketersediaan. Untuk informasi lebih lanjut tentang cara mengidentifikasi alamat IP pribadi, lihatthe section called “Mengidentifikasi alamat IP pribadi server web Apache Airflow Anda dan titik akhir VPC-nya”.

2. Kode status. Kami merekomendasikan penggunaan `200` dan kode `302` status dalam pengaturan grup target Anda. Jika tidak, target mungkin ditandai sebagai tidak sehat jika titik akhir VPC untuk server web Apache Airflow merespons dengan kesalahan. `302 Redirect`
3. Pendengar HTTPS. Anda harus menentukan port target untuk server web Apache Airflow. Contoh:

Protokol	Port
HTTPS	443

4. ACM domain baru. Jika ingin mengaitkan SSL/TLS sertifikat AWS Certificate Manager, Anda harus membuat domain baru untuk pendengar HTTPS untuk penyeimbang beban Anda.
5. Wilayah sertifikat ACM. Jika Anda ingin mengaitkan SSL/TLS sertifikat AWS Certificate Manager, Anda harus mengunggah ke lingkungan yang Wilayah AWS sama dengan lingkungan Anda. Contoh:

    - Example wilayah untuk mengunggah sertifikat

    ```
    aws acm import-certificate --certificate fileb://Certificate.pem --certificate-
    chain fileb://CertificateChain.pem --private-key fileb://PrivateKey.pem --
    region us-west-2
    ```

Membuat titik akhir layanan VPC yang diperlukan di VPC Amazon dengan perutean pribadi

Jaringan VPC Amazon yang ada tanpa akses internet memerlukan endpoint layanan VPC tambahan () untuk AWS PrivateLink menggunakan Apache Airflow di Amazon Managed Workflow untuk Apache Airflow. Halaman ini menjelaskan titik akhir VPC yang diperlukan untuk AWS layanan yang digunakan oleh Amazon MWAA, titik akhir VPC yang diperlukan untuk Apache Airflow, dan cara membuat dan melampirkan titik akhir VPC ke VPC Amazon yang ada dengan perutean pribadi.

> **Note**
>
> Jika Anda memilih Akses jaringan publik dan pribadi, titik akhir VPC untuk server web dibuat dan dikelola secara otomatis oleh Amazon MWAA. Anda tidak perlu membuat titik akhir VPC untuk konektivitas server web Apache Airflow. Namun, jika VPC Amazon Anda tidak memiliki

akses internet, Anda masih memerlukan titik akhir VPC untuk AWS layanan lain (seperti Amazon S3, CloudWatch Log, SQS, KMS, dan Amazon ECR) seperti yang dijelaskan di halaman ini.

Daftar Isi

- Harga
- Jaringan pribadi dan perutean pribadi
- (Diperlukan) Titik akhir VPC
- Melampirkan titik akhir VPC yang diperlukan
  - Titik akhir VPC diperlukan untuk AWS layanan
  - Titik akhir VPC diperlukan untuk Apache Airflow
- (Opsional) Aktifkan alamat IP pribadi untuk titik akhir antarmuka VPC Amazon S3 Anda
  - Menggunakan Route 53
  - VPC dengan DNS khusus

Harga

- AWS PrivateLink Harga

Jaringan pribadi dan perutean pribadi

Mode akses jaringan pribadi membatasi akses ke UI Apache Airflow kepada pengguna dalam VPC Amazon Anda yang telah diberikan akses ke kebijakan IAM untuk lingkungan Anda.

Ketika Anda membuat lingkungan dengan akses jaringan Private, Anda harus mengemas semua dependensi Anda dalam arsip roda Python (`.whl`), lalu mereferensikan di file Anda. `.whl` `requirements.txt` Untuk petunjuk tentang pengemasan dan pemasangan dependensi Anda menggunakan roda, lihat Mengelola dependensi menggunakan roda Python.

Gambar berikut menggambarkan di mana menemukan opsi Jaringan pribadi di konsol Amazon MWAA.

- Perutean pribadi. VPC Amazon tanpa akses internet membatasi lalu lintas jaringan dalam VPC. Halaman ini mengasumsikan VPC Amazon Anda tidak memiliki akses internet dan memerlukan titik akhir VPC untuk AWS setiap layanan yang digunakan oleh lingkungan Anda, dan titik akhir VPC untuk Apache Airflow yang sama dan Amazon VPC sebagai lingkungan Amazon MWAA Anda. Wilayah AWS

(Diperlukan) Titik akhir VPC

Bagian berikut menampilkan titik akhir VPC yang diperlukan untuk VPC Amazon tanpa akses internet. Ini mencantumkan titik akhir VPC untuk setiap AWS layanan yang digunakan oleh Amazon MWAA, termasuk titik akhir VPC yang diperlukan untuk Apache Airflow.

```
com.amazonaws.us-east-1.s3
com.amazonaws.us-east-1.monitoring
com.amazonaws.us-east-1.logs
com.amazonaws.us-east-1.sqs
com.amazonaws.us-east-1.kms
```

> **Note**
>
> Saat menggunakan Transit Gateway atau perutean lain yang tidak langsung menuju titik akhir AWS API, kami sarankan Anda AWS PrivateLink untuk menambahkan subnet pribadi Amazon MWAA Anda untuk layanan berikut:
>
> - Amazon S3
> - Amazon SQS
> - CloudWatch Log
> - CloudWatch metrik
> - AWS KMS (jika berlaku)
>
> Ini memastikan bahwa lingkungan Amazon MWAA Anda dapat berkomunikasi dengan aman dan efisien dengan layanan ini tanpa merutekan lalu lintas melalui internet publik, sehingga meningkatkan keamanan dan kinerja.

Melampirkan titik akhir VPC yang diperlukan

Bagian ini menjelaskan langkah-langkah untuk melampirkan titik akhir VPC yang diperlukan untuk VPC Amazon dengan perutean pribadi.

Titik akhir VPC diperlukan untuk AWS layanan

Bagian berikut menampilkan langkah-langkah untuk melampirkan titik akhir VPC untuk AWS layanan yang digunakan oleh lingkungan ke VPC Amazon yang ada.

Untuk melampirkan titik akhir VPC ke subnet pribadi Anda

1. Buka halaman Endpoints di konsol VPC Amazon.
2. Pilih Anda Wilayah AWS.
3. Buat titik akhir untuk Amazon S3:
    a. Pilih Buat Titik Akhir.
    b. Di bidang Filter menurut atribut atau cari berdasarkan kata kunci teks, ketik: **`.s3`**, lalu tekan Enter pada keyboard Anda.
    c. Sebaiknya pilih titik akhir layanan yang terdaftar untuk tipe Gateway.

       Sebagai contoh, **`com.amazonaws.us-west-2.s3 amazon Gateway`**.
    d. Pilih VPC Amazon lingkungan Anda di VPC.
    e. Pastikan bahwa dua subnet pribadi Anda di Availability Zone yang berbeda dipilih, dan DNS pribadi diaktifkan dengan memilih Aktifkan nama DNS.
    f. Pilih grup keamanan Amazon VPC lingkungan Anda.
    g. Pilih Akses Penuh dalam Kebijakan.
    h. Pilih Buat titik akhir.
4. Buat titik akhir untuk CloudWatch Log:
    a. Pilih Buat Titik Akhir.
    b. Di bidang Filter menurut atribut atau cari berdasarkan kata kunci teks, ketik: **`.logs`**, lalu tekan Enter pada keyboard Anda.
    c. Pilih titik akhir layanan.
    d. Pilih VPC Amazon lingkungan Anda di VPC.

e. Pastikan bahwa dua subnet pribadi Anda di Availability Zone yang berbeda dipilih, dan Aktifkan nama DNS diaktifkan.

f. Pilih grup keamanan Amazon VPC lingkungan Anda.

g. Pilih Akses Penuh dalam Kebijakan.

h. Pilih Buat titik akhir.

5. Buat titik akhir untuk CloudWatch Pemantauan:

a. Pilih Buat Titik Akhir.

b. Di bidang Filter menurut atribut atau cari berdasarkan kata kunci teks, ketik: **`.monitoring`**, lalu tekan Enter pada keyboard Anda.

c. Pilih titik akhir layanan.

d. Pilih VPC Amazon lingkungan Anda di VPC.

e. Pastikan bahwa dua subnet pribadi Anda di Availability Zone yang berbeda dipilih, dan Aktifkan nama DNS diaktifkan.

f. Pilih grup keamanan Amazon VPC lingkungan Anda.

g. Pilih Akses Penuh dalam Kebijakan.

h. Pilih Buat titik akhir.

6. Buat titik akhir untuk Amazon SQS:

a. Pilih Buat Titik Akhir.

b. Di bidang Filter menurut atribut atau cari berdasarkan kata kunci teks, ketik: **`.sqs`**, lalu tekan Enter pada keyboard Anda.

c. Pilih titik akhir layanan.

d. Pilih VPC Amazon lingkungan Anda di VPC.

e. Pastikan bahwa dua subnet pribadi Anda di Availability Zone yang berbeda dipilih, dan Aktifkan nama DNS diaktifkan.

f. Pilih grup keamanan Amazon VPC lingkungan Anda.

g. Pilih Akses Penuh dalam Kebijakan.

h. Pilih Buat titik akhir.

7. Buat titik akhir untuk AWS KMS:

a. Pilih Buat Titik Akhir.

b. Di bidang Filter menurut atribut atau cari berdasarkan kata kunci teks, ketik: **`.kms`**, lalu tekan Enter pada keyboard Anda.

c. Pilih titik akhir layanan.

d. Pilih VPC Amazon lingkungan Anda di VPC.

e. Pastikan bahwa dua subnet pribadi Anda di Availability Zone yang berbeda dipilih, dan Aktifkan nama DNS diaktifkan.

f. Pilih grup keamanan Amazon VPC lingkungan Anda.

g. Pilih Akses Penuh dalam Kebijakan.

h. Pilih Buat titik akhir.

Titik akhir VPC diperlukan untuk Apache Airflow

Bagian berikut menampilkan langkah-langkah untuk melampirkan titik akhir VPC untuk Apache Airflow ke Amazon VPC yang ada.

Untuk melampirkan titik akhir VPC ke subnet pribadi Anda

1. Buka halaman Endpoints di konsol VPC Amazon.
2. Pilih Anda Wilayah AWS.
3. Buat titik akhir untuk Apache Airflow API:

   a. Pilih Buat Titik Akhir.

   b. Di bidang Filter menurut atribut atau cari berdasarkan kata kunci teks, ketik: **`.airflow.api`**, lalu tekan Enter pada keyboard Anda.

   c. Pilih titik akhir layanan.

   d. Pilih VPC Amazon lingkungan Anda di VPC.

   e. Pastikan bahwa dua subnet pribadi Anda di Availability Zone yang berbeda dipilih, dan Aktifkan nama DNS diaktifkan.

   f. Pilih grup keamanan Amazon VPC lingkungan Anda.

   g. Pilih Akses Penuh dalam Kebijakan.

   h. Pilih Buat titik akhir.

4. Buat titik akhir pertama untuk lingkungan Apache Airflow:

   a. Pilih Buat Titik Akhir.

b. Di bidang Filter menurut atribut atau cari berdasarkan kata kunci teks, ketik: **`.airflow.env`**, lalu tekan Enter pada keyboard Anda.

c. Pilih titik akhir layanan.

d. Pilih VPC Amazon lingkungan Anda di VPC.

e. Pastikan bahwa dua subnet pribadi Anda di Availability Zone yang berbeda dipilih, dan Aktifkan nama DNS diaktifkan.

f. Pilih grup keamanan Amazon VPC lingkungan Anda.

g. Pilih Akses Penuh dalam Kebijakan.

h. Pilih Buat titik akhir.

5. Buat titik akhir kedua untuk operasi Apache Airflow:

a. Pilih Buat Titik Akhir.

b. Di bidang Filter menurut atribut atau cari berdasarkan kata kunci teks, ketik: **`.airflow.ops`**, lalu tekan Enter pada keyboard Anda.

c. Pilih titik akhir layanan.

d. Pilih VPC Amazon lingkungan Anda di VPC.

e. Pastikan bahwa dua subnet pribadi Anda di Availability Zone yang berbeda dipilih, dan Aktifkan nama DNS diaktifkan.

f. Pilih grup keamanan Amazon VPC lingkungan Anda.

g. Pilih Akses Penuh dalam Kebijakan.

h. Pilih Buat titik akhir.

(Opsional) Aktifkan alamat IP pribadi untuk titik akhir antarmuka VPC Amazon S3 Anda

Titik akhir Antarmuka Amazon S3 tidak mendukung DNS pribadi. Permintaan endpoint S3 masih diselesaikan ke alamat IP publik. Untuk menyelesaikan alamat S3 ke alamat IP pribadi, Anda perlu menambahkan zona host pribadi di Route 53 untuk titik akhir regional S3.

Menggunakan Route 53

Bagian ini menjelaskan langkah-langkah untuk mengaktifkan alamat IP pribadi untuk titik akhir Antarmuka S3 menggunakan Route 53.

1. Buat Zona Dihosting Pribadi untuk titik akhir antarmuka VPC Amazon S3 Anda (seperti, s3.eu-west-1.amazonaws.com) dan kaitkan dengan VPC Amazon Anda.
2. Buat catatan ALIAS A untuk titik akhir antarmuka VPC Amazon S3 Anda (seperti, s3.eu-west-1.amazonaws.com) yang menyelesaikan nama DNS Endpoint Antarmuka VPC Anda.
3. Buat ALIAS Catatan wildcard untuk titik akhir antarmuka Amazon S3 Anda (seperti, *. s3.eu-west-1.amazonaws.com) yang menyelesaikan nama DNS Endpoint Antarmuka VPC.

VPC dengan DNS khusus

Jika Amazon VPC Anda menggunakan perutean DNS kustom, Anda perlu membuat perubahan pada resolver DNS Anda (bukan Route 53, biasanya instance EC2 yang menjalankan server DNS) dengan membuat catatan CNAME. Contoh:

```
Name: s3.us-west-2.amazonaws.com
Type: CNAME
Value:  *.vpce-0f67d23e37648915c-e2q2e2j3.s3.us-west-2.vpce.amazonaws.com
```

Mengelola titik akhir VPC Amazon Anda sendiri di Amazon MWAA

Amazon MWAA menggunakan titik akhir Amazon VPC untuk berintegrasi dengan berbagai AWS layanan yang diperlukan untuk menyiapkan lingkungan Apache Airflow. Pengelolaan titik akhir Anda sendiri memiliki dua kasus penggunaan utama:

1. Ini berarti Anda dapat membuat lingkungan Apache Airflow di VPC Amazon bersama saat Anda menggunakan file untuk mengelola beberapa dan berbagi sumber AWS Organizationsdaya. Akun AWS
2. Ini memungkinkan Anda menggunakan kebijakan akses yang lebih ketat dengan mempersempit izin Anda ke sumber daya tertentu yang menggunakan titik akhir Anda.

Jika Anda memilih untuk mengelola titik akhir VPC Anda sendiri, Anda bertanggung jawab untuk membuat titik akhir Anda sendiri untuk lingkungan RDS untuk database PostgreSQL dan untuk server web lingkungan.

Untuk informasi lebih lanjut tentang bagaimana Amazon MWAA menyebarkan Apache Airflow di cloud, lihat diagram arsitektur Amazon MWAA.

> **Important**
>
> Amazon MWAA tidak memvalidasi pilihan tipe alamat IP (`AddressType`) untuk titik akhir yang dikelola pelanggan, jadi pastikan Anda menentukan dengan benar `AddressType` (opsi yang valid adalah atau). IPv4 IPv6

Menciptakan lingkungan di VPC Amazon bersama

Jika Anda menggunakannya AWS Organizationsuntuk mengelola beberapa sumber daya berbagi, Anda dapat menggunakan titik akhir VPC Akun AWS yang dikelola pelanggan dengan Amazon MWAA untuk berbagi sumber daya lingkungan dengan akun lain di organisasi Anda.

Saat Anda mengonfigurasi akses VPC bersama, akun yang memiliki VPC Amazon (pemilik) utama membagikan dua subnet pribadi yang diperlukan oleh Amazon MWAA dengan akun lain (peserta) yang termasuk dalam organisasi yang sama. Akun peserta yang berbagi subnet tersebut dapat melihat, membuat, memodifikasi, dan menghapus lingkungan di VPC Amazon bersama.

Asumsikan Anda memiliki akun`0wner`, yang bertindak sebagai `Root` akun dalam organisasi dan memiliki sumber daya Amazon VPC, dan akun peserta`Participant`, anggota organisasi yang sama. Saat `Participant` membuat MWAA Amazon baru di Amazon VPC yang dibagikannya, `0wner` Amazon MWAA pertama-tama akan membuat sumber daya layanan VPC, lalu memasuki status hingga 72 jam. `PENDING`

Setelah status lingkungan berubah dari `CREATING` ke`PENDING`, kepala sekolah yang bertindak atas nama `0wner` menciptakan titik akhir yang diperlukan. Untuk melakukan ini, Amazon MWAA mencantumkan database dan titik akhir server web di konsol Amazon MWAA. Anda juga dapat memanggil tindakan `GetEnvironment`API untuk mendapatkan titik akhir layanan.

> **Note**
>
> Jika VPC Amazon yang Anda gunakan untuk berbagi sumber daya adalah VPC Amazon pribadi, Anda masih harus menyelesaikan langkah-langkah yang dijelaskan di. the section called “Mengelola akses ke titik akhir VPC” Topiknya mencakup pengaturan serangkaian titik akhir VPC Amazon yang berbeda terkait dengan AWS layanan lain yang AWS terintegrasi, seperti Amazon ECR, Amazon ECS, dan Amazon SQS. Layanan ini sangat penting dalam mengoperasikan, dan mengelola, lingkungan Apache Airflow Anda di cloud.

Prasyarat

Sebelum membuat lingkungan Amazon MWAA di VPC bersama, Anda memerlukan sumber daya berikut:

- Dan Akun AWS, `Owner` untuk digunakan sebagai akun yang memiliki Amazon VPC.
- Unit AWS Organizationsorganisasi, `MyOrganization` dibuat sebagai root.
- Yang kedua Akun AWS,`Participant`, di bawah `MyOrganization` untuk melayani akun peserta yang menciptakan lingkungan baru.

Selain itu, kami menyarankan Anda membiasakan diri dengan tanggung jawab dan izin untuk pemilik dan peserta saat berbagi sumber daya di Amazon VPC.

Buat Amazon VPC

Pertama, buat VPC Amazon baru yang akan dibagikan oleh pemilik dan akun peserta:

1. Masuk ke konsol menggunakan`Owner`, lalu, buka CloudFormation konsol. Gunakan template berikut untuk membuat tumpukan. Tumpukan ini menyediakan sejumlah sumber daya jaringan termasuk VPC Amazon, dan subnet yang akan dibagikan kedua akun dalam skenario ini.

```
AWSTemplateFormatVersion: "2010-09-09"
Description: >-
This template deploys a VPC, with a pair of public and private subnets spread
 across two Availability Zones. It deploys an internet gateway, with a default
 route on the public subnets. It deploys a pair of NAT gateways (one in each AZ),
 and default routes for them in the private subnets.
 Parameters:
   EnvironmentName:
     Description: An environment name that is prefixed to resource names
     Type: String
     Default: mwaa-
   VpcCIDR:
     Description: Please enter the IP range (CIDR notation) for this VPC
     Type: String
     Default: 10.192.0.0/16
   PublicSubnet1CIDR:
     Description: >-
     Please enter the IP range (CIDR notation) for the public subnet in the first
 Availability Zone
     Type: String
```

```
    Default: 10.192.10.0/24
  PublicSubnet2CIDR:
    Description: >-
    Please enter the IP range (CIDR notation) for the public subnet in the second
Availability Zone
    Type: String
    Default: 10.192.11.0/24
  PrivateSubnet1CIDR:
    Description: >-
    Please enter the IP range (CIDR notation) for the private subnet in the first
Availability Zone
    Type: String
    Default: 10.192.20.0/24
  PrivateSubnet2CIDR:
    Description: >-
    Please enter the IP range (CIDR notation) for the private subnet in the second
Availability Zone
    Type: String
    Default: 10.192.21.0/24
Resources:
  VPC:
    Type: 'AWS::EC2::VPC'
    Properties:
    CidrBlock: !Ref VpcCIDR
    EnableDnsSupport: true
    EnableDnsHostnames: true
    Tags:
      - Key: Name
      Value: !Ref EnvironmentName
  InternetGateway:
    Type: 'AWS::EC2::InternetGateway'
    Properties:
    Tags:
      - Key: Name
      Value: !Ref EnvironmentName
  InternetGatewayAttachment:
    Type: 'AWS::EC2::VPCGatewayAttachment'
    Properties:
      InternetGatewayId: !Ref InternetGateway
      VpcId: !Ref VPC
  PublicSubnet1:
    Type: 'AWS::EC2::Subnet'
    Properties:
      VpcId: !Ref VPC
```

```
      AvailabilityZone: !Select
        - 0
        - !GetAZs ''
      CidrBlock: !Ref PublicSubnet1CIDR
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
        Value: !Sub '${EnvironmentName} Public Subnet (AZ1)'
  PublicSubnet2:
    Type: 'AWS::EC2::Subnet'
    Properties:
     VpcId: !Ref VPC
      AvailabilityZone: !Select
        - 1
        - !GetAZs ''
      CidrBlock: !Ref PublicSubnet2CIDR
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
        Value: !Sub '${EnvironmentName} Public Subnet (AZ2)'
  PrivateSubnet1:
    Type: 'AWS::EC2::Subnet'
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select
        - 0
        - !GetAZs ''
      CidrBlock: !Ref PrivateSubnet1CIDR
      MapPublicIpOnLaunch: false
        Tags:
        - Key: Name
          Value: !Sub '${EnvironmentName} Private Subnet (AZ1)'
  PrivateSubnet2:
    Type: 'AWS::EC2::Subnet'
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select
        - 1
        - !GetAZs ''
      CidrBlock: !Ref PrivateSubnet2CIDR
      MapPublicIpOnLaunch: false
      Tags:
        - Key: Name
        Value: !Sub '${EnvironmentName} Private Subnet (AZ2)'
```

```
  NatGateway1EIP:
    Type: 'AWS::EC2::EIP'
    DependsOn: InternetGatewayAttachment
    Properties:
      Domain: vpc
  NatGateway2EIP:
    Type: 'AWS::EC2::EIP'
    DependsOn: InternetGatewayAttachment
    Properties:
      Domain: vpc
  NatGateway1:
    Type: 'AWS::EC2::NatGateway'
    Properties:
      AllocationId: !GetAtt NatGateway1EIP.AllocationId
      SubnetId: !Ref PublicSubnet1
  NatGateway2:
    Type: 'AWS::EC2::NatGateway'
    Properties:
      AllocationId: !GetAtt NatGateway2EIP.AllocationId
      SubnetId: !Ref PublicSubnet2
  PublicRouteTable:
    Type: 'AWS::EC2::RouteTable'
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
        Value: !Sub '${EnvironmentName} Public Routes'
  DefaultPublicRoute:
    Type: 'AWS::EC2::Route'
    DependsOn: InternetGatewayAttachment
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway
  PublicSubnet1RouteTableAssociation:
    Type: 'AWS::EC2::SubnetRouteTableAssociation'
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet1
  PublicSubnet2RouteTableAssociation:
    Type: 'AWS::EC2::SubnetRouteTableAssociation'
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet2
```

```
  PrivateRouteTable1:
    Type: 'AWS::EC2::RouteTable'
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
        Value: !Sub '${EnvironmentName} Private Routes (AZ1)'
  DefaultPrivateRoute1:
    Type: 'AWS::EC2::Route'
    Properties:
      RouteTableId: !Ref PrivateRouteTable1
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId: !Ref NatGateway1
  PrivateSubnet1RouteTableAssociation:
    Type: 'AWS::EC2::SubnetRouteTableAssociation'
    Properties:
      RouteTableId: !Ref PrivateRouteTable1
      SubnetId: !Ref PrivateSubnet1
  PrivateRouteTable2:
    Type: 'AWS::EC2::RouteTable'
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
        Value: !Sub '${EnvironmentName} Private Routes (AZ2)'
  DefaultPrivateRoute2:
    Type: 'AWS::EC2::Route'
    Properties:
      RouteTableId: !Ref PrivateRouteTable2
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId: !Ref NatGateway2
  PrivateSubnet2RouteTableAssociation:
    Type: 'AWS::EC2::SubnetRouteTableAssociation'
    Properties:
      RouteTableId: !Ref PrivateRouteTable2
      SubnetId: !Ref PrivateSubnet2
  SecurityGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupName: mwaa-security-group
      GroupDescription: Security group with a self-referencing inbound rule.
      VpcId: !Ref VPC
  SecurityGroupIngress:
    Type: 'AWS::EC2::SecurityGroupIngress'
```

```
    Properties:
      GroupId: !Ref SecurityGroup
      IpProtocol: '-1'
      SourceSecurityGroupId: !Ref SecurityGroup
      Outputs:
        VPC:
          Description: A reference to the created VPC
          Value: !Ref VPC
          PublicSubnets:
          Description: A list of the public subnets
          Value: !Join
            - ','
            - - !Ref PublicSubnet1
            - !Ref PublicSubnet2
        PrivateSubnets:
          Description: A list of the private subnets
          Value: !Join
            - ','
            - - !Ref PrivateSubnet1
            - !Ref PrivateSubnet2
        PublicSubnet1:
          Description: A reference to the public subnet in the 1st Availability
Zone
          Value: !Ref PublicSubnet1
        PublicSubnet2:
          Description: A reference to the public subnet in the 2nd Availability
Zone
          Value: !Ref PublicSubnet2
        PrivateSubnet1:
          Description: A reference to the private subnet in the 1st Availability
Zone
          Value: !Ref PrivateSubnet1
        PrivateSubnet2:
          Description: A reference to the private subnet in the 2nd Availability
Zone
          Value: !Ref PrivateSubnet2
        SecurityGroupIngress:
          Description: Security group with self-referencing inbound rule
          Value: !Ref SecurityGroupIngress
```

2. Setelah sumber daya VPC Amazon baru disediakan, navigasikan ke AWS Resource Access Manager konsol, lalu pilih Buat berbagi sumber daya.

3. Pilih subnet yang Anda buat pada langkah pertama dari daftar subnet yang tersedia yang dapat Anda bagikan. `Participant`

Buat lingkungan

Selesaikan langkah-langkah berikut untuk membuat lingkungan Amazon MWAA dengan titik akhir Amazon VPC yang dikelola pelanggan.

1. Masuk menggunakan`Participant`, dan buka konsol Amazon MWAA. Selesaikan Langkah satu: Tentukan detail untuk menentukan bucket Amazon S3, folder DAG, dan dependensi untuk lingkungan baru Anda. Untuk informasi lebih lanjut, lihat memulai.
2. Pada halaman Konfigurasi pengaturan lanjutan, di bawah Jaringan, pilih subnet dari VPC Amazon bersama.
3. Di bawah Manajemen titik akhir pilih PELANGGAN dari daftar dropdown.
4. Pertahankan default untuk opsi yang tersisa di halaman, lalu pilih Buat lingkungan di halaman Tinjauan dan buat.

Lingkungan dimulai dalam suatu `CREATING` keadaan, kemudian berubah menjadi`PENDING`. Saat lingkungannya`PENDING`, tuliskan nama layanan endpoint Database dan nama layanan endpoint server web (jika Anda menyiapkan server web pribadi) menggunakan konsol.

Saat Anda membuat lingkungan baru menggunakan konsol Amazon MWAA. Amazon MWAA membuat grup keamanan baru dengan aturan masuk dan keluar yang diperlukan. Tuliskan ID grup keamanan.

Di bagian selanjutnya, `0wner` akan menggunakan titik akhir layanan dan ID grup keamanan untuk membuat titik akhir VPC Amazon baru di VPC Amazon bersama.

Buat titik akhir Amazon VPC

Selesaikan langkah-langkah berikut untuk membuat titik akhir VPC Amazon yang diperlukan untuk lingkungan Anda.

1. Masuk ke Konsol Manajemen AWS penggunaan`0wner`, terbuka https://console.aws.amazon.com/vpc/.
2. Pilih Grup keamanan dari panel navigasi kiri, lalu buat grup keamanan baru di VPC Amazon bersama menggunakan aturan masuk dan keluar berikut:

	Tipe	Protokol	Jenis sumber	Sumber
Ke dalam	Semua Lalu lintas	Semua	Semua	Grup keamanan lingkungan Anda
Ke luar	Semua Lalu lintas	Semua	Semua	`0.0.0.0/0`

> **Warning**
>
> `Owner`Akun harus menyiapkan grup keamanan di `Owner` akun untuk memungkinkan lalu lintas dari lingkungan baru ke VPC Amazon bersama. Anda dapat melakukan ini dengan membuat grup keamanan baru di`Owner`, atau mengedit yang sudah ada.

3. Pilih Endpoint, lalu buat endpoint baru untuk database lingkungan dan server web (jika dalam mode pribadi) menggunakan nama layanan endpoint dari langkah sebelumnya. Pilih VPC Amazon bersama, subnet yang Anda gunakan untuk lingkungan, dan grup keamanan lingkungan.

Jika berhasil, lingkungan akan berubah dari `PENDING` belakang ke`CREATING`, lalu akhirnya menjadi`AVAILABLE`. Jika sudah`AVAILABLE`, Anda dapat masuk ke konsol Apache Airflow.

Pemecahan Masalah Amazon VPC Bersama

Gunakan referensi berikut untuk menyelesaikan masalah yang Anda temui saat membuat lingkungan di VPC Amazon bersama.

Lingkungan **`CREATE_FAILED`** setelah **`PENDING`** status

- Verifikasi `Owner` bahwa berbagi subnet dengan `Participant` menggunakan AWS Resource Access Manager.
- Verifikasi bahwa titik akhir VPC Amazon untuk database dan server web dibuat dalam subnet yang sama yang terkait dengan lingkungan.
- Verifikasi bahwa grup keamanan yang digunakan dengan titik akhir Anda memungkinkan lalu lintas dari grup keamanan yang digunakan untuk lingkungan. `Owner`Akun membuat aturan

yang mereferensikan grup keamanan `Participant` sebagai*123456789012*/*security-group-id*:.

Tipe	Protokol	Jenis sumber	Sumber
Semua Lalu lintas	Semua	Semua	*123456789012* /*sg-0909e8e81919*

Untuk informasi selengkapnya, lihat Tanggung jawab dan izin untuk pemilik dan peserta

Lingkungan terjebak dalam **`PENDING`** status

Verifikasi setiap status titik akhir VPC untuk memastikannya. `Available` Jika Anda mengonfigurasi lingkungan dengan server web pribadi, Anda juga harus membuat titik akhir untuk server web. Jika lingkungan macetPENDING, ini mungkin menunjukkan bahwa titik akhir server web pribadi hilang.

`The Vpc Endpoint Service '`*`vpce-service-name`*`' does not exist`Kesalahan yang diterima

Jika Anda merujuk ke kesalahan berikut, verifikasi bahwa akun yang membuat titik akhir di `Owner` akun yang memiliki VPC bersama:

```
ClientError: An error occurred (InvalidServiceName) when calling the
 CreateVpcEndpoint operation:

The Vpc Endpoint Service 'vpce-service-name' does not exist
```

Tutorial untuk Alur Kerja Terkelola Amazon untuk Apache Airflow

Panduan ini mencakup step-by-step tutorial untuk menggunakan dan mengonfigurasi Alur Kerja Terkelola Amazon untuk lingkungan Apache Airflow.

Topik

- Tutorial: Mengkonfigurasi akses jaringan pribadi menggunakan AWS Client VPN
- Tutorial: Mengkonfigurasi akses jaringan pribadi menggunakan Linux Bastion Host
- Tutorial: Membatasi akses pengguna Amazon MWAA ke subset DAGs
- Tutorial: Otomatiskan pengelolaan titik akhir lingkungan Anda sendiri di Amazon MWAA

Tutorial: Mengkonfigurasi akses jaringan pribadi menggunakan AWS Client VPN

Tutorial ini memandu Anda melalui langkah-langkah untuk membuat terowongan VPN dari komputer Anda ke server web Apache Airflow untuk Amazon Managed Workflow untuk lingkungan Apache Airflow Anda. Untuk terhubung ke internet melalui terowongan VPN, Anda harus terlebih dahulu membuat AWS Client VPN titik akhir. Setelah diatur, titik akhir Client VPN bertindak sebagai server VPN yang memungkinkan koneksi aman dari komputer Anda ke sumber daya di VPC Anda. Anda kemudian akan terhubung ke Client VPN dari komputer Anda menggunakan AWS Client VPN for Desktop.

> **Note**
>
> Tutorial ini berlaku untuk mode akses jaringan pribadi. Jika Anda memilih Akses jaringan publik dan pribadi, Anda tidak memerlukan VPN untuk mengakses Apache Airflow UI karena dapat diakses melalui internet.

Bagian-bagian

- Jaringan pribadi
- Kasus penggunaan

- Sebelum Anda mulai
- Tujuan
- (Opsional) Langkah satu: Identifikasi VPC, aturan CIDR, dan keamanan VPC Anda
- Langkah kedua: Buat server dan sertifikat klien
- Langkah ketiga: Simpan CloudFormation Template secara lokal
- Langkah keempat: Buat Client VPN CloudFormation tumpukan
- Langkah lima: Kaitkan subnet ke Client VPN Anda
- Langkah enam: Tambahkan aturan masuknya otorisasi ke Client VPN Anda
- Langkah tujuh: Unduh file konfigurasi titik akhir Client VPN
- Langkah delapan: Connect ke AWS Client VPN
- Apa selanjutnya?

Jaringan pribadi

Tutorial ini mengasumsikan Anda telah memilih mode akses jaringan Pribadi untuk server web Apache Airflow Anda.

Mode akses jaringan pribadi membatasi akses ke UI Apache Airflow kepada pengguna dalam VPC Amazon Anda yang telah diberikan akses ke kebijakan IAM untuk lingkungan Anda.

Ketika Anda membuat lingkungan dengan akses jaringan Private, Anda harus mengemas semua dependensi Anda dalam arsip roda Python (`.whl`), lalu mereferensikan di file Anda. `.whl` `requirements.txt` Untuk petunjuk tentang pengemasan dan pemasangan dependensi Anda menggunakan roda, lihat Mengelola dependensi menggunakan roda Python.

Gambar berikut menggambarkan di mana menemukan opsi Jaringan pribadi di konsol Amazon MWAA.

Kasus penggunaan

Anda dapat menggunakan tutorial ini sebelum atau setelah Anda membuat lingkungan Amazon MWAA. Anda harus menggunakan VPC Amazon, grup keamanan VPC, dan subnet pribadi yang sama dengan lingkungan Anda. Jika Anda menggunakan tutorial ini setelah Anda membuat

lingkungan Amazon MWAA, setelah Anda menyelesaikan langkah-langkahnya, Anda dapat kembali ke konsol Amazon MWAA dan mengubah mode akses server web Apache Airflow Anda ke Jaringan pribadi.

Sebelum Anda mulai

1. Periksa izin pengguna. Pastikan akun Anda di AWS Identity and Access Management (IAM) memiliki izin yang cukup untuk membuat dan mengelola sumber daya VPC.
2. Gunakan VPC Amazon MWAA Anda. Tutorial ini mengasumsikan bahwa Anda mengaitkan Client VPN ke VPC yang ada. VPC Amazon harus sama Wilayah AWS dengan lingkungan Amazon MWAA dan memiliki dua subnet pribadi. Jika Anda belum membuat VPC Amazon, gunakan CloudFormation template di. Opsi tiga: Membuat jaringan VPC Amazon tanpa akses internet

Tujuan

Dalam tutorial ini, Anda akan melakukan hal berikut:

1. Buat AWS Client VPN titik akhir menggunakan CloudFormation templat untuk VPC Amazon yang ada.
2. Hasilkan sertifikat dan kunci server dan klien, lalu unggah sertifikat server dan kunci ke AWS Certificate Manager Wilayah AWS sama dengan lingkungan Amazon MWAA.
3. Unduh dan modifikasi file konfigurasi titik akhir Client VPN untuk Client VPN Anda, dan gunakan file tersebut untuk membuat profil VPN untuk terhubung menggunakan Client VPN for Desktop.

(Opsional) Langkah satu: Identifikasi VPC, aturan CIDR, dan keamanan VPC Anda

Bagian berikut menjelaskan cara menemukan ID untuk VPC Amazon Anda, grup keamanan VPC, dan cara untuk mengidentifikasi aturan CIDR yang Anda perlukan untuk membuat Client VPN Anda di langkah selanjutnya.

Identifikasi aturan CIDR Anda

Bagian berikut menjelaskan cara mengidentifikasi aturan CIDR, yang Anda perlukan untuk membuat Client VPN Anda.

Untuk mengidentifikasi CIDR untuk Client VPN Anda

1. Buka halaman VPC Amazon Anda di konsol VPC Amazon.
2. Gunakan pemilih wilayah di bilah navigasi untuk memilih yang Wilayah AWS sama dengan lingkungan Amazon MWAA.
3. Pilih Amazon VPC Anda.
4. Dengan asumsi CIDR untuk subnet pribadi Anda adalah:

   - Subnet Pribadi 1:10.192.10.0 `/24`
   - Subnet Pribadi 2:10.192.11.0 `/24`

   Jika CIDR untuk VPC Amazon Anda adalah `/16` 10.192.0.0, maka Client IPv4 CIDR yang akan Anda tentukan untuk Client VPN Anda adalah 10.192.0.0. `/22`
5. Simpan nilai CIDR ini, dan nilai ID VPC Anda untuk langkah selanjutnya.

Identifikasi VPC dan grup keamanan Anda

Bagian berikut menjelaskan cara menemukan ID VPC Amazon dan grup keamanan Anda, yang Anda perlukan untuk membuat Client VPN Anda.

> **Note**
>
> Anda mungkin menggunakan lebih dari satu grup keamanan. Anda harus menentukan semua grup keamanan VPC Anda di langkah selanjutnya.

Untuk mengidentifikasi kelompok-kelompok keamanan

1. Buka halaman Grup Keamanan di konsol VPC Amazon.
2. Gunakan pemilih wilayah di bilah navigasi untuk memilih. Wilayah AWS
3. Cari VPC Amazon di ID VPC, dan identifikasi grup keamanan yang terkait dengan VPC.
4. Simpan ID grup keamanan dan VPC Anda untuk langkah selanjutnya.

Langkah kedua: Buat server dan sertifikat klien

Titik akhir Client VPN mendukung 1024-bit dan 2048-bit RSA kunci ukuran saja. Bagian berikut menjelaskan cara menggunakan OpenVPN easy-rsa untuk menghasilkan sertifikat dan kunci server dan klien, dan kemudian mengunggah sertifikat ke ACM menggunakan (). AWS Command Line Interface AWS CLI

Untuk membuat sertifikat klien

1. Ikuti langkah-langkah cepat ini untuk membuat dan mengunggah sertifikat ke ACM melalui otentikasi dan otorisasi Klien AWS CLI di: Otentikasi bersama.
2. Dalam langkah-langkah ini, Anda harus menentukan Wilayah AWS sama dengan lingkungan Amazon MWAA dalam AWS CLI perintah saat mengunggah server dan sertifikat klien Anda. Berikut beberapa contoh cara menentukan wilayah dalam perintah ini:

   a. Example wilayah untuk sertifikat server

   ```
   aws acm import-certificate --certificate fileb://server.crt --private-key
    fileb://server.key --certificate-chain fileb://ca.crt --region us-west-2
   ```

   b. Example wilayah untuk sertifikat klien

   ```
   aws acm import-certificate --certificate fileb://client1.domain.tld.crt
    --private-key fileb://client1.domain.tld.key --certificate-chain fileb://
   ca.crt --region us-west-2
   ```

   c. Setelah langkah-langkah ini, simpan nilai yang dikembalikan dalam AWS CLI respons untuk sertifikat server dan ARN sertifikat klien. Anda akan menentukan ARN ini di CloudFormation template Anda untuk membuat Client VPN.
3. Dalam langkah-langkah ini, sertifikat klien dan kunci pribadi disimpan ke komputer Anda. Berikut adalah contoh di mana menemukan kredensional ini:

   a. Example di macOS

   Di macOS, konten disimpan di. `/Users/your-user/custom_folder` Jika Anda mencantumkan semua (`ls -a`) isi direktori ini, Anda mendapatkan sesuatu yang mirip dengan berikut ini:

   ```
   .
   ..
   ```

```
ca.crt
client1.domain.tld.crt
client1.domain.tld.key
server.crt
server.key
```

b. Setelah langkah-langkah ini, simpan konten atau catat lokasi sertifikat klien`client1.domain.tld.crt`, dan kunci pribadi masuk`client1.domain.tld.key`. Anda akan menambahkan nilai-nilai ini ke file konfigurasi untuk Client VPN Anda.

Langkah ketiga: Simpan CloudFormation Template secara lokal

Bagian berikut berisi CloudFormation template untuk membuat Client VPN. Anda harus menentukan VPC Amazon, grup keamanan VPC, dan subnet pribadi yang sama dengan lingkungan Amazon MWAA Anda.

- Salin isi template berikut dan simpan secara lokal sebagai`mwaa_vpn_client.yaml`. Anda juga dapat mengunduh template.

  Gantikan nilai-nilai berikut:

  - **`YOUR_CLIENT_ROOT_CERTIFICATE_ARN`**— ARN untuk sertifikat client1.domain.tld Anda di. `ClientRootCertificateChainArn`
  - **`YOUR_SERVER_CERTIFICATE_ARN`**— ARN untuk sertifikat server Anda di. `ServerCertificateArn`
  - Aturan CIDR IPv4 Klien di. `ClientCidrBlock` Aturan CIDR `10.192.0.0/22` disediakan.
  - ID VPC Amazon Anda masuk. `VpcId` VPC disediakan. `vpc-010101010101`
  - ID grup keamanan VPC Anda masuk. `SecurityGroupIds` Kelompok keamanan `sg-0101010101` disediakan.

```
AWSTemplateFormatVersion: 2010-09-09
Description: This template deploys a VPN Client Endpoint.
Resources:
  ClientVpnEndpoint:
    Type: 'AWS::EC2::ClientVpnEndpoint'
    Properties:
      AuthenticationOptions:
        - Type: "certificate-authentication"
```

```
      MutualAuthentication:
        ClientRootCertificateChainArn: "YOUR_CLIENT_ROOT_CERTIFICATE_ARN"
    ClientCidrBlock: 10.192.0.0/22
    ClientConnectOptions:
      Enabled: false
    ConnectionLogOptions:
      Enabled: false
    Description: "MWAA Client VPN"
    DnsServers: []
    SecurityGroupIds:
      - sg-0101010101
    SelfServicePortal: ''
    ServerCertificateArn: "YOUR_SERVER_CERTIFICATE_ARN"
    SplitTunnel: true
    TagSpecifications:
      - ResourceType: "client-vpn-endpoint"
        Tags:
        - Key: Name
          Value: MWAA-Client-VPN
    TransportProtocol: udp
    VpcId: vpc-010101010101
    VpnPort: 443
```

Note

Jika Anda menggunakan lebih dari satu grup keamanan untuk lingkungan Anda, Anda dapat menentukan beberapa grup keamanan dalam format berikut:

```
SecurityGroupIds:
 - sg-0112233445566778b
 - sg-0223344556677889f
```

Langkah keempat: Buat Client VPN CloudFormation tumpukan

Untuk membuat AWS Client VPN

1. Buka konsol AWS CloudFormation.
2. Pilih Template sudah siap, Upload file template.

3. Pilih Pilih file, dan pilih `mwaa_vpn_client.yaml` file Anda.
4. Pilih Berikutnya, Berikutnya.
5. Pilih pengakuan, lalu pilih Buat tumpukan.

Langkah lima: Kaitkan subnet ke Client VPN Anda

Untuk mengaitkan subnet pribadi ke AWS Client VPN

1. Buka konsol Amazon VPC.
2. Pilih halaman Endpoint Client VPN.
3. Pilih Client VPN Anda, lalu pilih tab Asosiasi, Associate.
4. Pilih yang berikut dalam daftar dropdown:
   - VPC Amazon Anda di VPC.
   - Salah satu subnet pribadi Anda di Pilih subnet untuk diasosiasikan.
5. Pilih Kaitkan.

> **Note**
>
> Dibutuhkan beberapa menit untuk VPC dan subnet untuk dikaitkan dengan Client VPN.

Langkah enam: Tambahkan aturan masuknya otorisasi ke Client VPN Anda

Anda perlu menambahkan aturan masuknya otorisasi menggunakan aturan CIDR untuk VPC Anda ke Client VPN Anda. Jika Anda ingin mengotorisasi pengguna atau grup tertentu dari Grup Direktori Aktif atau Penyedia SAML-based Identitas (IDP) Anda, lihat aturan Otorisasi dalam panduan Client VPN.

Untuk menambahkan CIDR ke AWS Client VPN

1. Buka konsol Amazon VPC.
2. Pilih halaman Endpoint Client VPN.
3. Pilih Client VPN Anda, lalu pilih tab Otorisasi, Otorisasi Ingress.
4. Tentukan hal berikut:

- Aturan CIDR Amazon VPC Anda di jaringan Tujuan untuk mengaktifkan. Contoh:

```
10.192.0.0/16
```

- Pilih Izinkan akses ke semua pengguna di Akses hibah ke.
- Masukkan nama deskriptif di Deskripsi.

5. Pilih Tambahkan aturan Otorisasi.

> **Note**
>
> Bergantung pada komponen jaringan untuk VPC Amazon Anda, Anda mungkin juga perlu aturan masuknya otorisasi ini ke daftar kontrol akses jaringan (NACL) Anda.

Langkah tujuh: Unduh file konfigurasi titik akhir Client VPN

Untuk mengunduh file konfigurasi

1. Ikuti langkah-langkah cepat ini untuk mengunduh file konfigurasi Client VPN di Unduh file konfigurasi titik akhir Client VPN.
2. Dalam langkah-langkah ini, Anda diminta untuk menambahkan string ke nama DNS endpoint Client VPN Anda. Inilah contohnya:

   - Example nama DNS titik akhir

     Jika nama DNS titik akhir Client VPN Anda adalah:

     ```
     remote cvpn-endpoint-0909091212aaee1.prod.clientvpn.us-west-1.amazonaws.com 443
     ```

     Anda dapat menambahkan string untuk mengidentifikasi titik akhir Client VPN Anda seperti ini:

     ```
     remote mwaavpn.cvpn-endpoint-0909091212aaee1.prod.clientvpn.us-
     west-1.amazonaws.com 443
     ```

3. Dalam langkah-langkah ini, Anda diminta untuk menambahkan konten sertifikat klien antara kumpulan `<cert></cert>` tag baru dan isi kunci pribadi di antara kumpulan `<key></key>` tag baru. Inilah contohnya:

   a. Buka prompt perintah dan ubah direktori ke lokasi sertifikat klien dan kunci pribadi Anda.

   b. Example macOS client1.domain.tld.crt

      Untuk menampilkan konten `client1.domain.tld.crt` file di macOS, Anda dapat menggunakan. `cat client1.domain.tld.crt`

      Salin nilai dari terminal dan tempel `downloaded-client-config.ovpn` seperti ini:

```
ZZZ1111dddaBBB
-----END CERTIFICATE-----
</ca>
<cert>
-----BEGIN CERTIFICATE-----
YOUR client1.domain.tld.crt
-----END CERTIFICATE-----
</cert>
```

   c. Example macOS client1.domain.tld.key

      Untuk menampilkan isi`client1.domain.tld.key`, Anda dapat menggunakan`cat client1.domain.tld.key`.

      Salin nilai dari terminal dan tempel `downloaded-client-config.ovpn` seperti ini:

```
ZZZ1111dddaBBB
-----END CERTIFICATE-----
</ca>
<cert>
-----BEGIN CERTIFICATE-----
YOUR client1.domain.tld.crt
-----END CERTIFICATE-----
</cert>
<key>
-----BEGIN CERTIFICATE-----
YOUR client1.domain.tld.key
-----END CERTIFICATE-----
</key>
```

Langkah delapan: Connect ke AWS Client VPN

Klien untuk AWS Client VPN disediakan secara gratis. Anda dapat menghubungkan komputer Anda secara langsung AWS Client VPN untuk pengalaman VPN end-to-end.

Untuk terhubung ke Client VPN

1. Unduh dan instal AWS Client VPN untuk Desktop.
2. Buka AWS Client VPN.
3. Pilih File, Profil terkelola di menu klien VPN.
4. Pilih Tambahkan profil, lalu pilih`downloaded-client-config.ovpn`.
5. Masukkan nama deskriptif di Nama Tampilan.
6. Pilih Tambahkan profil, Selesai.
7. Pilih Hubungkan.

Setelah terhubung ke Client VPN, Anda harus memutuskan sambungan dari VPN lain untuk mengakses sumber daya apa pun di VPC Amazon Anda.

> **Note**
>
> Anda mungkin harus keluar dari klien, dan mulai lagi sebelum Anda dapat terhubung.

Apa selanjutnya?

- Pelajari cara membuat lingkungan Amazon MWAA di. Memulai dengan Amazon Managed Workflows for Apache Airflow Anda harus membuat lingkungan yang Wilayah AWS sama dengan Client VPN, dan menggunakan VPC, subnet pribadi, dan grup keamanan yang sama dengan Client VPN.

Tutorial: Mengkonfigurasi akses jaringan pribadi menggunakan Linux Bastion Host

Tutorial ini memandu Anda melalui langkah-langkah untuk membuat terowongan SSH dari komputer Anda ke server web Apache Airflow untuk Amazon Managed Workflow untuk lingkungan Apache

Airflow Anda. Ini mengasumsikan Anda telah membuat lingkungan Amazon MWAA. Setelah diatur, Linux Bastion Host bertindak sebagai server lompat yang memungkinkan koneksi aman dari komputer Anda ke sumber daya di VPC Anda. Anda kemudian akan menggunakan add-on manajemen proxy SOCKS untuk mengontrol pengaturan proxy di browser Anda untuk mengakses UI Apache Airflow Anda.

> **Note**
>
> Tutorial ini berlaku untuk mode akses jaringan pribadi. Jika Anda memilih Akses jaringan publik dan pribadi, Anda tidak memerlukan host benteng untuk mengakses UI Apache Airflow karena dapat diakses melalui internet.

Bagian-bagian

- Jaringan pribadi
- Kasus penggunaan
- Sebelum Anda mulai
- Tujuan
- Langkah satu: Buat instance bastion
- Langkah kedua: Buat terowongan ssh
- Langkah ketiga: Konfigurasikan grup keamanan bastion sebagai aturan masuk
- Langkah empat: Salin URL Apache Airflow
- Langkah lima: Konfigurasikan pengaturan proxy
- Langkah enam: Buka Apache Airflow UI
- Apa selanjutnya?

Jaringan pribadi

Tutorial ini mengasumsikan Anda telah memilih mode akses jaringan Pribadi untuk server web Apache Airflow Anda.

Mode akses jaringan pribadi membatasi akses ke UI Apache Airflow kepada pengguna dalam VPC Amazon Anda yang telah diberikan akses ke kebijakan IAM untuk lingkungan Anda.

Ketika Anda membuat lingkungan dengan akses jaringan Private, Anda harus mengemas semua dependensi Anda dalam arsip roda Python (`.whl`), lalu mereferensikan di file Anda. `.whl requirements.txt` Untuk petunjuk tentang pengemasan dan pemasangan dependensi Anda menggunakan roda, lihat Mengelola dependensi menggunakan roda Python.

Gambar berikut menggambarkan di mana menemukan opsi Jaringan pribadi di konsol Amazon MWAA.

Kasus penggunaan

Anda dapat menggunakan tutorial ini setelah Anda membuat lingkungan Amazon MWAA. Anda harus menggunakan VPC Amazon, grup keamanan VPC, dan subnet publik yang sama dengan lingkungan Anda.

Sebelum Anda mulai

1. Periksa izin pengguna. Pastikan akun Anda di AWS Identity and Access Management (IAM) memiliki izin yang cukup untuk membuat dan mengelola sumber daya VPC.
2. Gunakan VPC Amazon MWAA Anda. Tutorial ini mengasumsikan bahwa Anda mengaitkan host bastion ke VPC yang ada. VPC Amazon harus berada di wilayah yang sama dengan lingkungan Amazon MWAA Anda dan memiliki dua subnet pribadi, seperti yang didefinisikan dalam. Buat jaringan VPC
3. Buat kunci SSH Anda perlu membuat kunci SSH Amazon EC2 (.pem) di Wilayah yang sama dengan lingkungan Amazon MWAA Anda untuk terhubung ke server virtual. Jika Anda tidak memiliki kunci SSH, lihat Buat atau impor key pair di Panduan Pengguna Amazon EC2.

Tujuan

Dalam tutorial ini, Anda akan melakukan hal berikut:

1. Buat instance Linux Bastion Host menggunakan CloudFormation template untuk VPC yang ada.
2. Otorisasi lalu lintas masuk ke grup keamanan instans bastion menggunakan aturan ingress di port. 22
3. Otorisasi lalu lintas masuk dari grup keamanan lingkungan Amazon MWAA ke grup keamanan instans benteng.

4. Buat terowongan SSH ke instance bastion.
5. Instal dan konfigurasikan FoxyProxy add-on untuk browser Firefox untuk mengakses Apache Airflow UI.

Langkah satu: Buat instance bastion

Bagian berikut menjelaskan langkah-langkah untuk membuat instance bastion linux menggunakan CloudFormation template untuk VPC yang ada di CloudFormation konsol.

Untuk membuat Linux Bastion Host

1. Buka halaman Deploy Quick Start di CloudFormation konsol.
2. Gunakan pemilih wilayah di bilah navigasi untuk memilih yang Wilayah AWS sama dengan lingkungan Amazon MWAA Anda.
3. Pilih Berikutnya.
4. Masukkan nama di bidang teks nama Stack, sepertimwaa-linux-bastion.
5. Pada panel Parameter, Konfigurasi jaringan, pilih opsi berikut:
   a. Pilih ID VPC lingkungan Amazon MWAA Anda.
   b. Pilih ID subnet 1 Publik lingkungan Amazon MWAA Anda.
   c. Pilih ID subnet 2 Publik lingkungan Amazon MWAA Anda.
   d. Masukkan rentang alamat yang paling sempit (misalnya, rentang CIDR internal) di CIDR akses eksternal benteng yang diizinkan.

   > **Note**
   >
   > Cara termudah untuk mengidentifikasi rentang adalah dengan menggunakan rentang CIDR yang sama dengan subnet publik Anda. Misalnya, subnet publik dalam CloudFormation template pada Buat jaringan VPC halaman adalah `10.192.10.0/24` dan`10.192.11.0/24`.

6. Pada panel konfigurasi Amazon EC2, pilih yang berikut ini:
   a. Pilih kunci SSH Anda di daftar dropdown di Nama pasangan kunci.
   b. Masukkan nama di Nama Host Bastion.
   c. Pilih true untuk penerusan TCP.

> **Warning**
>
> Penerusan TCP harus disetel ke true pada langkah ini. Jika tidak, Anda tidak akan dapat membuat terowongan SSH di langkah berikutnya.

7. Pilih Berikutnya, Berikutnya.
8. Pilih pengakuan, lalu pilih Buat tumpukan.

Untuk mempelajari lebih lanjut tentang arsitektur Linux Bastion Host Anda, lihat Linux Bastion Hosts on the AWS Cloud: Architecture.

Langkah kedua: Buat terowongan ssh

Langkah-langkah berikut menjelaskan cara membuat terowongan ssh ke bastion linux Anda. Terowongan SSH menerima permintaan dari alamat IP lokal Anda ke bastion linux, itulah sebabnya penerusan TCP untuk benteng linux diatur ke langkah sebelumnya. `true`

macOS/Linux

Untuk membuat terowongan menggunakan baris perintah

1. Buka halaman Instans di konsol Amazon EC2.
2. Pilih instans.
3. Salin alamat di DNS IPv4 Publik. Misalnya, `ec2-4-82-142-1.compute-1.amazonaws.com`.
4. Di prompt perintah Anda, arahkan ke direktori tempat kunci SSH Anda disimpan.
5. Jalankan perintah berikut untuk terhubung ke instance bastion menggunakan ssh. Ganti nilai sampel dengan nama kunci SSH Anda di`mykeypair.pem`.

```
ssh -i mykeypair.pem -N -D 8157 ec2-user@YOUR_PUBLIC_IPV4_DNS
```

Windows (PuTTY)

Untuk membuat terowongan menggunakan PuTTY

1. Buka halaman Instans di konsol Amazon EC2.

2. Pilih instans.
3. Salin alamat di DNS IPv4 Publik. Misalnya, `ec2-4-82-142-1.compute-1.amazonaws.com`.
4. Buka PuTTY, pilih Session.
5. Masukkan nama host di Nama Host sebagai ec2-user@ ***YOUR_PUBLIC_IPV4_DNS*** dan port sebagai. `22`
6. Perluas tab SSH, pilih Auth. Dalam file Kunci Pribadi untuk otentikasi, pilih file “ppk” lokal Anda.
7. Di bawah SSH, pilih tab Tunnels, lalu pilih opsi Dinamis dan Otomatis.
8. Di Port Sumber, tambahkan `8157` port (atau port lain yang tidak digunakan), lalu biarkan port Tujuan kosong. Pilih Tambahkan.
9. Pilih tab Sesi dan masukkan nama sesi. Sebagai contoh, `SSH Tunnel`.
10. Pilih Simpan, Buka.

> **Note**
>
> Anda mungkin perlu memasukkan frasa lulus untuk kunci publik Anda.

> **Note**
>
> Jika Anda menerima `Permission denied (publickey)` kesalahan, sebaiknya gunakan AWSSupport-TroubleshootSSHalat ini, dan pilih Jalankan Otomasi ini (konsol) untuk memecahkan masalah pengaturan SSH Anda.

Langkah ketiga: Konfigurasikan grup keamanan bastion sebagai aturan masuk

Akses ke server dan akses internet reguler dari server diperbolehkan dengan grup keamanan pemeliharaan khusus yang terpasang pada server tersebut. Langkah-langkah berikut menjelaskan cara mengkonfigurasi grup keamanan bastion sebagai sumber lalu lintas masuk ke grup keamanan VPC lingkungan.

1. Buka halaman Lingkungan di konsol Amazon MWAA.

2. Pilih lingkungan.
3. Pada panel Networking, pilih grup keamanan VPC.
4. Pilih Edit aturan masuk.
5. Pilih Tambahkan aturan.
6. Pilih ID grup keamanan VPC Anda di daftar dropdown Sumber.
7. Biarkan opsi yang tersisa kosong, atau atur ke nilai defaultnya.
8. Pilih Simpan aturan.

Langkah empat: Salin URL Apache Airflow

Langkah-langkah berikut menjelaskan cara membuka konsol Amazon MWAA dan menyalin URL ke UI Apache Airflow.

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.
3. Salin URL di Airflow UI untuk langkah selanjutnya.

Langkah lima: Konfigurasikan pengaturan proxy

Jika Anda menggunakan terowongan SSH dengan penerusan port dinamis, Anda harus menggunakan add-on manajemen proksi SOCKS untuk mengendalikan pengaturan proksi di peramban Anda. Misalnya, Anda dapat menggunakan `--proxy-server` fitur Chromium untuk memulai sesi browser, atau menggunakan FoxyProxy ekstensi di browser Mozilla FireFox .

Opsi satu: Siapkan Terowongan SSH menggunakan penerusan port lokal

Jika Anda tidak ingin menggunakan proxy SOCKS, Anda dapat mengatur terowongan SSH menggunakan penerusan port lokal. Contoh perintah berikut mengakses antarmuka web Amazon ResourceManagerEC2 dengan meneruskan lalu lintas pada port lokal 8157.

1. Buka jendela prompt perintah baru.
2. Masukkan perintah berikut untuk membuka terowongan SSH.

```
ssh -i mykeypair.pem -N -L 8157:YOUR_VPC_ENDPOINT_ID-vpce.us-
east-1.airflow.amazonaws.com:443 ubuntu@YOUR_PUBLIC_IPV4_DNS.us-
east-1.compute.amazonaws.com
```

`-L`menandakan penggunaan penerusan port lokal yang dapat Anda gunakan untuk menentukan port lokal yang digunakan untuk meneruskan data ke port jarak jauh yang diidentifikasi pada server web lokal node.

3. Masukkan `http://localhost:8157/` di browser Anda.

> **Note**
>
> Anda mungkin perlu menggunakannya`https://localhost:8157/`.

Opsi dua: Proxy menggunakan baris perintah

Anda dapat menggunakan sebagian besar browser web untuk mengkonfigurasi proxy menggunakan baris perintah atau parameter konfigurasi. Misalnya, dengan Chromium Anda dapat memulai browser dengan perintah berikut:

```
chromium --proxy-server="socks5://localhost:8157"
```

Ini memulai sesi browser yang menggunakan terowongan ssh yang Anda buat di langkah sebelumnya untuk mem-proxy permintaannya. Anda dapat membuka URL lingkungan Amazon MWAA Pribadi Anda (dengan https://) sebagai berikut:

```
https://YOUR_VPC_ENDPOINT_ID-vpce.us-east-1.airflow.amazonaws.com/home.
```

Opsi tiga: Proksi yang digunakan FoxyProxy untuk Mozilla Firefox

Contoh berikut menunjukkan konfigurasi FoxyProxy Standar (versi 7.5.1) untuk Mozilla Firefox. FoxyProxy menyediakan satu set alat manajemen proxy. Ini memungkinkan Anda menggunakan server proxy untuk URL yang cocok dengan pola yang sesuai dengan domain yang digunakan oleh Apache Airflow UI.

1. Di Firefox, buka halaman ekstensi FoxyProxy Standar.
2. Pilih Tambahkan ke Firefox.
3. Pilih Tambahkan.
4. Pilih FoxyProxy ikon di bilah alat browser Anda, pilih Opsi.
5. Salin kode berikut dan simpan secara lokal sebagai`mwaa-proxy.json`. Ganti nilai sampel *YOUR_HOST_NAME* dengan URL Apache Airflow Anda.

```
{
  "e0b7kh1606694837384": {
    "type": 3,
    "color": "#66cc66",
    "title": "airflow",
    "active": true,
    "address": "localhost",
    "port": 8157,
    "proxyDNS": false,
    "username": "",
    "password": "",
    "whitePatterns": [
      {
        "title": "airflow-ui",
        "pattern": "YOUR_HOST_NAME",
        "type": 1,
        "protocols": 1,
        "active": true
      }
    ],
    "blackPatterns": [],
    "pacURL": "",
    "index": -1
  },
  "k20d21508277536715": {
    "active": true,
    "title": "Default",
    "notes": "These are the settings that are used when no patterns match a URL.",
    "color": "#0055E5",
    "type": 5,
    "whitePatterns": [
      {
        "title": "all URLs",
        "active": true,
        "pattern": "*",
        "type": 1,
        "protocols": 1
      }
    ],
    "blackPatterns": [],
      "index": 9007199254740991
  },
  "logging": {
```

```
    "active": true,
    "maxSize": 500
  },
  "mode": "patterns",
  "browserVersion": "82.0.3",
  "foxyProxyVersion": "7.5.1",
  "foxyProxyEdition": "standard"
}
```

6. Pada Pengaturan Impor dari FoxyProxy 6.0+ panel, pilih Pengaturan Impor dan pilih file. `mwaa-proxy.json`
7. Pilih OK.

Langkah enam: Buka Apache Airflow UI

Langkah-langkah berikut menjelaskan cara membuka UI Apache Airflow Anda.

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih Buka UI Aliran Udara.

Apa selanjutnya?

- Pelajari cara menjalankan perintah CLI Airflow di terowongan SSH ke host bastion di. Referensi perintah CLI Apache Airflow
- Pelajari cara mengunggah kode DAG ke bucket Amazon S3 Anda. Menambahkan atau memperbarui DAGs

Tutorial: Membatasi akses pengguna Amazon MWAA ke subset DAGs

Amazon MWAA mengelola akses ke lingkungan Anda dengan memetakan prinsipal IAM Anda ke satu atau beberapa peran default Apache Airflow. Gunakan tutorial berikut untuk membatasi pengguna Amazon MWAA individu untuk hanya mengakses dan berinteraksi dengan DAG tertentu atau satu set. DAGs

> **Note**
>
> Langkah-langkah dalam tutorial ini dapat diselesaikan menggunakan akses federasi, selama peran IAM dapat diasumsikan.

Topik

- Prasyarat
- Langkah pertama: Berikan akses server web Amazon MWAA ke kepala IAM Anda dengan peran default Apache Airflow. Public
- Langkah kedua: Buat peran kustom Apache Airflow baru
- Langkah ketiga: Tetapkan peran yang Anda buat untuk pengguna Amazon MWAA Anda
- Langkah selanjutnya
- Sumber daya terkait

Prasyarat

Untuk menyelesaikan langkah-langkah dalam tutorial ini, Anda memerlukan yang berikut:

- Lingkungan Amazon MWAA dengan banyak DAGs
- Prinsipal IAM, `Admin` dengan AdministratorAccessizin, dan pengguna IAM`MWAAUser`, sebagai prinsipal yang dapat Anda batasi akses DAG. Untuk informasi selengkapnya tentang peran admin, lihat fungsi pekerjaan Administrator di Panduan Pengguna IAM

  > **Note**
  >
  > Jangan melampirkan kebijakan izin langsung ke pengguna IAM Anda. Sebaiknya siapkan peran IAM yang dapat diasumsikan pengguna untuk mendapatkan akses sementara ke sumber daya Amazon MWAA Anda.

- AWS Command Line Interface versi 2 diinstal.

Langkah pertama: Berikan akses server web Amazon MWAA ke kepala IAM Anda dengan peran default Apache Airflow. **`Public`**

Untuk memberikan izin menggunakan Konsol Manajemen AWS

1. Masuk ke `Admin` peran Anda Akun AWS dan buka konsol IAM.
2. Di panel navigasi kiri, pilih Pengguna, lalu pilih pengguna Amazon MWAA IAM Anda dari tabel pengguna.
3. Pada halaman detail pengguna, di bawah Ringkasan, pilih tab Izin, lalu pilih Kebijakan izin untuk memperluas kartu dan pilih Tambahkan izin.
4. Di bagian Hibah izin, pilih Lampirkan kebijakan yang ada secara langsung, lalu pilih Buat kebijakan untuk membuat dan melampirkan kebijakan izin kustom Anda sendiri.
5. Pada halaman Buat kebijakan, pilih JSON, lalu salin dan tempel kebijakan izin JSON berikut di editor kebijakan. Kebijakan ini memberikan akses server web ke pengguna dengan peran Apache `Public` Airflow default.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "airflow:CreateWebLoginToken",
            "Resource": [
            "arn:aws:airflow:us-
east-1:111122223333:role/YOUR_ENVIRONMENT_NAME/Public"
            ]
        }
    ]
}
```

Langkah kedua: Buat peran kustom Apache Airflow baru

Untuk membuat peran baru menggunakan Apache Airflow UI

1. Menggunakan peran IAM administrator Anda, buka konsol Amazon MWAA dan luncurkan UI Apache Airflow lingkungan Anda.
2. Dari panel navigasi di bagian atas, arahkan kursor pada Keamanan untuk membuka daftar tarik-turun, lalu pilih Daftar Peran untuk mengakses peran Apache Airflow default.
3. Dari daftar peran, pilih Pengguna, lalu di awal halaman pilih Tindakan untuk membuka dropdown. Pilih Salin Peran, dan konfirmasikan Ok

   > **Note**
   >
   > Salin peran Ops atau Viewer untuk memberikan akses yang lebih atau lebih sedikit.

4. Temukan peran baru yang Anda buat di tabel dan pilih Edit catatan.
5. Pada halaman Edit Peran, lakukan hal berikut:

   - Untuk Nama, masukkan nama baru untuk peran di bidang teks. Misalnya, **`Restricted`**.
   - Untuk daftar Izin, hapus `can read on DAGs` dan`can edit on DAGs`, lalu tambahkan izin baca dan tulis untuk kumpulan yang ingin DAGs Anda berikan akses. Misalnya, untuk DAG,`example_dag.py`, tambahkan **`can read on DAG:`*`example_dag`*** dan**`can edit on DAG:`*`example_dag`***.

   Pilih Simpan. Sekarang Anda memiliki peran baru yang membatasi akses ke subset yang DAGs tersedia di lingkungan Amazon MWAA Anda. Anda dapat menetapkan peran ini ke pengguna Apache Airflow yang ada.

Langkah ketiga: Tetapkan peran yang Anda buat untuk pengguna Amazon MWAA Anda

Untuk menetapkan peran baru

1. Menggunakan kredensi akses untuk`MWAAUser`, jalankan perintah CLI berikut untuk mengambil URL server web lingkungan Anda.

```
aws mwaa get-environment --name YOUR_ENVIRONMENT_NAME | jq
 '.Environment.WebserverUrl'
```

Jika berhasil, Anda akan merujuk ke output berikut:

```
"ab1b2345-678a-90a1-a2aa-34a567a8a901.c13.us-west-2.airflow.amazonaws.com"
```

2. Dengan `MWAAUser` masuk ke Konsol Manajemen AWS, buka jendela browser baru dan akses yang berikut ini URI. Ganti `Webserver-URL` dengan informasi Anda.

```
https://<Webserver-URL>/home
```

Jika berhasil, Anda akan mendapatkan halaman `Forbidden` kesalahan karena `MWAAUser` belum diberikan izin untuk mengakses Apache Airflow UI.

3. Dengan `Admin` masuk ke Konsol Manajemen AWS, buka konsol Amazon MWAA lagi dan luncurkan UI Apache Airflow lingkungan Anda.
4. Dari dasbor UI, perluas dropdown Keamanan, dan kali ini pilih Daftar Pengguna.
5. Di tabel pengguna, temukan pengguna Apache Airflow baru dan pilih Edit record. Nama depan pengguna akan cocok dengan nama pengguna IAM Anda dalam pola berikut:`user/mwaa-user`.
6. Pada halaman Edit Pengguna, di bagian Peran, tambahkan peran kustom baru yang Anda buat, lalu pilih Simpan.

> **Note**
>
> Bidang Nama Belakang diperlukan, tetapi spasi memenuhi persyaratan.

`Public`Kepala IAM memberikan `MWAAUser` izin untuk mengakses Apache Airflow UI, sementara peran baru memberikan izin tambahan yang diperlukan untuk mendapatkannya. DAGs

> **Important**
>
> Salah satu dari 5 peran default (sepertiAdmin) yang tidak diizinkan oleh IAM yang ditambahkan menggunakan UI Apache Airflow akan dihapus pada login pengguna berikutnya.

Langkah selanjutnya

- Untuk mempelajari selengkapnya tentang mengelola akses ke lingkungan Amazon MWAA Anda, dan untuk mendapatkan contoh kebijakan JSON IAM yang dapat Anda gunakan untuk pengguna lingkungan Anda, lihat the section called “Mengakses lingkungan Amazon MWAA”

Sumber daya terkait

- Kontrol Akses (Dokumentasi Apache Airflow) - Pelajari lebih lanjut tentang peran default Apache Airflow di situs web dokumentasi Apache Airflow.

Tutorial: Otomatiskan pengelolaan titik akhir lingkungan Anda sendiri di Amazon MWAA

Jika Anda menggunakannya AWS Organizationsuntuk mengelola beberapa sumber daya berbagi Akun AWS itu, Amazon MWAA memungkinkan Anda membuat dan mengelola titik akhir VPC Amazon Anda sendiri. Ini berarti Anda dapat menggunakan kebijakan keamanan yang lebih ketat yang memungkinkan akses hanya ke sumber daya yang dibutuhkan oleh lingkungan Anda.

Saat Anda membuat lingkungan di VPC Amazon bersama, akun yang memiliki VPC Amazon (pemilik) utama membagikan dua subnet pribadi yang diperlukan oleh Amazon MWAA dengan akun lain (peserta) yang termasuk dalam organisasi yang sama. Akun peserta yang berbagi subnet tersebut kemudian dapat melihat, membuat, memodifikasi, dan menghapus lingkungan di VPC bersama.

Saat Anda membuat lingkungan di Amazon VPC bersama, atau dibatasi kebijakan, Amazon MWAA pertama-tama akan membuat sumber daya VPC layanan, lalu memasukkan `PENDING`status hingga 72 jam.

Ketika status lingkungan berubah dari `CREATING` ke`PENDING`, Amazon MWAA mengirimkan EventBridge pemberitahuan Amazon tentang perubahan status. Ini memungkinkan akun pemilik membuat titik akhir yang diperlukan atas nama peserta berdasarkan informasi layanan titik akhir dari konsol atau API Amazon MWAA, atau secara terprogram Berikut ini, kami membuat titik akhir VPC Amazon baru menggunakan fungsi Lambda dan aturan yang mendengarkan pemberitahuan perubahan status Amazon MWAA. EventBridge

Di sini, kami membuat titik akhir baru di VPC Amazon yang sama dengan lingkungan. Untuk menyiapkan VPC Amazon bersama, buat EventBridge aturan dan fungsi Lambda di akun pemilik, dan lingkungan Amazon MWAA di akun peserta.

Topik

- Prasyarat
- Buat Amazon VPC
- Buat fungsi Lambda
- Buat EventBridge aturan
- Buat lingkungan Amazon MWAA

Prasyarat

Untuk menyelesaikan langkah-langkah dalam tutorial ini, Anda memerlukan yang berikut:

- ...

Buat Amazon VPC

Gunakan CloudFormation template dan AWS CLI perintah berikut untuk membuat VPC Amazon baru. Template menyiapkan sumber daya VPC Amazon dan memodifikasi kebijakan titik akhir untuk membatasi akses ke antrian tertentu.

1. Unduh CloudFormation template, lalu unzip `.yml` file.
2. Di jendela prompt perintah baru, navigasikan ke folder tempat Anda menyimpan template, lalu gunakan `create-stack`untuk membuat tumpukan. `--template-body`Bendera menentukan jalur ke template.

```
aws cloudformation create-stack --stack-name stack-name --template-body file://cfn-vpc-private-network.yml
```

Di bagian selanjutnya, Anda akan membuat fungsi Lambda.

Buat fungsi Lambda

Gunakan kode Python berikut dan kebijakan IAM JSON untuk membuat fungsi Lambda baru dan peran eksekusi. Fungsi ini membuat titik akhir Amazon VPC untuk server web Apache Airflow pribadi dan antrian Amazon SQS. Amazon MWAA menggunakan Amazon SQS untuk mengantri tugas dengan Celery di antara beberapa pekerja saat menskalakan lingkungan Anda.

1. Unduh kode fungsi Python.
2. Unduh kebijakan izin IAM, lalu unzip file.
3. Buka prompt perintah, lalu arahkan ke folder tempat Anda menyimpan kebijakan izin JSON. Gunakan `create-role`perintah IAM untuk membuat peran baru.

   ```
   aws iam create-role --role-name function-role \
         --assume-role-policy-document file://lambda-mwaa-vpce-policy.json
   ```

   Perhatikan peran ARN dari respon. AWS CLI Pada langkah berikutnya, kami menentukan peran baru ini sebagai peran eksekusi fungsi menggunakan ARN-nya.
4. Arahkan ke folder tempat Anda menyimpan kode fungsi, lalu gunakan `create-function`perintah untuk membuat fungsi baru.

   ```
   aws lambda create-function --function-name mwaa-vpce-lambda \
   --zip-file file://mwaa-lambda-shared-vpc.zip --runtime python3.8 --role
    arn:aws:iam::123456789012:role/function-role --handler lambda_handler
   ```

   Perhatikan fungsi ARN dari respon. AWS CLI Pada langkah berikutnya kita menentukan ARN untuk mengkonfigurasi fungsi sebagai target untuk aturan baru EventBridge .

Di bagian berikutnya, Anda membuat EventBridge aturan yang memanggil fungsi ini ketika lingkungan memasuki `PENDING` status.

Buat EventBridge aturan

Lakukan hal berikut untuk membuat aturan baru yang mendengarkan notifikasi Amazon MWAA dan menargetkan fungsi Lambda baru Anda.

1. Gunakan EventBridge `put-rule` perintah untuk membuat EventBridge aturan baru.

   ```
   aws events put-rule --name "mwaa-lambda-rule" \
   --event-pattern "{\"source\":[\"aws.airflow\"],\"detail-type\":[\"MWAA Environment
    Status Change\"]}"
   ```

   Pola acara mendengarkan notifikasi yang dikirimkan Amazon MWAA setiap kali status lingkungan berubah.

   ```
   {
        "source": ["aws.airflow"],
        "detail-type": ["MWAA Environment Status Change"]
        }
   ```

2. Gunakan `put-targets` perintah untuk menambahkan fungsi Lambda sebagai target untuk aturan baru.

   ```
   aws events put-targets --rule "mwaa-lambda-rule" \
   --targets "Id"="1","Arn"="arn:aws:lambda:us-east-1:123456789012:function:mwaa-vpce-
   lambda"
   ```

Anda siap membuat lingkungan Amazon MWAA baru dengan titik akhir Amazon VPC yang dikelola pelanggan.

Buat lingkungan Amazon MWAA

Gunakan konsol Amazon MWAA untuk membuat lingkungan baru dengan titik akhir Amazon VPC yang dikelola pelanggan.

1. Buka konsol Amazon MWAA, dan pilih Buat lingkungan.
2. Untuk Nama masukkan nama yang unik.
3. Untuk versi Airflow pilih versi terbaru.

4. Pilih bucket Amazon S3 dan DAGs folder, seperti **`dags/`** untuk digunakan dengan lingkungan, lalu pilih Berikutnya.
5. Pada halaman Konfigurasi pengaturan lanjutan, lakukan hal berikut:
   a. Untuk Virtual Private Cloud, pilih VPC Amazon yang Anda buat di langkah sebelumnya.
   b. Untuk akses server web, pilih Jaringan publik (internet dapat diakses).
   c. Untuk grup Keamanan, pilih grup keamanan yang Anda buat CloudFormation. Karena grup keamanan untuk AWS PrivateLink titik akhir dari langkah sebelumnya adalah referensi sendiri, Anda harus memilih grup keamanan yang sama untuk lingkungan Anda.
   d. Untuk manajemen Endpoint, pilih Titik akhir yang dikelola Pelanggan.
6. Pertahankan pengaturan default yang tersisa, lalu pilih Berikutnya.
7. Tinjau pilihan Anda, lalu pilih Buat lingkungan.

> **Tip**
>
> Untuk informasi selengkapnya tentang menyiapkan lingkungan baru, lihat Memulai Amazon MWAA.

Saat lingkungannya`PENDING`, Amazon MWAA mengirimkan notifikasi yang cocok dengan pola acara yang Anda tetapkan untuk aturan Anda. Aturan tersebut memanggil fungsi Lambda Anda. Fungsi ini mem-parsing peristiwa notifikasi dan mendapatkan informasi titik akhir yang diperlukan untuk server web dan antrean Amazon SQS. Kemudian menciptakan titik akhir di VPC Amazon Anda.

Ketika titik akhir tersedia, Amazon MWAA melanjutkan pembuatan lingkungan Anda. Saat siap, status lingkungan berubah menjadi `AVAILABLE` dan Anda dapat mengakses server web Apache Airflow menggunakan konsol Amazon MWAA.

Contoh kode untuk Alur Kerja Terkelola Amazon untuk Apache Airflow

Panduan ini berisi contoh kode, termasuk DAGs dan plugin khusus, yang dapat Anda gunakan di lingkungan Alur Kerja Terkelola Amazon untuk Apache Airflow. Untuk lebih banyak contoh menggunakan Apache Airflow AWS dengan layanan, lihat direktori di repositori `dags`Apache Airflow. GitHub

Sampel

- Menggunakan DAG untuk mengimpor variabel di CLI
- Membuat koneksi SSH menggunakan SSHOperator
- Menggunakan kunci rahasia AWS Secrets Manager untuk koneksi Apache Airflow Snowflake
- Menggunakan DAG untuk menulis metrik khusus CloudWatch
- Pembersihan basis data Aurora PostgreSQL di lingkungan Amazon MWAA
- Mengekspor metadata lingkungan ke file CSV di Amazon S3
- Menggunakan kunci rahasia AWS Secrets Manager untuk variabel Apache Airflow
- Menggunakan kunci rahasia AWS Secrets Manager untuk koneksi Apache Airflow
- Membuat plugin khusus dengan Oracle
- Mengubah zona waktu DAG di Amazon MWAA
- Menyegarkan token CodeArtifact
- Membuat plugin khusus dengan Apache Hive dan Hadoop
- Membuat plugin khusus untuk Apache Airflow PythonVirtualenvOperator
- Memanggil DAGs dengan fungsi Lambda
- Memanggil DAGs di lingkungan MWAA Amazon yang berbeda
- Menggunakan Amazon MWAA dengan Amazon RDS untuk Microsoft SQL Server
- Menggunakan Amazon MWAA dengan Amazon EKS
- Menghubungkan ke Amazon ECS menggunakan ECSOperator
- Menggunakan dbt dengan Amazon MWAA
- AWS Blog dan Tutorial

Menggunakan DAG untuk mengimpor variabel di CLI

Contoh kode berikut mengimpor variabel menggunakan CLI di Alur Kerja Terkelola Amazon untuk Apache Airflow.

Topik

- Versi
- Prasyarat
- Izin
- Dependensi
- Contoh kode
- Apa selanjutnya?

Versi

Anda dapat menggunakan contoh kode pada halaman ini dengan Apache Airflow v2 di Python 3.10 dan Apache Airflowv3 di Python 3.11.

Prasyarat

Tidak ada izin tambahan yang diperlukan untuk menggunakan contoh kode di halaman ini.

Izin

Anda Akun AWS membutuhkan akses ke `AmazonMWAAAirflowCliAccess` kebijakan. Untuk mempelajari lebih lanjut, lihatKebijakan CLI Apache Airflow: AmazonMWAAAirflowCliAccess.

Dependensi

Untuk menggunakan contoh kode ini dengan Apache Airflow v2 dan yang lebih baru, tidak diperlukan dependensi tambahan. Gunakan aws-mwaa-docker-imagesuntuk menginstal Apache Airflow.

Contoh kode

Kode contoh berikut mengambil tiga input: nama lingkungan Amazon MWAA Anda (in`mwaa_env`), lingkungan Anda (in`aws_region`), dan file lokal yang berisi variabel yang ingin Anda impor (in). Wilayah AWS `var_file`

```
import boto3
import json
import requests
import base64
import getopt
import sys

argv = sys.argv[1:]
mwaa_env=''
aws_region=''
var_file=''

try:
    opts, args = getopt.getopt(argv, 'e:v:r:', ['environment', 'variable-
file','region'])
    #if len(opts) == 0 and len(opts) > 3:
    if len(opts) != 3:
        print ('Usage: -e MWAA environment -v variable file location and filename -r
 aws region')
    else:
        for opt, arg in opts:
            if opt in ("-e"):
                mwaa_env=arg
            elif opt in ("-r"):
                aws_region=arg
            elif opt in ("-v"):
                var_file=arg

        boto3.setup_default_session(region_name="{}".format(aws_region))
        mwaa_env_name = "{}".format(mwaa_env)

        client = boto3.client('mwaa')
        mwaa_cli_token = client.create_cli_token(
            Name=mwaa_env_name
        )

        with open ("{}".format(var_file), "r") as myfile:
            fileconf = myfile.read().replace('\n', '')

        json_dictionary = json.loads(fileconf)
        for key in json_dictionary:
            print(key, " ", json_dictionary[key])
            val = (key + " " + json_dictionary[key])
```

```
            mwaa_auth_token = 'Bearer ' + mwaa_cli_token['CliToken']
            mwaa_webserver_hostname = 'https://{0}/aws_mwaa/
cli'.format(mwaa_cli_token['WebServerHostname'])
            raw_data = "variables set {0}".format(val)
            mwaa_response = requests.post(
                mwaa_webserver_hostname,
                headers={
                    'Authorization': mwaa_auth_token,
                    'Content-Type': 'text/plain'
                    },
                data=raw_data
                )
            mwaa_std_err_message = base64.b64decode(mwaa_response.json()
['stderr']).decode('utf8')
            mwaa_std_out_message = base64.b64decode(mwaa_response.json()
['stdout']).decode('utf8')
            print(mwaa_response.status_code)
            print(mwaa_std_err_message)
            print(mwaa_std_out_message)

except:
    print('Use this script with the following options: -e MWAA environment -v variable
 file location and filename -r aws region')
    print("Unexpected error:", sys.exc_info()[0])
    sys.exit(2)
```

Apa selanjutnya?

- Pelajari cara mengunggah kode DAG dalam contoh ini ke `dags` folder di bucket Amazon S3 Anda. Menambahkan atau memperbarui DAGs

Membuat koneksi SSH menggunakan **`SSHOperator`**

Contoh berikut menjelaskan bagaimana Anda dapat menggunakan `SSHOperator` dalam grafik asiklik terarah (DAG) untuk menyambung ke EC2 instans Amazon jarak jauh dari lingkungan Alur Kerja Terkelola Amazon untuk Apache Airflow. Anda dapat menggunakan pendekatan serupa untuk terhubung ke instans jarak jauh apa pun dengan akses SSH.

Dalam contoh berikut, Anda mengunggah kunci rahasia SSH (`.pem`) ke `dags` direktori lingkungan Anda di Amazon S3. Kemudian, Anda menginstal dependensi yang diperlukan menggunakan

`requirements.txt` dan membuat koneksi Apache Airflow baru di UI. Terakhir, Anda menulis DAG yang membuat koneksi SSH ke instance jarak jauh.

Topik

- Versi
- Prasyarat
- Izin
- Persyaratan
- Salin kunci rahasia Anda ke Amazon S3
- Buat koneksi Apache Airflow baru
- Contoh kode

Versi

Anda dapat menggunakan contoh kode pada halaman ini dengan Apache Airflow v2 di Python 3.10 dan Apache Airflowv3 di Python 3.11.

Prasyarat

Untuk menggunakan kode sampel di halaman ini, Anda memerlukan yang berikut:

- Lingkungan Amazon MWAA.
- Kunci rahasia SSH. Contoh kode mengasumsikan Anda memiliki EC2 instans Amazon dan `.pem` di Wilayah yang sama dengan lingkungan Amazon MWAA Anda. Jika Anda tidak memiliki kunci, lihat Membuat atau mengimpor key pair di Panduan EC2 Pengguna Amazon.

Izin

Tidak diperlukan izin tambahan untuk menggunakan contoh kode di halaman ini.

Persyaratan

Tambahkan parameter berikut `requirements.txt` untuk menginstal `apache-airflow-providers-ssh` paket di server web. Setelah lingkungan Anda diperbarui dan Amazon MWAA berhasil menginstal dependensi, Anda akan mendapatkan jenis koneksi SSH baru di UI.

```
-c https://raw.githubusercontent.com/apache/airflow/constraints-Airflow-version/
constraints-Python-version.txt
apache-airflow-providers-ssh
```

> **Note**
>
> `-c`mendefinisikan kendala URL di. `requirements.txt` Ini memastikan bahwa Amazon MWAA menginstal versi paket yang benar untuk lingkungan Anda.

Salin kunci rahasia Anda ke Amazon S3

Gunakan AWS Command Line Interface perintah berikut untuk menyalin `.pem` kunci Anda ke `dags` direktori lingkungan Anda di Amazon S3.

```
aws s3 cp your-secret-key.pem s3://amzn-s3-demo-bucket/dags/
```

Amazon MWAA menyalin kontendags, termasuk `.pem` kunci, ke `/usr/local/airflow/dags/` direktori lokal, Dengan melakukan ini, Apache Airflow dapat mengakses kunci.

Buat koneksi Apache Airflow baru

Untuk membuat koneksi SSH baru menggunakan Apache Airflow UI

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Dari daftar lingkungan, pilih Open Airflow UI untuk lingkungan Anda.
3. Pada halaman Apache Airflow UI, pilih Admin dari bilah navigasi utama untuk memperluas daftar dropdown, lalu pilih Connections.
4. Pada halaman Daftar Koneksi, pilih +, atau Tambahkan tombol catatan baru untuk menambahkan koneksi baru.
5. Pada halaman Add Connection, tambahkan informasi berikut:
   a. Untuk Connection Id, masukkan**`ssh_new`**.
   b. Untuk Jenis Koneksi, pilih SSH dari daftar dropdown.

> **Note**
>
> Jika jenis koneksi SSH tidak tersedia dalam daftar, Amazon MWAA belum menginstal paket yang diperlukan. `apache-airflow-providers-ssh` Perbarui `requirements.txt` file Anda untuk menyertakan paket ini, lalu coba lagi.

c. Untuk Host, masukkan alamat IP untuk EC2 instance Amazon yang ingin Anda sambungkan. Misalnya, **`12.345.67.89`**.

d. Untuk Nama Pengguna, masukkan **`ec2-user`** jika Anda terhubung ke EC2 instans Amazon. Nama pengguna Anda mungkin berbeda, tergantung pada jenis instance jarak jauh yang ingin Anda sambungkan Apache Airflow.

e. Untuk Ekstra, masukkan pasangan kunci-nilai berikut dalam format JSON:

```
{ "key_file": "/usr/local/airflow/dags/your-secret-key.pem" }
```

Pasangan kunci-nilai ini menginstruksikan Apache Airflow untuk mencari kunci rahasia di direktori lokal. `/dags`

Contoh kode

DAG berikut menggunakan `SSHOperator` untuk menyambung ke EC2 instance Amazon target Anda, lalu menjalankan perintah `hostname` Linux untuk mencetak nama instance. Anda dapat memodifikasi DAG untuk menjalankan perintah atau skrip apa pun pada instance jarak jauh.

1. Buka terminal, dan arahkan ke direktori tempat kode DAG Anda disimpan. Misalnya:

```
cd dags
```

2. Salin isi contoh kode berikut dan simpan secara lokal sebagai`ssh.py`.

```
from airflow.decorators import dag
from datetime import datetime
from airflow.providers.ssh.operators.ssh import SSHOperator

@dag(
    dag_id="ssh_operator_example",
    schedule_interval=None,
```

```
    start_date=datetime(2022, 1, 1),
    catchup=False,
    )
def ssh_dag():
    task_1=SSHOperator(
        task_id="ssh_task",
        ssh_conn_id='ssh_new',
        command='hostname',
    )

my_ssh_dag = ssh_dag()
```

3. Jalankan AWS CLI perintah berikut untuk menyalin DAG ke bucket lingkungan Anda, lalu picu DAG menggunakan Apache Airflow UI.

```
aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. Jika berhasil, Anda akan mendapatkan output yang mirip dengan yang berikut di log tugas `ssh_task` di `ssh_operator_example` DAG:

```
[2022-01-01, 12:00:00 UTC] {{base.py:79}} INFO - Using connection to: id: ssh_new.
 Host: 12.345.67.89, Port: None,
Schema: , Login: ec2-user, Password: None, extra: {'key_file': '/usr/local/airflow/
dags/your-secret-key.pem'}
[2022-01-01, 12:00:00 UTC] {{ssh.py:264}} WARNING - Remote Identification Change is
 not verified. This won't protect against Man-In-The-Middle attacks [2022-01-01,
 12:00:00 UTC] {{ssh.py:270}} WARNING - No Host Key Verification. This won't
 protect against Man-In-The-Middle attacks
[2022-01-01, 12:00:00 UTC] {{transport.py:1819}} INFO - Connected (version 2.0,
 client OpenSSH_7.4)
[2022-01-01, 12:00:00 UTC] {{transport.py:1819}} INFO - Authentication (publickey)
 successful!
[2022-01-01, 12:00:00 UTC] {{ssh.py:139}} INFO - Running command: hostname
[2022-01-01, 12:00:00 UTC]{{ssh.py:171}} INFO - ip-123-45-67-89.us-
west-2.compute.internal
[2022-01-01, 12:00:00 UTC] {{taskinstance.py:1280}} INFO - Marking task as SUCCESS.
 dag_id=ssh_operator_example, task_id=ssh_task, execution_date=20220712T200914,
 start_date=20220712T200915, end_date=20220712T200916
```

Menggunakan kunci rahasia AWS Secrets Manager untuk koneksi Apache Airflow Snowflake

Contoh panggilan berikut AWS Secrets Manager untuk mendapatkan kunci rahasia untuk koneksi Apache Airflow Snowflake di Amazon Managed Workflows untuk Apache Airflow. Ini mengasumsikan Anda telah menyelesaikan langkah-langkahnya. Mengkonfigurasi koneksi Apache Airflow menggunakan rahasia AWS Secrets Manager

Topik

- Versi
- Prasyarat
- Izin
- Persyaratan
- Contoh kode
- Apa selanjutnya?

Versi

Anda dapat menggunakan contoh kode pada halaman ini dengan Apache Airflow v2 di Python 3.10 dan Apache Airflowv3 di Python 3.11.

Prasyarat

Untuk menggunakan kode sampel di halaman ini, Anda memerlukan yang berikut:

- Backend Secrets Manager sebagai opsi konfigurasi Apache Airflow seperti yang tercantum dalam. Mengkonfigurasi koneksi Apache Airflow menggunakan rahasia AWS Secrets Manager
- String koneksi Apache Airflow di Secrets Manager seperti yang tercantum di. Mengkonfigurasi koneksi Apache Airflow menggunakan rahasia AWS Secrets Manager

Izin

- Izin Secrets Manager seperti yang tercantum dalamMengkonfigurasi koneksi Apache Airflow menggunakan rahasia AWS Secrets Manager.

Persyaratan

Untuk menggunakan kode sampel pada halaman ini, tambahkan dependensi berikut ke Anda. `requirements.txt` Untuk mempelajari lebih lanjut, lihat[Menginstal dependensi Python](#).

```
apache-airflow-providers-snowflake==1.3.0
```

Contoh kode

Langkah-langkah berikut menjelaskan cara membuat kode DAG yang memanggil Secrets Manager untuk mendapatkan rahasianya.

1. Di prompt perintah Anda, arahkan ke direktori tempat kode DAG Anda disimpan. Misalnya:

   ```
   cd dags
   ```

2. Salin isi contoh kode berikut dan simpan secara lokal sebagai`snowflake_connection.py`.

   ```
   """
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

   Permission is hereby granted, free of charge, to any person obtaining a copy of
   this software and associated documentation files (the "Software"), to deal in
   the Software without restriction, including without limitation the rights to
   use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
   the Software, and to permit persons to whom the Software is furnished to do so.

   THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
   IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
   FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
   COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
   IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
   CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
   """
   from airflow import DAG
   from airflow.providers.snowflake.operators.snowflake import SnowflakeOperator
   from airflow.utils.dates import days_ago

   snowflake_query = [
       """use warehouse "MY_WAREHOUSE";""",
       """select * from "SNOWFLAKE_SAMPLE_DATA"."WEATHER"."WEATHER_14_TOTAL" limit
    100;""",
   ```

```
]

with DAG(dag_id='snowflake_test', schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
    snowflake_select = SnowflakeOperator(
        task_id="snowflake_select",
        sql=snowflake_query,
        snowflake_conn_id="snowflake_conn",
    )
```

Apa selanjutnya?

- Pelajari cara mengunggah kode DAG dalam contoh ini ke `dags` folder di bucket Amazon S3 Anda. Menambahkan atau memperbarui DAGs

Menggunakan DAG untuk menulis metrik khusus CloudWatch

Anda dapat menggunakan contoh kode berikut untuk menulis grafik asiklik terarah (DAG) yang menjalankan a `PythonOperator` untuk mengambil metrik tingkat OS untuk lingkungan Amazon MWAA. DAG kemudian menerbitkan data sebagai metrik khusus ke Amazon. CloudWatch

Metrik tingkat OS khusus memberi Anda visibilitas tambahan tentang bagaimana pekerja lingkungan Anda memanfaatkan sumber daya seperti memori virtual dan CPU. Anda dapat menggunakan informasi ini untuk memilih kelas lingkungan yang paling sesuai dengan beban kerja Anda.

Topik

- Versi
- Prasyarat
- Izin
- Dependensi
- Contoh kode

Versi

Anda dapat menggunakan contoh kode pada halaman ini dengan Apache Airflow v2 di Python 3.10 dan Apache Airflowv3 di Python 3.11.

Prasyarat

Untuk menggunakan contoh kode pada halaman ini, Anda memerlukan yang berikut:

- Lingkungan Amazon MWAA.

Izin

Tidak ada izin tambahan yang diperlukan untuk menggunakan contoh kode di halaman ini.

Dependensi

- Tidak ada dependensi tambahan yang diperlukan untuk menggunakan contoh kode di halaman ini.

Contoh kode

1. Di prompt perintah Anda, arahkan ke folder tempat kode DAG Anda disimpan. Misalnya:

```
cd dags
```

2. Salin isi contoh kode berikut dan simpan secara lokal sebagai`dag-custom-metrics.py`. Ganti `MWAA-ENV-NAME` dengan nama lingkungan Anda.

```
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
from airflow.utils.dates import days_ago
from datetime import datetime
import os,json,boto3,psutil,socket

def publish_metric(client,name,value,cat,unit='None'):
    environment_name = os.getenv("MWAA_ENV_NAME")
    value_number=float(value)
    hostname = socket.gethostname()
    ip_address = socket.gethostbyname(hostname)
    print('writing value',value_number,'to metric',name)
    response = client.put_metric_data(
        Namespace='MWAA-Custom',
        MetricData=[
            {
                'MetricName': name,
```

```
                'Dimensions': [
                    {
                        'Name': 'Environment',
                        'Value': environment_name
                    },
                    {
                        'Name': 'Category',
                        'Value': cat
                    },
                    {
                        'Name': 'Host',
                        'Value': ip_address
                    },
                ],
                'Timestamp': datetime.now(),
                'Value': value_number,
                'Unit': unit
            },
        ]
    )
    print(response)
    return response

def python_fn(**kwargs):
    client = boto3.client('cloudwatch')

    cpu_stats = psutil.cpu_stats()
    print('cpu_stats', cpu_stats)

    virtual = psutil.virtual_memory()
    cpu_times_percent = psutil.cpu_times_percent(interval=0)

    publish_metric(client=client, name='virtual_memory_total',
 cat='virtual_memory', value=virtual.total, unit='Bytes')
    publish_metric(client=client, name='virtual_memory_available',
 cat='virtual_memory', value=virtual.available, unit='Bytes')
    publish_metric(client=client, name='virtual_memory_used', cat='virtual_memory',
 value=virtual.used, unit='Bytes')
    publish_metric(client=client, name='virtual_memory_free', cat='virtual_memory',
 value=virtual.free, unit='Bytes')
    publish_metric(client=client, name='virtual_memory_active',
 cat='virtual_memory', value=virtual.active, unit='Bytes')
    publish_metric(client=client, name='virtual_memory_inactive',
 cat='virtual_memory', value=virtual.inactive, unit='Bytes')
```

```
    publish_metric(client=client, name='virtual_memory_percent',
 cat='virtual_memory', value=virtual.percent, unit='Percent')

    publish_metric(client=client, name='cpu_times_percent_user',
 cat='cpu_times_percent', value=cpu_times_percent.user, unit='Percent')
    publish_metric(client=client, name='cpu_times_percent_system',
 cat='cpu_times_percent', value=cpu_times_percent.system, unit='Percent')
    publish_metric(client=client, name='cpu_times_percent_idle',
 cat='cpu_times_percent', value=cpu_times_percent.idle, unit='Percent')

    return "OK"


with DAG(dag_id=os.path.basename(__file__).replace(".py", ""),
 schedule_interval='*/5 * * * *', catchup=False, start_date=days_ago(1)) as dag:
    t = PythonOperator(task_id="memory_test", python_callable=python_fn,
 provide_context=True)
```

3. Jalankan AWS CLI perintah berikut untuk menyalin DAG ke bucket lingkungan Anda, lalu picu DAG menggunakan Apache Airflow UI.

```
aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. Jika DAG berhasil berjalan, Anda mendapatkan sesuatu yang mirip dengan yang berikut di log Apache Airflow Anda:

```
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO -
 cpu_stats scpustats(ctx_switches=3253992384, interrupts=1964237163,
 soft_interrupts=492328209, syscalls=0)
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO - writing value
 16024199168.0 to metric virtual_memory_total
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO - {'ResponseMetadata':
 {'RequestId': 'fad289ac-aa51-46a9-8b18-24e4e4063f4d', 'HTTPStatusCode': 200,
 'HTTPHeaders': {'x-amzn-requestid': 'fad289ac-aa51-46a9-8b18-24e4e4063f4d',
 'content-type': 'text/xml', 'content-length': '212', 'date': 'Tue, 16 Aug 2022
 17:54:45 GMT'}, 'RetryAttempts': 0}}
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO - writing value
 14356287488.0 to metric virtual_memory_available
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO - {'ResponseMetadata':
 {'RequestId': '6ef60085-07ab-4865-8abf-dc94f90cab46', 'HTTPStatusCode': 200,
 'HTTPHeaders': {'x-amzn-requestid': '6ef60085-07ab-4865-8abf-dc94f90cab46',
 'content-type': 'text/xml', 'content-length': '212', 'date': 'Tue, 16 Aug 2022
 17:54:45 GMT'}, 'RetryAttempts': 0}}
```

```
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO - writing value
 1342296064.0 to metric virtual_memory_used
[2022-08-16, 10:54:46 UTC] {{logging_mixin.py:109}} INFO - {'ResponseMetadata':
 {'RequestId': 'd5331438-5d3c-4df2-bc42-52dcf8d60a00', 'HTTPStatusCode': 200,
 'HTTPHeaders': {'x-amzn-requestid': 'd5331438-5d3c-4df2-bc42-52dcf8d60a00',
 'content-type': 'text/xml', 'content-length': '212', 'date': 'Tue, 16 Aug 2022
 17:54:45 GMT'}, 'RetryAttempts': 0}}
...
[2022-08-16, 10:54:46 UTC] {{python.py:152}} INFO - Done. Returned value was: OK
[2022-08-16, 10:54:46 UTC] {{taskinstance.py:1280}} INFO - Marking task as SUCCESS.
 dag_id=dag-custom-metrics, task_id=memory_test, execution_date=20220816T175444,
 start_date=20220816T175445, end_date=20220816T175446
[2022-08-16, 10:54:46 UTC] {{local_task_job.py:154}} INFO - Task exited with return
 code 0
```

Pembersihan basis data Aurora PostgreSQL di lingkungan Amazon MWAA

Alur Kerja Terkelola Amazon untuk Apache Airflow menggunakan database Aurora PostgreSQL sebagai database metadata Apache Airflow, tempat DAG berjalan dan instance tugas disimpan. Kode contoh berikut secara berkala menghapus entri dari database Aurora PostgreSQL khusus untuk lingkungan Amazon MWAA Anda.

Topik

- Versi
- Prasyarat
- Dependensi
- Contoh kode

Versi

Contoh kode di halaman ini khusus untuk Apache Airflow v2 dan v3 yang didukung di Amazon MWAA. Lihat versi Apache Airflow yang didukung.

Prasyarat

Untuk menggunakan kode sampel di halaman ini, Anda memerlukan yang berikut:

- Lingkungan Amazon MWAA.

Dependensi

Untuk menggunakan contoh kode ini dengan Apache Airflow v2, tidak diperlukan dependensi tambahan. Gunakan aws-mwaa-docker-images untuk menginstal Apache Airflow.

Contoh kode

DAG berikut membersihkan database metadata untuk tabel yang ditentukan dalam. `TABLES_TO_CLEAN` Contoh menghapus data dari tabel yang ditentukan yang lebih tua dari 30 hari. Untuk menyesuaikan seberapa jauh entri dihapus, atur `MAX_AGE_IN_DAYS` ke nilai yang berbeda.

Apache Airflow v3.0.6 to 3.2.1

```
from datetime import datetime
from airflow import DAG
from airflow.providers.standard.operators.bash import BashOperator

# Note: Database commands might time out if running longer than 5 minutes. If this
 occurs, please increase the MAX_AGE_IN_DAYS (or change
# timestamp parameter to an earlier date) for initial runs, then reduce on
 subsequent runs until the desired retention is met.

MAX_AGE_IN_DAYS = 30

# To clean specific tables, please provide a comma-separated list per
# https://airflow.apache.org/docs/apache-airflow/stable/cli-and-env-variables-
ref.html#clean

# A value of None will clean all tables
TABLES_TO_CLEAN = None

with DAG(
    dag_id="clean_db_dag",
    schedule=None,
    catchup=False,
    start_date=datetime(2026, 1, 1),
) as dag:
    tables_flag = f"--tables '{TABLES_TO_CLEAN}' " if TABLES_TO_CLEAN else ""

    bash_command = (
```

```
        f"TIMESTAMP=$(date -u -d '{MAX_AGE_IN_DAYS} days ago' '+%Y-%m-%d %H:%M:%S'
 2>/dev/null "
        f"|| date -u -v-{MAX_AGE_IN_DAYS}d '+%Y-%m-%d %H:%M:%S') && "
        "echo \"Cleaning records before: $TIMESTAMP\" && "
        "airflow db clean "
        "--clean-before-timestamp \"$TIMESTAMP\" "
        f"{tables_flag}"
        "--skip-archive --yes"
    )

    cli_command = BashOperator(
        task_id="bash_command",
        bash_command=bash_command,
    )
```

Apache Airflow v2.7.2 to 2.11.0

```
from airflow import DAG
from airflow.models.param import Param
from airflow.operators.bash_operator import BashOperator
from airflow.utils.dates import days_ago

from datetime import datetime, timedelta

# Note: Database commands might time out if running longer than 5 minutes. If this
 occurs, please increase the MAX_AGE_IN_DAYS (or change
# timestamp parameter to an earlier date) for initial runs, then reduce on
 subsequent runs until the desired retention is met.

MAX_AGE_IN_DAYS = 30

# To clean specific tables, please provide a comma-separated list per
# https://airflow.apache.org/docs/apache-airflow/stable/cli-and-env-variables-
ref.html#clean
# A value of None will clean all tables

TABLES_TO_CLEAN = None

with DAG(
    dag_id="clean_db_dag",
    schedule_interval=None,
    catchup=False,
    start_date=days_ago(1),
```

```
    params={
        "timestamp": Param(
            default=(datetime.now()-timedelta(days=MAX_AGE_IN_DAYS)).strftime("%Y-
%m-%d %H:%M:%S"),
            type="string",
            minLength=1,
            maxLength=255,
        ),
    }
) as dag:
    if TABLES_TO_CLEAN:
        bash_command="airflow db clean --clean-before-timestamp
 '{{ params.timestamp }}' --tables '"+TABLES_TO_CLEAN+"' --skip-archive --yes"
    else:
        bash_command="airflow db clean --clean-before-timestamp
 '{{ params.timestamp }}' --skip-archive --yes"

    cli_command = BashOperator(
        task_id="bash_command",
        bash_command=bash_command
    )
```

Mengekspor metadata lingkungan ke file CSV di Amazon S3

Gunakan contoh kode berikut untuk membuat grafik asiklik terarah (DAG) yang menanyakan database untuk berbagai informasi yang dijalankan DAG, dan menulis data ke `.csv` file yang disimpan di Amazon S3.

Anda mungkin ingin mengekspor informasi dari database Aurora PostgreSQL lingkungan Anda untuk memeriksa data secara lokal, mengarsipkannya dalam penyimpanan objek, atau menggabungkannya dengan alat seperti Amazon S3 ke Amazon Redshift operator dan pembersihan database, untuk memindahkan metadata Amazon MWAA keluar dari lingkungan, tetapi melestarikannya untuk analisis masa depan.

Anda dapat menanyakan database untuk salah satu objek yang tercantum dalam model Apache Airflow. Contoh kode ini menggunakan tiga model,`DagRun`,`TaskFail`, dan`TaskInstance`, yang memberikan informasi yang relevan Untuk DAG berjalan.

Topik

- Versi

- Prasyarat
- Izin
- Persyaratan
- Contoh kode

Versi

Anda dapat menggunakan contoh kode pada halaman ini dengan Apache Airflow v2 di Python 3.10 dan Apache Airflowv3 di Python 3.11.

Prasyarat

Untuk menggunakan kode sampel di halaman ini, Anda memerlukan yang berikut:

- Lingkungan Amazon MWAA.
- Bucket Amazon S3 baru tempat Anda ingin mengekspor informasi metadata Anda.

Izin

Amazon MWAA memerlukan izin untuk `s3:PutObject` tindakan Amazon S3 untuk menulis informasi metadata yang ditanyakan ke Amazon S3. Tambahkan pernyataan kebijakan berikut ke peran eksekusi lingkungan Anda.

```
{
  "Effect": "Allow",
  "Action": "s3:PutObject*",
  "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
}
```

Kebijakan ini membatasi akses tulis saja*amzn-s3-demo-bucket*.

Persyaratan

Untuk menggunakan contoh kode ini dengan Apache Airflow v2 dan yang lebih baru, tidak diperlukan dependensi tambahan. Gunakan aws-mwaa-docker-imagesuntuk menginstal Apache Airflow.

Contoh kode

Langkah-langkah berikut menjelaskan bagaimana Anda dapat membuat DAG yang menanyakan Aurora PostgreSQL dan menulis hasilnya ke bucket Amazon S3 baru Anda.

1. Di terminal Anda, arahkan ke direktori tempat kode DAG Anda disimpan. Contoh:

```
cd dags
```

2. Salin isi contoh kode berikut dan simpan secara lokal sebagai`metadata_to_csv.py`. Anda dapat mengubah nilai yang ditetapkan `MAX_AGE_IN_DAYS` untuk mengontrol usia rekaman terlama kueri DAG Anda dari database metadata.

```
from airflow.decorators import dag, task
from airflow import settings
import os
import boto3
from airflow.utils.dates import days_ago
from airflow.models import DagRun, TaskFail, TaskInstance
import csv, re
from io import StringIO

DAG_ID = os.path.basename(__file__).replace(".py", "")

MAX_AGE_IN_DAYS = 30
S3_BUCKET = '<your-export-bucket>'
S3_KEY = 'files/export/{0}.csv'

# You can add other objects to export from the metadatabase,
OBJECTS_TO_EXPORT = [
    [DagRun,DagRun.execution_date],
    [TaskFail,TaskFail.end_date],
    [TaskInstance, TaskInstance.execution_date],
]

@task()
def export_db_task(**kwargs):
    session = settings.Session()
    print("session: ",str(session))

    oldest_date = days_ago(MAX_AGE_IN_DAYS)
    print("oldest_date: ",oldest_date)
```

```
    s3 = boto3.client('s3')

    for x in OBJECTS_TO_EXPORT:
        query = session.query(x[0]).filter(x[1] >= days_ago(MAX_AGE_IN_DAYS))
        print("type",type(query))
        allrows=query.all()
        name=re.sub("[<>']", "", str(x[0]))
        print(name,": ",str(allrows))

        if len(allrows) > 0:
            outfileStr=""
            f = StringIO(outfileStr)
            w = csv.DictWriter(f, vars(allrows[0]).keys())
            w.writeheader()
            for y in allrows:
                w.writerow(vars(y))
            outkey = S3_KEY.format(name[6:])
            s3.put_object(Bucket=S3_BUCKET, Key=outkey, Body=f.getvalue())

@dag(
    dag_id=DAG_ID,
    schedule_interval=None,
    start_date=days_ago(1),
    )
def export_db():
    t = export_db_task()

metadb_to_s3_test = export_db()
```

3. Jalankan AWS CLI perintah berikut untuk menyalin DAG ke bucket lingkungan Anda, lalu picu DAG menggunakan Apache Airflow UI.

```
aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. Jika berhasil, Anda akan menampilkan yang serupa dengan yang berikut di log tugas untuk `export_db` tugas:

```
[2022-01-01, 12:00:00 PDT] {{logging_mixin.py:109}} INFO - type <class
 'sqlalchemy.orm.query.Query'>
[2022-01-01, 12:00:00 PDT] {{logging_mixin.py:109}} INFO - class
 airflow.models.dagrun.DagRun : [your-tasks]
```

```
[2022-01-01, 12:00:00 PDT] {{logging_mixin.py:109}} INFO - type <class
 'sqlalchemy.orm.query.Query'>
[2022-01-01, 12:00:00 PDT] {{logging_mixin.py:109}} INFO - class
 airflow.models.taskfail.TaskFail :  [your-tasks]
[2022-01-01, 12:00:00 PDT] {{logging_mixin.py:109}} INFO - type <class
 'sqlalchemy.orm.query.Query'>
[2022-01-01, 12:00:00 PDT] {{logging_mixin.py:109}} INFO - class
 airflow.models.taskinstance.TaskInstance :  [your-tasks]
[2022-01-01, 12:00:00 PDT] {{python.py:152}} INFO - Done. Returned value was: OK
[2022-01-01, 12:00:00 PDT] {{taskinstance.py:1280}} INFO - Marking task as
 SUCCESS. dag_id=metadb_to_s3, task_id=export_db, execution_date=20220101T000000,
 start_date=20220101T000000, end_date=20220101T000000
[2022-01-01, 12:00:00 PDT] {{local_task_job.py:154}} INFO - Task exited with return
 code 0
[2022-01-01, 12:00:00 PDT] {{local_task_job.py:264}} INFO - 0 downstream tasks
 scheduled from follow-on schedule check
```

Anda sekarang dapat mengakses dan mengunduh `.csv` file yang diekspor di bucket Amazon S3 baru Anda. `/files/export/`

Menggunakan kunci rahasia AWS Secrets Manager untuk variabel Apache Airflow

Contoh panggilan berikut AWS Secrets Manager untuk mendapatkan kunci rahasia untuk variabel Apache Airflow di Amazon Managed Workflows untuk Apache Airflow. Ini mengasumsikan Anda telah menyelesaikan langkah-langkahnya. Mengkonfigurasi koneksi Apache Airflow menggunakan rahasia AWS Secrets Manager

Topik

- Versi
- Prasyarat
- Izin
- Persyaratan
- Contoh kode
- Apa selanjutnya?

Versi

Anda dapat menggunakan contoh kode pada halaman ini dengan Apache Airflow v2 di Python 3.10 dan Apache Airflowv3 di Python 3.11.

Prasyarat

Untuk menggunakan kode sampel di halaman ini, Anda memerlukan yang berikut:

- Backend Secrets Manager sebagai opsi konfigurasi Apache Airflow seperti yang tercantum dalam. Mengkonfigurasi koneksi Apache Airflow menggunakan rahasia AWS Secrets Manager
- String variabel Apache Airflow di Secrets Manager seperti yang tercantum dalam. Mengkonfigurasi koneksi Apache Airflow menggunakan rahasia AWS Secrets Manager

Izin

- Izin Secrets Manager seperti yang tercantum dalamMengkonfigurasi koneksi Apache Airflow menggunakan rahasia AWS Secrets Manager.

Persyaratan

Untuk menggunakan contoh kode ini dengan Apache Airflow v2 dan yang lebih baru, tidak diperlukan dependensi tambahan. Gunakan aws-mwaa-docker-imagesuntuk menginstal Apache Airflow.

Contoh kode

Langkah-langkah berikut menjelaskan cara membuat kode DAG yang memanggil Secrets Manager untuk mendapatkan rahasianya.

1. Di prompt perintah Anda, arahkan ke direktori tempat kode DAG Anda disimpan. Misalnya:

```
cd dags
```

2. Salin isi contoh kode berikut dan simpan secara lokal sebagai`secrets-manager-var.py`.

```
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
from airflow.models import Variable
```

```
from airflow.utils.dates import days_ago
from datetime import timedelta
import os
DAG_ID = os.path.basename(__file__).replace(".py", "")
DEFAULT_ARGS = {
    'owner': 'airflow',
    'depends_on_past': False,
    'email': ['airflow@example.com'],
    'email_on_failure': False,
    'email_on_retry': False,
}
def get_variable_fn(**kwargs):
    my_variable_name = Variable.get("test-variable", default_var="undefined")
    print("my_variable_name: ", my_variable_name)
    return my_variable_name
with DAG(
    dag_id=DAG_ID,
    default_args=DEFAULT_ARGS,
    dagrun_timeout=timedelta(hours=2),
    start_date=days_ago(1),
    schedule_interval='@once',
    tags=['variable']
) as dag:
    get_variable = PythonOperator(
        task_id="get_variable",
        python_callable=get_variable_fn,
        provide_context=True
    )
```

Apa selanjutnya?

- Pelajari cara mengunggah kode DAG dalam contoh ini ke `dags` folder di bucket Amazon S3 Anda. Menambahkan atau memperbarui DAGs

Menggunakan kunci rahasia AWS Secrets Manager untuk koneksi Apache Airflow

Contoh panggilan berikut AWS Secrets Manager untuk mendapatkan kunci rahasia untuk koneksi Apache Airflow di Amazon Managed Workflows untuk Apache Airflow. Ini mengasumsikan Anda telah

menyelesaikan langkah-langkahnya. Mengkonfigurasi koneksi Apache Airflow menggunakan rahasia AWS Secrets Manager

Topik

- Versi
- Prasyarat
- Izin
- Persyaratan
- Contoh kode
- Apa selanjutnya?

Versi

Anda dapat menggunakan contoh kode pada halaman ini dengan Apache Airflow v2 di Python 3.10 dan Apache Airflowv3 di Python 3.11.

Prasyarat

Untuk menggunakan kode sampel di halaman ini, Anda memerlukan yang berikut:

- Backend Secrets Manager sebagai opsi konfigurasi Apache Airflow seperti yang tercantum dalam. Mengkonfigurasi koneksi Apache Airflow menggunakan rahasia AWS Secrets Manager
- String koneksi Apache Airflow di Secrets Manager seperti yang tercantum di. Mengkonfigurasi koneksi Apache Airflow menggunakan rahasia AWS Secrets Manager

Izin

- Izin Secrets Manager seperti yang tercantum dalamMengkonfigurasi koneksi Apache Airflow menggunakan rahasia AWS Secrets Manager.

Persyaratan

Untuk menggunakan contoh kode ini dengan Apache Airflow v2 dan yang lebih baru, tidak diperlukan dependensi tambahan. Gunakan aws-mwaa-docker-imagesuntuk menginstal Apache Airflow.

Contoh kode

Langkah-langkah berikut menjelaskan cara membuat kode DAG yang memanggil Secrets Manager untuk mendapatkan rahasianya.

1. Di prompt perintah Anda, arahkan ke direktori tempat kode DAG Anda disimpan. Misalnya:

```
cd dags
```

2. Salin isi contoh kode berikut dan simpan secara lokal sebagai`secrets-manager.py`.

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
from airflow import DAG, settings, secrets
from airflow.operators.python import PythonOperator
from airflow.utils.dates import days_ago
from airflow.providers.amazon.aws.hooks.base_aws import AwsBaseHook

from datetime import timedelta
import os

### The steps to create this secret key can be found at: https://
docs.aws.amazon.com/mwaa/latest/userguide/connections-secrets-manager.html
sm_secretId_name = 'airflow/connections/myconn'

default_args = {
    'owner': 'airflow',
    'start_date': days_ago(1),
    'depends_on_past': False
```

```
}


### Gets the secret myconn from Secrets Manager
def read_from_aws_sm_fn(**kwargs):
    ### set up Secrets Manager
    hook = AwsBaseHook(client_type='secretsmanager')
    client = hook.get_client_type(region_name='us-east-1')
    response = client.get_secret_value(SecretId=sm_secretId_name)
    myConnSecretString = response["SecretString"]

    return myConnSecretString

### 'os.path.basename(__file__).replace(".py", "")' uses the file name secrets-
manager.py for a DAG ID of secrets-manager
with DAG(
        dag_id=os.path.basename(__file__).replace(".py", ""),
        default_args=default_args,
        dagrun_timeout=timedelta(hours=2),
        start_date=days_ago(1),
        schedule_interval=None
) as dag:
    write_all_to_aws_sm = PythonOperator(
        task_id="read_from_aws_sm",
        python_callable=read_from_aws_sm_fn,
        provide_context=True
    )
```

Apa selanjutnya?

- Pelajari cara mengunggah kode DAG dalam contoh ini ke `dags` folder di bucket Amazon S3 Anda. Menambahkan atau memperbarui DAGs

Membuat plugin khusus dengan Oracle

Contoh berikut memandu Anda melalui langkah-langkah untuk membuat plugin khusus menggunakan Oracle untuk Amazon MWAA dan dapat dikombinasikan dengan plugin dan binari khusus lainnya di file plugins.zip Anda.

Daftar Isi

- Versi
- Prasyarat
- Izin
- Persyaratan
- Contoh kode
- Buat plugin kustom
  - Unduh dependensi
  - PluginKhusus
  - Plugins.zip
- Opsi konfigurasi Airflow
- Apa selanjutnya?

Versi

Anda dapat menggunakan contoh kode pada halaman ini dengan Apache Airflow v2 di Python 3.10 dan Apache Airflowv3 di Python 3.11.

Prasyarat

Untuk menggunakan kode sampel di halaman ini, Anda memerlukan yang berikut:

- Lingkungan Amazon MWAA.
- Pencatatan pekerja diaktifkan pada tingkat log apa pun, `CRITICAL` atau di bagian sebelumnya untuk lingkungan Anda. Untuk informasi selengkapnya tentang jenis log Amazon MWAA dan cara mengelola grup log Anda, lihat the section called “Mengakses log Aliran Udara”

Izin

Tidak diperlukan izin tambahan untuk menggunakan contoh kode di halaman ini.

Persyaratan

Untuk menggunakan kode sampel pada halaman ini, tambahkan dependensi berikut ke Anda. `requirements.txt` Untuk mempelajari lebih lanjut, lihatMenginstal dependensi Python.

```
-c https://raw.githubusercontent.com/apache/airflow/constraints-2.0.2/
constraints-3.7.txt
cx_Oracle
apache-airflow-providers-oracle
```

Contoh kode

Langkah-langkah berikut menjelaskan cara membuat kode DAG yang akan menguji plugin kustom.

1. Di prompt perintah Anda, arahkan ke direktori tempat kode DAG Anda disimpan. Misalnya:

   ```
   cd dags
   ```

2. Salin isi contoh kode berikut dan simpan secara lokal sebagai`oracle.py`.

   ```
   from airflow import DAG
   from airflow.operators.python_operator import PythonOperator
   from airflow.utils.dates import days_ago
   import os
   import cx_Oracle

   DAG_ID = os.path.basename(__file__).replace(".py", "")

   def testHook(**kwargs):
       cx_Oracle.init_oracle_client()
       version = cx_Oracle.clientversion()
       print("cx_Oracle.clientversion",version)
       return version

   with DAG(dag_id=DAG_ID, schedule_interval=None, catchup=False,
    start_date=days_ago(1)) as dag:
       hook_test = PythonOperator(
           task_id="hook_test",
           python_callable=testHook,
           provide_context=True
       )
   ```

Buat plugin kustom

Bagian ini menjelaskan cara mengunduh dependensi, membuat plugin kustom dan plugins.zip.

Unduh dependensi

Amazon MWAA akan mengekstrak konten plugins.zip ke dalam setiap penjadwal `/usr/local/airflow/plugins` Amazon MWAA dan wadah pekerja. Ini digunakan untuk menambahkan binari ke lingkungan Anda. Langkah-langkah berikut menjelaskan cara merakit file yang diperlukan untuk plugin kustom.

Tarik gambar wadah Amazon Linux

1. Di command prompt Anda, tarik image container Amazon Linux, dan jalankan container secara lokal. Misalnya:

   ```
   docker pull amazonlinux
         docker run -it amazonlinux:latest /bin/bash
   ```

   Prompt perintah Anda dapat memanggil baris perintah bash. Misalnya:

   ```
   bash-4.2#
   ```

2. Instal fasilitas asinkron I/O asli Linux (libaio).

   ```
   yum -y install libaio
   ```

3. Biarkan jendela ini terbuka untuk langkah selanjutnya. Kami akan menyalin file-file berikut secara lokal:`lib64/libaio.so.1`,,`lib64/libaio.so.1.0.0`. `lib64/libaio.so.1.0.1`

Unduh folder klien

1. Instal paket unzip secara lokal. Misalnya:

   ```
   sudo yum install unzip
   ```

2. Buat `oracle_plugin` direktori. Misalnya:

   ```
   mkdir oracle_plugin
   cd oracle_plugin
   ```

3. Gunakan perintah curl berikut untuk mendownload instantclient-basic-linux.x64-18.5.0.0.0dbru.zip dari Oracle Instant Client Downloads untuk Linux x86-64 (64-bit).

```
curl https://download.oracle.com/otn_software/linux/instantclient/185000/
instantclient-basic-linux.x64-18.5.0.0.0dbru.zip > client.zip
```

4. Buka zip `client.zip` file. Misalnya:

```
unzip *.zip
```

Ekstrak file dari Docker

1. Dalam prompt perintah baru, tampilkan dan tuliskan ID kontainer Docker Anda. Misalnya:

```
docker container ls
```

   Prompt perintah Anda dapat mengembalikan semua kontainer dan mereka IDs. Misalnya:

```
debc16fd6970
```

2. Di `oracle_plugin` direktori Anda, ekstrak `lib64/libaio.so.1.0.1` file `lib64/libaio.so.1lib64/libaio.so.1.0.0`,, ke `instantclient_18_5` folder lokal. Misalnya:

```
docker cp debc16fd6970:/lib64/libaio.so.1 instantclient_18_5/
docker cp debc16fd6970:/lib64/libaio.so.1.0.0 instantclient_18_5/
docker cp debc16fd6970:/lib64/libaio.so.1.0.1 instantclient_18_5/
```

PluginKhusus

Apache Airflow akan mengeksekusi isi file Python di folder plugin saat startup. Ini digunakan untuk mengatur dan memodifikasi variabel lingkungan. Langkah-langkah berikut menjelaskan kode sampel untuk plugin kustom.

- Salin isi contoh kode berikut dan simpan secara lokal sebagai`env_var_plugin_oracle.py`.

```
from airflow.plugins_manager import AirflowPlugin
import os

os.environ["LD_LIBRARY_PATH"]='/usr/local/airflow/plugins/instantclient_18_5'
os.environ["DPI_DEBUG_LEVEL"]="64"
```

```
class EnvVarPlugin(AirflowPlugin):
    name = 'env_var_plugin'
```

Plugins.zip

Langkah-langkah berikut menjelaskan cara membuat`plugins.zip`. Isi contoh ini dapat dikombinasikan dengan plugin dan binari Anda yang lain menjadi satu `plugins.zip` file.

Zip isi direktori plugin

1. Di prompt perintah Anda, arahkan ke `oracle_plugin` direktori. Misalnya:

   ```
   cd oracle_plugin
   ```

2. Zip `instantclient_18_5` direktori di plugins.zip. Misalnya:

   ```
   zip -r ../plugins.zip ./
   ```

   Prompt perintah Anda menampilkan:

   ```
   oracle_plugin$ ls
   client.zip  instantclient_18_5
   ```

3. Hapus `client.zip` file. Misalnya:

   ```
   rm client.zip
   ```

Zip file env_var_plugin_oracle.py

1. Tambahkan `env_var_plugin_oracle.py` file ke root plugins.zip. Misalnya:

   ```
   zip plugins.zip env_var_plugin_oracle.py
   ```

2. plugins.zip Anda sekarang menyertakan yang berikut ini:

   ```
   env_var_plugin_oracle.py
   instantclient_18_5/
   ```

Opsi konfigurasi Airflow

Jika Anda menggunakan Apache Airflow v2, `core.lazy_load_plugins : False` tambahkan sebagai opsi konfigurasi Apache Airflow. Untuk mempelajari lebih lanjut, lihat Menggunakan opsi konfigurasi untuk memuat plugin di 2.

Apa selanjutnya?

- Pelajari cara mengunggah `requirements.txt` file dalam contoh ini ke bucket Amazon S3 Anda. Menginstal dependensi Python
- Pelajari cara mengunggah kode DAG dalam contoh ini ke `dags` folder di bucket Amazon S3 Anda. Menambahkan atau memperbarui DAGs
- Pelajari lebih lanjut cara mengunggah `plugins.zip` file dalam contoh ini ke bucket Amazon S3 Anda. Menginstal plugin kustom

Mengubah zona waktu DAG di Amazon MWAA

Apache Airflow menjadwalkan grafik asiklik terarah (DAG) Anda di UTC+0 secara default. Langkah-langkah berikut menyajikan bagaimana Anda dapat mengubah zona waktu di mana Amazon MWAA menjalankan Anda DAGs dengan Pendulum. Secara opsional, topik ini menunjukkan bagaimana Anda dapat membuat plugin khusus untuk mengubah zona waktu log Apache Airflow lingkungan Anda.

Topik

- Versi
- Prasyarat
- Izin
- Buat plugin untuk mengubah zona waktu di log Airflow
- Buat plugins.zip
- Contoh kode
- Apa selanjutnya?

Versi

Anda dapat menggunakan contoh kode pada halaman ini dengan Apache Airflow v2 di Python 3.10 dan Apache Airflowv3 di Python 3.11.

Prasyarat

Untuk menggunakan kode sampel di halaman ini, Anda memerlukan yang berikut:

- Lingkungan Amazon MWAA.

Izin

Tidak diperlukan izin tambahan untuk menggunakan contoh kode di halaman ini.

Buat plugin untuk mengubah zona waktu di log Airflow

Apache Airflow menjalankan file Python di direktori saat start-up. `plugins` Dengan plugin berikut, Anda dapat mengganti zona waktu pelaksana, yang memodifikasi zona waktu di mana Apache Airflow menulis log.

1. Buat direktori bernama `plugins` untuk plugin kustom Anda, dan arahkan ke direktori. Contoh:

```
$ mkdir plugins
$ cd plugins
```

2. Salin isi contoh kode berikut dan simpan secara lokal seperti `dag-timezone-plugin.py` di `plugins` folder.

```
import time
import os

os.environ['TZ'] = 'America/Los_Angeles'
time.tzset()
```

3. Di `plugins` direktori, buat file Python kosong bernama. `__init__.py` `plugins`Direktori Anda harus mirip dengan yang berikut ini:

```
plugins/
 |-- __init__.py
```

```
|-- dag-timezone-plugin.py
```

Buat `plugins.zip`

Langkah-langkah berikut menjelaskan cara membuat`plugins.zip`. Isi contoh ini dapat dikombinasikan dengan plugin dan binari lain menjadi satu `plugins.zip` file.

1. Di prompt perintah Anda, arahkan ke `plugins` direktori dari langkah sebelumnya. Contoh:

```
cd plugins
```

2. Zip konten dalam `plugins` direktori Anda.

```
zip -r ../plugins.zip ./
```

3. Unggah `plugins.zip` ke bucket S3 Anda

```
aws s3 cp plugins.zip s3://your-mwaa-bucket/
```

Contoh kode

Untuk mengubah zona waktu default (UTC+0) di mana DAG berjalan, kita akan menggunakan perpustakaan bernama Pendulum, pustaka Python untuk bekerja dengan datetime yang sadar zona waktu.

1. Di prompt perintah Anda, arahkan ke direktori tempat Anda DAGs disimpan. Contoh:

```
cd dags
```

2. Salin konten dari contoh berikut dan simpan sebagai`tz-aware-dag.py`.

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from datetime import datetime, timedelta
# Import the Pendulum library.
import pendulum

# Instantiate Pendulum and set your timezone.
local_tz = pendulum.timezone("America/Los_Angeles")
```

```
with DAG(
    dag_id = "tz_test",
    schedule_interval="0 12 * * *",
    catchup=False,
    start_date=datetime(2022, 1, 1, tzinfo=local_tz)
) as dag:
    bash_operator_task = BashOperator(
        task_id="tz_aware_task",
        dag=dag,
        bash_command="date"
    )
```

3. Jalankan AWS CLI perintah berikut untuk menyalin DAG ke bucket lingkungan Anda, lalu picu DAG menggunakan Apache Airflow UI.

```
aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. Jika berhasil, Anda akan menampilkan yang serupa dengan yang berikut di log tugas untuk `tz_aware_task` di `tz_test` DAG:

```
[2022-08-01, 12:00:00 PDT] {{subprocess.py:74}} INFO - Running command: ['bash', '-
c', 'date']
[2022-08-01, 12:00:00 PDT] {{subprocess.py:85}} INFO - Output:
[2022-08-01, 12:00:00 PDT] {{subprocess.py:89}} INFO - Mon Aug  1 12:00:00 PDT 2022
[2022-08-01, 12:00:00 PDT] {{subprocess.py:93}} INFO - Command exited with return
 code 0
[2022-08-01, 12:00:00 PDT] {{taskinstance.py:1280}} INFO - Marking task as
 SUCCESS. dag_id=tz_test, task_id=tz_aware_task, execution_date=20220801T190033,
 start_date=20220801T190035, end_date=20220801T190035
[2022-08-01, 12:00:00 PDT] {{local_task_job.py:154}} INFO - Task exited with return
 code 0
[2022-08-01, 12:00:00 PDT] {{local_task_job.py:264}} INFO - 0 downstream tasks
 scheduled from follow-on schedule check
```

Apa selanjutnya?

- Pelajari lebih lanjut cara mengunggah `plugins.zip` file dalam contoh ini ke bucket Amazon S3 Anda. Menginstal plugin kustom

Menyegarkan token CodeArtifact

Jika Anda menggunakan CodeArtifact untuk menginstal dependensi Python, Amazon MWAA memerlukan token aktif. Untuk memungkinkan Amazon MWAA mengakses CodeArtifact repositori saat runtime, Anda dapat menggunakan skrip startup dan menyetelnya dengan token. `PIP_EXTRA_INDEX_URL`

Topik berikut menjelaskan bagaimana Anda dapat membuat skrip startup yang menggunakan operasi `get_authorization_token`CodeArtifact API untuk mengambil token baru setiap kali lingkungan Anda dimulai, atau diperbarui.

Topik

- Versi
- Prasyarat
- Izin
- Contoh kode
- Apa selanjutnya?

Versi

Anda dapat menggunakan contoh kode pada halaman ini dengan Apache Airflow v2 di Python 3.10 dan Apache Airflowv3 di Python 3.11.

Prasyarat

Untuk menggunakan kode sampel di halaman ini, Anda memerlukan yang berikut:

- Lingkungan Amazon MWAA.
- CodeArtifact Repositori tempat Anda menyimpan dependensi untuk lingkungan Anda.

Izin

Untuk menyegarkan CodeArtifact token dan menulis hasilnya ke Amazon S3 Amazon MWAA harus memiliki izin berikut dalam peran eksekusi.

- `codeartifact:GetAuthorizationToken`Tindakan ini memungkinkan Amazon MWAA untuk mengambil token baru dari. CodeArtifact Kebijakan berikut memberikan izin untuk setiap

CodeArtifact domain yang Anda buat. Anda dapat membatasi akses ke domain Anda dengan memodifikasi nilai sumber daya dalam pernyataan, dan hanya menentukan domain yang ingin diakses oleh lingkungan Anda.

```
{
  "Effect": "Allow",
  "Action": "codeartifact:GetAuthorizationToken",
  "Resource": "arn:aws:codeartifact:us-west-2:*:domain/*"
}
```

- `sts:GetServiceBearerToken`Tindakan ini diperlukan untuk memanggil operasi CodeArtifact `GetAuthorizationToken`API. Operasi ini mengembalikan token yang harus digunakan saat menggunakan manajer paket seperti `pip` dengan CodeArtifact. Untuk menggunakan manajer paket dengan CodeArtifact repositori, peran peran eksekusi lingkungan Anda harus mengizinkan `sts:GetServiceBearerToken` seperti yang tercantum dalam pernyataan kebijakan berikut.

```
{
  "Sid": "AllowServiceBearerToken",
  "Effect": "Allow",
  "Action": "sts:GetServiceBearerToken",
  "Resource": "*"
}
```

Contoh kode

Langkah-langkah berikut menjelaskan bagaimana Anda dapat membuat skrip start up yang memperbarui CodeArtifact token.

1. Salin isi contoh kode berikut dan simpan secara lokal sebagai`code_artifact_startup_script.sh`.

```
#!/bin/sh

# Startup script for MWAA, refer to https://docs.aws.amazon.com/mwaa/latest/
userguide/using-startup-script.html

set -eu

# setup code artifact endpoint and token
# https://pip.pypa.io/en/stable/cli/pip_install/#cmdoption-0
```

```
# https://docs.aws.amazon.com/mwaa/latest/userguide/samples-code-artifact.html
DOMAIN="amazon"
DOMAIN_OWNER="112233445566"
REGION="us-west-2"
REPO_NAME="MyRepo"
echo "Getting token for CodeArtifact with args: --domain $DOMAIN --region $REGION
 --domain-owner $DOMAIN_OWNER"
TOKEN=$(aws codeartifact get-authorization-token --domain $DOMAIN --region $REGION
 --domain-owner $DOMAIN_OWNER | jq -r '.authorizationToken')
echo "Setting Pip env var for '--index-url' to point to CodeArtifact"
export PIP_EXTRA_INDEX_URL="https://aws:$TOKEN@$DOMAIN-
$DOMAIN_OWNER.d.codeartifact.$REGION.amazonaws.com/pypi/$REPO_NAME/simple/"
echo "CodeArtifact startup setup complete"
```

2. Arahkan ke folder tempat Anda menyimpan skrip. Gunakan `cp` di jendela prompt baru untuk mengunggah skrip ke bucket Anda. Ganti *amzn-s3-demo-bucket* dengan informasi Anda.

   ```
   aws s3 cp code_artifact_startup_script.sh s3://amzn-s3-demo-bucket/
   code_artifact_startup_script.sh
   ```

   Jika berhasil, Amazon S3 mengeluarkan jalur URL ke objek:

   ```
   upload: ./code_artifact_startup_script.sh to s3://amzn-s3-demo-bucket/
   code_artifact_startup_script.sh
   ```

   Setelah Anda mengunggah skrip, lingkungan Anda memperbarui dan menjalankan skrip saat startup.

Apa selanjutnya?

- Pelajari cara menggunakan skrip startup untuk menyesuaikan lingkungan Anda. the section called “Menggunakan skrip startup”
- Pelajari cara mengunggah kode DAG dalam contoh ini ke `dags` folder di bucket Amazon S3 Anda. Menambahkan atau memperbarui DAGs
- Pelajari lebih lanjut tentang cara mengunggah `plugins.zip` file dalam contoh ini ke bucket Amazon S3 Anda. Menginstal plugin kustom

Membuat plugin khusus dengan Apache Hive dan Hadoop

Amazon MWAA mengekstrak isi a `plugins.zip` to. `/usr/local/airflow/plugins` Ini dapat digunakan untuk menambahkan binari ke wadah Anda. Selain itu, Apache Airflow mengeksekusi isi file Python di folder saat startup —memungkinkan Anda untuk mengatur `plugins` dan memodifikasi variabel lingkungan. Contoh berikut memandu Anda melalui langkah-langkah untuk membuat plugin khusus menggunakan Apache Hive dan Hadoop di Amazon Managed Workflows untuk lingkungan Apache Airflow dan dapat dikombinasikan dengan plugin dan binari khusus lainnya.

Topik

- Versi
- Prasyarat
- Izin
- Persyaratan
- Unduh dependensi
- PluginKhusus
- Plugins.zip
- Contoh kode
- Opsi konfigurasi Airflow
- Apa selanjutnya?

Versi

Anda dapat menggunakan contoh kode pada halaman ini dengan Apache Airflow v2 di Python 3.10 dan Apache Airflowv3 di Python 3.11.

Prasyarat

Untuk menggunakan kode sampel di halaman ini, Anda memerlukan yang berikut:

- Lingkungan Amazon MWAA.

Izin

Tidak diperlukan izin tambahan untuk menggunakan contoh kode di halaman ini.

Persyaratan

Untuk menggunakan kode contoh pada halaman ini, tambahkan dependensi berikut ke Anda. `requirements.txt` Untuk mempelajari lebih lanjut, lihatMenginstal dependensi Python.

```
-c https://raw.githubusercontent.com/apache/airflow/constraints-2.0.2/
constraints-3.7.txt
apache-airflow-providers-amazon[apache.hive]
```

Unduh dependensi

Amazon MWAA akan mengekstrak konten plugins.zip ke dalam setiap penjadwal `/usr/local/airflow/plugins` Amazon MWAA dan wadah pekerja. Ini digunakan untuk menambahkan binari ke lingkungan Anda. Langkah-langkah berikut menjelaskan cara merakit file yang diperlukan untuk plugin kustom.

1. Di command prompt Anda, navigasikan ke direktori tempat Anda ingin membuat plugin Anda. Misalnya:

   ```
   cd plugins
   ```

2. Unduh Hadoop dari cermin, misalnya:

   ```
   wget https://downloads.apache.org/hadoop/common/hadoop-3.3.0/hadoop-3.3.0.tar.gz
   ```

3. Unduh Hive dari cermin, misalnya:

   ```
   wget https://downloads.apache.org/hive/hive-3.1.2/apache-hive-3.1.2-bin.tar.gz
   ```

4. Buatlah sebuah direktori. Misalnya:

   ```
   mkdir hive_plugin
   ```

5. Ekstrak Hadoop.

   ```
   tar -xvzf hadoop-3.3.0.tar.gz -C hive_plugin
   ```

6. Ekstrak Sarang.

   ```
   tar -xvzf apache-hive-3.1.2-bin.tar.gz -C hive_plugin
   ```

PluginKhusus

Apache Airflow akan mengeksekusi isi file Python di folder plugin saat startup. Ini digunakan untuk mengatur dan memodifikasi variabel lingkungan. Langkah-langkah berikut menjelaskan kode sampel untuk plugin kustom.

1. Di prompt perintah Anda, arahkan ke `hive_plugin` direktori. Misalnya:

```
cd hive_plugin
```

2. Salin isi contoh kode berikut dan simpan secara lokal seperti `hive_plugin.py` di `hive_plugin` direktori.

```
from airflow.plugins_manager import AirflowPlugin
import os
os.environ["JAVA_HOME"]="/usr/lib/jvm/jre"
os.environ["HADOOP_HOME"]='/usr/local/airflow/plugins/hadoop-3.3.0'
os.environ["HADOOP_CONF_DIR"]='/usr/local/airflow/plugins/hadoop-3.3.0/etc/hadoop'
os.environ["HIVE_HOME"]='/usr/local/airflow/plugins/apache-hive-3.1.2-bin'
os.environ["PATH"] = os.getenv("PATH") + ":/usr/local/airflow/plugins/
hadoop-3.3.0:/usr/local/airflow/plugins/apache-hive-3.1.2-bin/bin:/usr/local/
airflow/plugins/apache-hive-3.1.2-bin/lib"
os.environ["CLASSPATH"] = os.getenv("CLASSPATH") + ":/usr/local/airflow/plugins/
apache-hive-3.1.2-bin/lib"
class EnvVarPlugin(AirflowPlugin):
    name = 'hive_plugin'
```

3. Atasi konten teks berikut dan simpan secara lokal seperti `.airflowignore` di `hive_plugin` direktori.

```
hadoop-3.3.0
apache-hive-3.1.2-bin
```

Plugins.zip

Langkah-langkah berikut menjelaskan cara membuat`plugins.zip`. Isi contoh ini dapat dikombinasikan dengan plugin dan binari lain menjadi satu `plugins.zip` file.

1. Di prompt perintah Anda, arahkan ke `hive_plugin` direktori dari langkah sebelumnya. Misalnya:

```
cd hive_plugin
```

2. Zip konten di dalam `plugins` folder Anda.

```
zip -r ../hive_plugin.zip ./
```

Contoh kode

Langkah-langkah berikut menjelaskan cara membuat kode DAG yang akan menguji plugin kustom.

1. Di prompt perintah Anda, arahkan ke direktori tempat kode DAG Anda disimpan. Misalnya:

```
cd dags
```

2. Salin isi contoh kode berikut dan simpan secara lokal sebagai`hive.py`.

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from airflow.utils.dates import days_ago

with DAG(dag_id="hive_test_dag", schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
    hive_test = BashOperator(
        task_id="hive_test",
        bash_command='hive --help'
    )
```

Opsi konfigurasi Airflow

Jika Anda menggunakan Apache Airflow v2, `core.lazy_load_plugins : False` tambahkan sebagai opsi konfigurasi Apache Airflow. Untuk mempelajari lebih lanjut, lihat Menggunakan opsi konfigurasi untuk memuat plugin di 2.

Apa selanjutnya?

- Pelajari cara mengunggah `requirements.txt` file dalam contoh ini ke bucket Amazon S3 Anda. Menginstal dependensi Python

- Pelajari cara mengunggah kode DAG dalam contoh ini ke `dags` folder di bucket Amazon S3 Anda. Menambahkan atau memperbarui DAGs
- Pelajari lebih lanjut cara mengunggah `plugins.zip` file dalam contoh ini ke bucket Amazon S3 Anda. Menginstal plugin kustom

Membuat plugin khusus untuk Apache Airflow PythonVirtualenvOperator

Contoh berikut menjelaskan cara menambal Apache `PythonVirtualenvOperator` Airflow dengan plugin khusus di Amazon Managed Workflows untuk Apache Airflow.

Topik

- Versi
- Prasyarat
- Izin
- Persyaratan
- Kode sampel plugin kustom
- Plugins.zip
- Contoh kode
- Opsi konfigurasi Airflow
- Apa selanjutnya?

Versi

Anda dapat menggunakan contoh kode pada halaman ini dengan Apache Airflow v2 di Python 3.10 dan Apache Airflowv3 di Python 3.11.

Prasyarat

Untuk menggunakan kode sampel di halaman ini, Anda memerlukan yang berikut:

- Lingkungan Amazon MWAA.

Izin

Tidak ada izin tambahan yang diperlukan untuk menggunakan contoh kode di halaman ini.

Persyaratan

Untuk menggunakan kode sampel pada halaman ini, tambahkan dependensi berikut ke Anda. `requirements.txt` Untuk mempelajari lebih lanjut, lihat[Menginstal dependensi Python](#).

```
virtualenv
```

Kode sampel plugin kustom

Apache Airflow akan mengeksekusi isi file Python di folder plugin saat startup. Plugin ini akan menambal built-in `PythonVirtualenvOperator` selama proses startup itu agar kompatibel dengan Amazon MWAA. Langkah-langkah berikut menampilkan kode sampel untuk plugin kustom.

1. Di prompt perintah Anda, arahkan ke `plugins` direktori di bagian sebelumnya. Misalnya:

   ```
   cd plugins
   ```

2. Salin isi contoh kode berikut dan simpan secara lokal sebagai`virtual_python_plugin.py`.

   ```
   """
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

   Permission is hereby granted, free of charge, to any person obtaining a copy of
   this software and associated documentation files (the "Software"), to deal in
   the Software without restriction, including without limitation the rights to
   use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
   the Software, and to permit persons to whom the Software is furnished to do so.

   THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
   IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
   FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
   COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
   IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
   CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
   """
   from airflow.plugins_manager import AirflowPlugin
   import airflow.utils.python_virtualenv
   from typing import List
   ```

```
def _generate_virtualenv_cmd(tmp_dir: str, python_bin: str, system_site_packages:
 bool) -> List[str]:
    cmd = ['python3','/usr/local/airflow/.local/lib/python3.7/site-packages/
virtualenv', tmp_dir]
    if system_site_packages:
        cmd.append('--system-site-packages')
    if python_bin is not None:
        cmd.append(f'--python={python_bin}')
    return cmd

airflow.utils.python_virtualenv._generate_virtualenv_cmd=_generate_virtualenv_cmd

class VirtualPythonPlugin(AirflowPlugin):
    name = 'virtual_python_plugin'
```

Plugins.zip

Langkah-langkah berikut menjelaskan cara membuat`plugins.zip`.

1. Di prompt perintah Anda, arahkan ke direktori yang berisi `virtual_python_plugin.py` di bagian sebelumnya. Misalnya:

   ```
   cd plugins
   ```

2. Zip konten di dalam `plugins` folder Anda.

   ```
   zip plugins.zip virtual_python_plugin.py
   ```

Contoh kode

Langkah-langkah berikut menjelaskan cara membuat kode DAG untuk plugin kustom.

1. Di prompt perintah Anda, arahkan ke direktori tempat kode DAG Anda disimpan. Misalnya:

   ```
   cd dags
   ```

2. Salin isi contoh kode berikut dan simpan secara lokal sebagai`virtualenv_test.py`.

   ```
   """
   ```

```
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
from airflow import DAG
from airflow.operators.python import PythonVirtualenvOperator
from airflow.utils.dates import days_ago
import os

os.environ["PATH"] = os.getenv("PATH") + ":/usr/local/airflow/.local/bin"

def virtualenv_fn():
    import boto3
    print("boto3 version ",boto3.__version__)

with DAG(dag_id="virtualenv_test", schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
    virtualenv_task = PythonVirtualenvOperator(
        task_id="virtualenv_task",
        python_callable=virtualenv_fn,
        requirements=["boto3>=1.17.43"],
        system_site_packages=False,
        dag=dag,
    )
```

Opsi konfigurasi Airflow

Jika Anda menggunakan Apache Airflow v2, `core.lazy_load_plugins : False` tambahkan sebagai opsi konfigurasi Apache Airflow. Untuk mempelajari lebih lanjut, lihat Menggunakan opsi konfigurasi untuk memuat plugin di 2.

Apa selanjutnya?

- Pelajari cara mengunggah `requirements.txt` file dalam contoh ini ke bucket Amazon S3 Anda. Menginstal dependensi Python
- Pelajari cara mengunggah kode DAG dalam contoh ini ke `dags` folder di bucket Amazon S3 Anda. Menambahkan atau memperbarui DAGs
- Pelajari lebih lanjut cara mengunggah `plugins.zip` file dalam contoh ini ke bucket Amazon S3 Anda. Menginstal plugin kustom

Memanggil DAGs dengan fungsi Lambda

Contoh kode berikut menggunakan AWS Lambdafungsi untuk mendapatkan token CLI Apache Airflow dan memanggil grafik asiklik terarah (DAG) di lingkungan Amazon MWAA.

Topik

- Versi
- Prasyarat
- Izin
- Dependensi
- Contoh kode

Versi

Anda dapat menggunakan contoh kode pada halaman ini dengan Apache Airflow v2 di Python 3.10 dan Apache Airflowv3 di Python 3.11.

Prasyarat

Untuk menggunakan contoh kode ini, Anda harus:

- Gunakan mode akses jaringan publik untuk lingkungan Amazon MWAA Anda.
- Memiliki fungsi Lambda menggunakan runtime Python terbaru.

> **Note**
>
> Jika fungsi Lambda dan lingkungan Amazon MWAA Anda berada di VPC yang sama, Anda dapat menggunakan kode ini di jaringan pribadi. Untuk konfigurasi ini, peran eksekusi fungsi Lambda memerlukan izin untuk memanggil operasi API Amazon Elastic Compute Cloud (Amazon EC2). CreateNetworkInterface Anda dapat memberikan izin ini menggunakan kebijakan `AWSLambdaVPCAccessExecutionRole` AWS-managed.

Izin

Untuk menggunakan contoh kode di halaman ini, peran eksekusi lingkungan Amazon MWAA Anda memerlukan akses untuk melakukan tindakan. `airflow:CreateCliToken` Anda dapat memberikan izin ini menggunakan kebijakan `AmazonMWAAAirflowCliAccess` AWS-managed:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "airflow:CreateCliToken"
            ],
            "Resource": "*"
        }
    ]
}
```

Untuk informasi lebih lanjut, lihatKebijakan CLI Apache Airflow: AmazonMWAAAirflowCliAccess.

Dependensi

Untuk menggunakan contoh kode ini dengan Apache Airflow v2 dan yang lebih baru, tidak diperlukan dependensi tambahan. Gunakan aws-mwaa-docker-imagesuntuk menginstal Apache Airflow.

Contoh kode

1. Buka AWS Lambda konsol di https://console.aws.amazon.com/lambda/.
2. Pilih fungsi Lambda Anda dari daftar Fungsi.
3. Pada halaman fungsi, salin kode berikut dan ganti yang berikut ini dengan nama sumber daya Anda:

   - `YOUR_ENVIRONMENT_NAME`— Nama lingkungan Amazon MWAA Anda.
   - `YOUR_DAG_NAME`— Nama DAG yang ingin Anda panggil.

```
import boto3
import http.client
import base64
import ast
mwaa_env_name = 'YOUR_ENVIRONMENT_NAME'
dag_name = 'YOUR_DAG_NAME'
mwaa_cli_command = 'dags trigger'

client = boto3.client('mwaa')

def lambda_handler(event, context):
    # get web token
    mwaa_cli_token = client.create_cli_token(
        Name=mwaa_env_name
    )

    conn = http.client.HTTPSConnection(mwaa_cli_token['WebServerHostname'])
    payload = mwaa_cli_command + " " + dag_name
    headers = {
      'Authorization': 'Bearer ' + mwaa_cli_token['CliToken'],
      'Content-Type': 'text/plain'
    }
    conn.request("POST", "/aws_mwaa/cli", payload, headers)
    res = conn.getresponse()
    data = res.read()
    dict_str = data.decode("UTF-8")
    mydata = ast.literal_eval(dict_str)
    return base64.b64decode(mydata['stdout'])
```

4. Pilih Deploy.

5. Pilih Uji untuk menjalankan fungsi Anda menggunakan konsol Lambda.
6. Untuk memverifikasi bahwa Lambda berhasil memanggil DAG, gunakan konsol Amazon MWAA untuk menavigasi ke UI Apache Airflow lingkungan Anda, lalu lakukan hal berikut:
   a. Pada DAGshalaman, cari DAG target baru Anda dalam daftar DAGs.
   b. Di bawah Last Run, periksa stempel waktu untuk menjalankan DAG terbaru. Stempel waktu ini harus sangat cocok dengan stempel waktu terbaru `invoke_dag` di lingkungan Anda yang lain.
   c. Di bawah Tugas Terbaru, periksa apakah proses terakhir berhasil.

Memanggil DAGs di lingkungan MWAA Amazon yang berbeda

Contoh kode berikut membuat token Apache Airflow CLI. Kode tersebut kemudian menggunakan grafik asiklik terarah (DAG) di satu lingkungan Amazon MWAA untuk memanggil DAG di lingkungan MWAA Amazon yang berbeda.

Topik

- Versi
- Prasyarat
- Izin
- Dependensi
- Contoh kode

Versi

Anda dapat menggunakan contoh kode pada halaman ini dengan Apache Airflow v2 di Python 3.10 dan Apache Airflowv3 di Python 3.11.

Prasyarat

Untuk menggunakan contoh kode pada halaman ini, Anda memerlukan yang berikut:

- Dua lingkungan Amazon MWAA dengan akses server web jaringan publik, termasuk lingkungan Anda saat ini.
- Contoh DAG yang diunggah ke bucket Amazon Simple Storage Service (Amazon S3) lingkungan target Anda.

Izin

Untuk menggunakan contoh kode di halaman ini, peran eksekusi lingkungan Anda harus memiliki izin untuk membuat token CLI Apache Airflow. Anda dapat melampirkan kebijakan AWS-managed `AmazonMWAAAirflowCliAccess` untuk memberikan izin ini.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "airflow:CreateCliToken"
            ],
            "Resource": "*"
        }
    ]
}
```

Untuk informasi lebih lanjut, lihatKebijakan CLI Apache Airflow: AmazonMWAAAirflowCliAccess.

Dependensi

Untuk menggunakan contoh kode ini dengan Apache Airflow v2 dan yang lebih baru, tidak diperlukan dependensi tambahan. Gunakan aws-mwaa-docker-imagesuntuk menginstal Apache Airflow.

Contoh kode

Contoh kode berikut mengasumsikan bahwa Anda menggunakan DAG di lingkungan Anda saat ini untuk memanggil DAG di lingkungan lain.

1. Di terminal Anda, arahkan ke direktori tempat kode DAG Anda disimpan. Contoh:

   ```
   cd dags
   ```

2. Salin konten contoh kode berikut dan simpan secara lokal sebagai`invoke_dag.py`. Ganti nilai berikut dengan informasi Anda.

- `your-new-environment-name`— Nama lingkungan lain di mana Anda ingin memanggil DAG.
- `your-target-dag-id`— ID DAG di lingkungan lain yang ingin Anda panggil.

```
from airflow.decorators import dag, task
import boto3
from datetime import datetime, timedelta
import os, requests

DAG_ID = os.path.basename(__file__).replace(".py", "")

@task()
def invoke_dag_task(**kwargs):
    client = boto3.client('mwaa')
    token = client.create_cli_token(Name='your-new-environment-name')
    url = f"https://{token['WebServerHostname']}/aws_mwaa/cli"
    body = 'dags trigger your-target-dag-id'
    headers = {
        'Authorization' : 'Bearer ' + token['CliToken'],
        'Content-Type': 'text/plain'
        }
    requests.post(url, data=body, headers=headers)

@dag(
    dag_id=DAG_ID,
    schedule_interval=None,
    start_date=datetime(2022, 1, 1),
    dagrun_timeout=timedelta(minutes=60),
    catchup=False
    )
def invoke_dag():
    t = invoke_dag_task()

invoke_dag_test = invoke_dag()
```

3. Jalankan AWS CLI perintah berikut untuk menyalin DAG ke bucket lingkungan Anda, lalu picu DAG menggunakan Apache Airflow UI.

```
aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. Jika DAG berhasil berjalan, Anda akan mendapatkan output yang mirip dengan yang berikut di log tugas untuk`invoke_dag_task`.

```
[2022-01-01, 12:00:00 PDT] {{python.py:152}} INFO - Done. Returned value was: None
[2022-01-01, 12:00:00 PDT] {{taskinstance.py:1280}} INFO - Marking task as SUCCESS.
 dag_id=invoke_dag, task_id=invoke_dag_task, execution_date=20220101T120000,
 start_date=20220101T120000, end_date=20220101T120000
[2022-01-01, 12:00:00 PDT] {{local_task_job.py:154}} INFO - Task exited with return
 code 0
[2022-01-01, 12:00:00 PDT] {{local_task_job.py:264}} INFO - 0 downstream tasks
 scheduled from follow-on schedule check
```

   Untuk memverifikasi bahwa DAG Anda berhasil dipanggil, navigasikan ke Apache Airflow UI untuk lingkungan baru Anda, lalu lakukan hal berikut:

   a. Pada DAGshalaman, cari DAG target baru Anda dalam daftar DAGs.
   b. Di bawah Last Run, periksa stempel waktu untuk menjalankan DAG terbaru. Stempel waktu ini harus sangat cocok dengan stempel waktu terbaru `invoke_dag` di lingkungan Anda yang lain.
   c. Di bawah Tugas Terbaru, periksa apakah proses terakhir berhasil.

Menggunakan Amazon MWAA dengan Amazon RDS untuk Microsoft SQL Server

Anda dapat menggunakan Alur Kerja Terkelola Amazon untuk Apache Airflow untuk menyambung ke RDS untuk SQL Server. Contoh kode berikut digunakan DAGs pada Amazon Managed Workflows untuk lingkungan Apache Airflow untuk menyambung dan mengeksekusi kueri pada Amazon RDS for Microsoft SQL Server.

Topik

- Versi
- Prasyarat
- Dependensi
- Koneksi Apache Airflow v2
- Contoh kode
- Apa selanjutnya?

Versi

Anda dapat menggunakan contoh kode pada halaman ini dengan Apache Airflow v2 di Python 3.10 dan Apache Airflowv3 di Python 3.11.

Prasyarat

Untuk menggunakan kode sampel di halaman ini, Anda memerlukan yang berikut:

- Lingkungan Amazon MWAA.
- Amazon MWAA dan RDS untuk SQL Server berjalan di Amazon VPC yang sama/
- Grup keamanan VPC Amazon MWAA dan server dikonfigurasi dengan koneksi berikut:
  - Aturan masuk untuk port 1433 terbuka untuk Amazon RDS di grup keamanan Amazon MWAA
  - Atau aturan keluar untuk port 1433 terbuka dari Amazon MWAA Ke RDS
- Apache Airflow Connection untuk RDS untuk SQL Server mencerminkan nama host, port, nama pengguna, dan kata sandi dari database server Amazon RDS SQL yang dibuat dalam proses sebelumnya.

Dependensi

Untuk menggunakan kode sampel di bagian ini, tambahkan dependensi berikut ke kode Anda`requirements.txt`. Untuk mempelajari lebih lanjut, lihatMenginstal dependensi Python.

```
apache-airflow-providers-microsoft-mssql==1.0.1
   apache-airflow-providers-odbc==1.0.1
   pymssql==2.2.1
```

Koneksi Apache Airflow v2

Jika Anda menggunakan koneksi di Apache Airflow v2, pastikan objek koneksi Airflow menyertakan pasangan nilai kunci berikut:

1. Id Konn: mssql_default
2. Jenis Conn: Amazon Web Services
3. Tuan rumah: `YOUR_DB_HOST`
4. Skema:

5. Masuk: admin
6. Kata Sandi:
7. Pelabuhan: 1433
8. Ekstra:

Contoh kode

1. Di prompt perintah Anda, arahkan ke direktori tempat kode DAG Anda disimpan. Contoh:

```
cd dags
```

2. Salin isi contoh kode berikut dan simpan secara lokal sebagai`sql-server.py`.

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
import pymssql
import logging
import sys
from airflow import DAG
from datetime import datetime
from airflow.operators.mssql_operator import MsSqlOperator
from airflow.operators.python_operator import PythonOperator

default_args = {
    'owner': 'aws',
    'depends_on_past': False,
    'start_date': datetime(2019, 2, 20),
    'provide_context': True
}
```

```
dag = DAG(
    'mssql_conn_example', default_args=default_args, schedule_interval=None)

drop_db = MsSqlOperator(
   task_id="drop_db",
   sql="DROP DATABASE IF EXISTS testdb;",
   mssql_conn_id="mssql_default",
   autocommit=True,
   dag=dag
)

create_db = MsSqlOperator(
   task_id="create_db",
   sql="create database testdb;",
   mssql_conn_id="mssql_default",
   autocommit=True,
   dag=dag
)

create_table = MsSqlOperator(
   task_id="create_table",
   sql="CREATE TABLE testdb.dbo.pet (name VARCHAR(20), owner VARCHAR(20));",
   mssql_conn_id="mssql_default",
   autocommit=True,
   dag=dag
)

insert_into_table = MsSqlOperator(
   task_id="insert_into_table",
   sql="INSERT INTO testdb.dbo.pet VALUES ('Olaf', 'Disney');",
   mssql_conn_id="mssql_default",
   autocommit=True,
   dag=dag
)

def select_pet(**kwargs):
   try:
        conn = pymssql.connect(
            server='sampledb.<xxxxxx>.<region>.rds.amazonaws.com',
            user='admin',
            password='<yoursupersecretpassword>',
            database='testdb'
        )
```

```
        # Create a cursor from the connection
        cursor = conn.cursor()
        cursor.execute("SELECT * from testdb.dbo.pet")
        row = cursor.fetchone()

        if row:
            print(row)
   except:
      logging.error("Error when creating pymssql database connection: %s",
 sys.exc_info()[0])

select_query = PythonOperator(
    task_id='select_query',
    python_callable=select_pet,
    dag=dag,
)

drop_db >> create_db >> create_table >> insert_into_table >> select_query
```

Apa selanjutnya?

- Pelajari cara mengunggah `requirements.txt` file dalam contoh ini ke bucket Amazon S3 Anda. Menginstal dependensi Python
- Pelajari cara mengunggah kode DAG dalam contoh ini ke `dags` folder di bucket Amazon S3 Anda. Menambahkan atau memperbarui DAGs
- Jelajahi contoh skrip dan contoh modul pymssql lainnya.
- Pelajari lebih lanjut tentang mengeksekusi kode SQL dalam database Microsoft SQL tertentu menggunakan mssql_operator dalam panduan referensi Apache Airflow.

Menggunakan Amazon MWAA dengan Amazon EKS

Contoh berikut menunjukkan cara menggunakan Alur Kerja Terkelola Amazon untuk Apache Airflow dengan Amazon EKS.

Topik

- Versi
- Prasyarat

- Buat kunci publik untuk Amazon EC2
- Buat cluster
- Buat mwaa namespace
- Buat peran untuk mwaa namespace
- Buat dan lampirkan peran IAM untuk klaster Amazon EKS
- Buat file requirements.txt
- Membuat pemetaan identitas untuk Amazon EKS
- Buat kubeconfig
- Buat DAG
- Tambahkan DAG dan kube_config.yaml ke ember Amazon S3
- Aktifkan dan picu contoh

Versi

Anda dapat menggunakan contoh kode pada halaman ini dengan Apache Airflow v2 di Python 3.10 dan Apache Airflowv3 di Python 3.11.

Prasyarat

Untuk menggunakan contoh dalam topik ini, Anda memerlukan yang berikut:

- Lingkungan Amazon MWAA.
- eksctl. Untuk mempelajari lebih lanjut, lihat Instal eksctl.
- kubectl. Untuk mempelajari lebih lanjut, lihat Menginstal dan Mengatur kubectl. Dalam beberapa kasus ini diinstal dengan eksctl.
- Sebuah key pair EC2 di Wilayah tempat Anda membuat lingkungan Amazon MWAA Anda. Untuk mempelajari lebih lanjut, lihat Membuat atau mengimpor key pair.

> **Note**
>
> Bila Anda menggunakan `eksctl` perintah, Anda dapat menyertakan `--profile` untuk menentukan profil selain default.

Buat kunci publik untuk Amazon EC2

Gunakan perintah berikut untuk membuat kunci publik dari private key pair Anda.

```
ssh-keygen -y -f myprivatekey.pem > mypublickey.pub
```

Untuk mempelajari lebih lanjut, lihat Mengambil kunci publik untuk key pair Anda.

Buat cluster

Gunakan perintah berikut untuk membuat cluster. Jika Anda menginginkan nama khusus untuk klaster atau membuatnya di Wilayah yang berbeda, ganti nama dan nilai Wilayah. Anda harus membuat cluster di Wilayah yang sama tempat Anda membuat lingkungan Amazon MWAA. Ganti nilai subnet agar sesuai dengan subnet di jaringan VPC Amazon Anda yang Anda gunakan untuk Amazon MWAA. Ganti nilai untuk `ssh-public-key` mencocokkan kunci yang Anda gunakan. Anda dapat menggunakan kunci yang ada dari Amazon EC2 yang berada di Wilayah yang sama, atau membuat kunci baru di Wilayah yang sama tempat Anda membuat lingkungan Amazon MWAA.

```
eksctl create cluster \
--name mwaa-eks \
--region us-west-2 \
--version 1.18 \
--nodegroup-name linux-nodes \
--nodes 3 \
--nodes-min 1 \
--nodes-max 4 \
--with-oidc \
--ssh-access \
--ssh-public-key MyPublicKey \
--managed \
--vpc-public-subnets "subnet-1111111111111111, subnet-2222222222222222" \
--vpc-private-subnets "subnet-33333333333333333, subnet-44444444444444444"
```

Butuh beberapa waktu untuk menyelesaikan pembuatan cluster. Setelah selesai, Anda dapat memverifikasi bahwa cluster berhasil dibuat dan Penyedia IAM OIDC dikonfigurasi dengan menggunakan perintah berikut:

```
eksctl utils associate-iam-oidc-provider \
--region us-west-2 \
--cluster mwaa-eks \
```

```
--approve
```

Buat **mwaa** namespace

Setelah mengonfirmasi bahwa cluster berhasil dibuat, gunakan perintah berikut untuk membuat namespace untuk pod.

```
kubectl create namespace mwaa
```

Buat peran untuk **mwaa** namespace

Setelah membuat namespace, buat role dan role-binding untuk pengguna Amazon MWAA di EKS yang dapat menjalankan pod di namespace MWAA. Jika Anda menggunakan nama yang berbeda untuk namespace, ganti mwaa in `-n` *mwaa* dengan nama yang Anda gunakan.

```
cat << EOF | kubectl apply -f - -n mwaa
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
name: mwaa-role
rules:
  - apiGroups:
  - ""
  - "apps"
  - "batch"
  - "extensions"
resources:
  - "jobs"
  - "pods"
  - "pods/attach"
   - "pods/exec"
  - "pods/log"
  - "pods/portforward"
  - "secrets"
  - "services"
verbs:
  - "create"
  - "delete"
  - "describe"
  - "get"
  - "list"
  - "patch"
```

```
  - "update"
---
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: mwaa-role-binding
  subjects:
    - kind: User
  name: mwaa-service
  roleRef:
    kind: Role
  name: mwaa-role
  apiGroup: rbac.authorization.k8s.io
EOF
```

Konfirmasikan bahwa peran baru dapat mengakses kluster Amazon EKS dengan menjalankan perintah berikut. Pastikan untuk menggunakan nama yang benar jika Anda tidak menggunakan*mwaa*:

```
kubectl get pods -n mwaa --as mwaa-service
```

Anda mendapatkan pesan yang dikembalikan yang mengatakan:

```
No resources found in mwaa namespace.
```

Buat dan lampirkan peran IAM untuk klaster Amazon EKS

Anda harus membuat peran IAM dan kemudian mengikatnya ke cluster Amazon EKS (k8s) sehingga dapat digunakan untuk otentikasi melalui IAM. Peran ini hanya digunakan untuk masuk ke cluster, dan tidak memiliki izin apa pun untuk panggilan konsol atau API.

Buat peran baru untuk lingkungan Amazon MWAA menggunakan langkah-langkah di. Peran eksekusi Amazon MWAA Namun, alih-alih membuat dan melampirkan kebijakan yang dijelaskan dalam topik itu, lampirkan kebijakan berikut:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": "airflow:PublishMetrics",
            "Resource": "arn:aws:airflow:us-east-1:111122223333:environment/
${MWAA_ENV_NAME}"
        },
        {
            "Effect": "Deny",
            "Action": "s3:ListAllMyBuckets",
            "Resource": [
                "arn:aws:s3:::{MWAA_S3_BUCKET}",
                "arn:aws:s3:::{MWAA_S3_BUCKET}/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject*",
                "s3:GetBucket*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::{MWAA_S3_BUCKET}",
                "arn:aws:s3:::{MWAA_S3_BUCKET}/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "logs:PutLogEvents",
                "logs:GetLogEvents",
                "logs:GetLogRecord",
                "logs:GetLogGroupFields",
                "logs:GetQueryResults",
                "logs:DescribeLogGroups"
            ],
            "Resource": [
            "arn:aws:logs:us-east-1:111122223333:log-group:airflow-
${MWAA_ENV_NAME}-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricData",
```

```
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "sqs:ChangeMessageVisibility",
                "sqs:DeleteMessage",
                "sqs:GetQueueAttributes",
                "sqs:GetQueueUrl",
                "sqs:ReceiveMessage",
                "sqs:SendMessage"
            ],
            "Resource": "arn:aws:sqs:us-east-1:*:airflow-celery-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey*",
                "kms:Encrypt"
            ],
            "NotResource": "arn:aws:kms:*:111122223333:key/*",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": [
                    "sqs.us-east-1.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster"
            ],
            "Resource": "arn:aws:eks:us-east-1:111122223333:cluster/
${EKS_CLUSTER_NAME}"
        }
    ]
}
```

Setelah Anda membuat peran, edit lingkungan Amazon MWAA Anda untuk menggunakan peran yang Anda buat sebagai peran eksekusi untuk lingkungan. Untuk mengubah peran, edit lingkungan yang akan digunakan. Anda memilih peran eksekusi di bawah Izin.

Masalah yang diketahui:

- Ada masalah yang diketahui dengan peran ARNs dengan subpath yang tidak dapat diautentikasi dengan Amazon EKS. Solusi untuk ini adalah membuat peran layanan secara manual daripada menggunakan yang dibuat oleh Amazon MWAA itu sendiri. Untuk mempelajari lebih lanjut, lihat Peran dengan jalur tidak berfungsi saat jalur disertakan dalam ARN mereka di configmap aws-auth
- Jika daftar layanan Amazon MWAA tidak tersedia di IAM, Anda harus memilih kebijakan layanan alternatif, seperti Amazon EC2, lalu memperbarui kebijakan kepercayaan peran agar sesuai dengan yang berikut:

  JSON

  ```
  {
    "Version":"2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
          "Service": [
            "airflow-env.amazonaws.com",
            "airflow.amazonaws.com"
          ]
        },
        "Action": "sts:AssumeRole"
      }
    ]
  }
  ```

  Untuk mempelajari lebih lanjut, lihat Cara menggunakan kebijakan kepercayaan dengan peran IAM.

Buat file requirements.txt

Untuk menggunakan kode sampel di bagian ini, pastikan Anda telah menambahkan salah satu opsi database berikut ke bagian Anda`requirements.txt`. Untuk mempelajari lebih lanjut, lihatMenginstal dependensi Python.

```
kubernetes
apache-airflow[cncf.kubernetes]==3.0.0
```

Membuat pemetaan identitas untuk Amazon EKS

Gunakan ARN untuk peran yang Anda buat dalam perintah berikut untuk membuat pemetaan identitas untuk Amazon EKS. Ubah Wilayah *us-east-1* ke Wilayah tempat Anda menciptakan lingkungan. Ganti ARN untuk peran tersebut, dan terakhir, ganti *mwaa-execution-role* dengan peran eksekusi lingkungan Anda.

```
eksctl create iamidentitymapping \
--region us-east-1 \
--cluster mwaa-eks \
--arn arn:aws:iam::123456789012:role/mwaa-execution-role \
--username mwaa-service
```

Buat **kubeconfig**

Gunakan perintah berikut untuk membuat`kubeconfig`:

```
aws eks update-kubeconfig \
--region us-west-2 \
--kubeconfig ./kube_config.yaml \
--name mwaa-eks \
--alias aws
```

Jika Anda menggunakan profil tertentu saat menjalankan, `update-kubeconfig` Anda perlu menghapus `env:` bagian yang ditambahkan ke file kube_config.yaml sehingga berfungsi dengan benar dengan Amazon MWAA. Untuk melakukannya, hapus yang berikut dari file dan kemudian simpan:

```
env:
 - name: AWS_PROFILE
 value: profile_name
```

Buat DAG

Gunakan contoh kode berikut untuk membuat file Python, seperti `mwaa_pod_example.py` untuk DAG.

```
"""
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""
from airflow import DAG
from datetime import datetime
from airflow.providers.cncf.kubernetes.operators.kubernetes_pod import
 KubernetesPodOperator

default_args = {
   'owner': 'aws',
   'depends_on_past': False,
   'start_date': datetime(2019, 2, 20),
   'provide_context': True
}

dag = DAG(
   'kubernetes_pod_example', default_args=default_args, schedule_interval=None)

#use a kube_config stored in s3 dags folder for now
kube_config_path = '/usr/local/airflow/dags/kube_config.yaml'

podRun = KubernetesPodOperator(
                       namespace="mwaa",
                       image="ubuntu:18.04",
                       cmds=["bash"],
                       arguments=["-c", "ls"],
                       labels={"foo": "bar"},
                       name="mwaa-pod-test",
                       task_id="pod-task",
                       get_logs=True,
                       dag=dag,
                       is_delete_operator_pod=False,
```

```
                    config_file=kube_config_path,
                    in_cluster=False,
                    cluster_context='aws'
                    )
```

Tambahkan DAG dan **kube_config.yaml** ke ember Amazon S3

Masukkan DAG yang Anda buat dan `kube_config.yaml` file ke dalam bucket Amazon S3 untuk lingkungan Amazon MWAA. Anda dapat memasukkan file ke dalam bucket menggunakan konsol Amazon S3 atau. AWS Command Line Interface

Aktifkan dan picu contoh

Di Apache Airflow, aktifkan contoh dan kemudian picu.

Setelah berjalan dan selesai dengan sukses, gunakan perintah berikut untuk memverifikasi pod:

```
kubectl get pods -n mwaa
```

Anda mendapatkan output yang mirip dengan berikut ini:

```
NAME READY STATUS RESTARTS AGE
mwaa-pod-test-aa11bb22cc334444555566667778888 0/1 Completed 0 2m23s
```

Anda kemudian dapat memverifikasi output pod dengan perintah berikut. Ganti nilai nama dengan nilai yang dikembalikan dari perintah sebelumnya:

```
kubectl logs -n mwaa mwaa-pod-test-aa11bb22cc334444555566667778888
```

Menghubungkan ke Amazon ECS menggunakan **ECSOperator**

Topik ini menjelaskan bagaimana Anda dapat menggunakan wadah Amazon Elastic Container Service (Amazon ECS) dari Amazon MWAA. `ECSOperator` Pada langkah-langkah berikut, Anda akan menambahkan izin yang diperlukan ke peran eksekusi lingkungan Anda, menggunakan CloudFormation template untuk membuat klaster Amazon ECS Fargate, dan terakhir membuat dan mengunggah DAG yang terhubung ke cluster baru Anda.

Topik

- Versi
- Prasyarat
- Izin
- Buat kluster Amazon ECS
- Contoh kode

Versi

Anda dapat menggunakan contoh kode pada halaman ini dengan Apache Airflow v2 di Python 3.10 dan Apache Airflowv3 di Python 3.11.

Prasyarat

Untuk menggunakan kode sampel di halaman ini, Anda memerlukan yang berikut:

- Lingkungan Amazon MWAA.

Izin

- Peran eksekusi untuk lingkungan Anda memerlukan izin untuk menjalankan tugas di Amazon ECS. Anda dapat melampirkan kebijakan FullAccess AWS terkelola Amazonecs_ ke peran eksekusi, atau membuat dan melampirkan kebijakan berikut ke peran eksekusi Anda.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask",
                "ecs:DescribeTasks"
            ],
            "Resource": "*"
        },
        {
            "Action": "iam:PassRole",
```

```
            "Effect": "Allow",
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "ecs-tasks.amazonaws.com"
                }
            }
        }
    ]
}
```

- Selain menambahkan premisi yang diperlukan untuk menjalankan tugas di Amazon ECS, Anda juga harus mengubah pernyataan kebijakan CloudWatch Log dalam peran eksekusi Amazon MWAA Anda untuk mengizinkan akses ke grup log tugas Amazon ECS seperti yang tercantum di bawah ini. Grup log Amazon ECS dibuat oleh CloudFormation template dithe section called “Buat kluster Amazon ECS”.

```
{
  "Effect": "Allow",
  "Action": [
    "logs:CreateLogStream",
    "logs:CreateLogGroup",
    "logs:PutLogEvents",
    "logs:GetLogEvents",
    "logs:GetLogRecord",
    "logs:GetLogGroupFields",
    "logs:GetQueryResults"
  ],
  "Resource": [
    "arn:aws:logs:us-east-1:123456789012:log-group:airflow-environment-name-*",
    "arn:aws:logs:*:*:log-group:ecs-mwaa-group:*"
  ]
}
```

Untuk informasi selengkapnya tentang peran eksekusi Amazon MWAA, dan cara melampirkan kebijakan, lihat. Peran eksekusi

Buat kluster Amazon ECS

Dengan menggunakan CloudFormation template berikut, Anda akan membangun cluster Amazon ECS Fargate untuk digunakan dengan alur kerja Amazon MWAA Anda. Untuk informasi selengkapnya, lihat Membuat definisi tugas di Amazon Elastic Container Service Developer Guide.

1. Buat file JSON dengan kode berikut dan simpan sebagai`ecs-mwaa-cfn.json`.

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "This template deploys an ECS Fargate cluster with an Amazon
 Linux image as a test for MWAA.",
    "Parameters": {
        "VpcId": {
            "Type": "AWS::EC2::VPC::Id",
            "Description": "Select a VPC that allows instances access to ECR, as
 used with MWAA."
        },
        "SubnetIds": {
            "Type": "List<AWS::EC2::Subnet::Id>",
            "Description": "Select at two private subnets in your selected VPC, as
 used with MWAA."
        },
        "SecurityGroups": {
            "Type": "List<AWS::EC2::SecurityGroup::Id>",
            "Description": "Select at least one security group in your selected
 VPC, as used with MWAA."
        }
    },
    "Resources": {
        "Cluster": {
            "Type": "AWS::ECS::Cluster",
            "Properties": {
                "ClusterName": {
                    "Fn::Sub": "${AWS::StackName}-cluster"
                }
            }
        },
        "LogGroup": {
            "Type": "AWS::Logs::LogGroup",
            "Properties": {
                "LogGroupName": {
                    "Ref": "AWS::StackName"
```

```
                },
                "RetentionInDays": 30
            }
        },
        "ExecutionRole": {
            "Type": "AWS::IAM::Role",
            "Properties": {
                "AssumeRolePolicyDocument": {
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": "ecs-tasks.amazonaws.com"
                            },
                            "Action": "sts:AssumeRole"
                        }
                    ]
                },
                "ManagedPolicyArns": [
                    "arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy"
                ]
            }
        },
        "TaskDefinition": {
            "Type": "AWS::ECS::TaskDefinition",
            "Properties": {
                "Family": {
                    "Fn::Sub": "${AWS::StackName}-task"
                },
                "Cpu": 2048,
                "Memory": 4096,
                "NetworkMode": "awsvpc",
                "ExecutionRoleArn": {
                    "Ref": "ExecutionRole"
                },
                "ContainerDefinitions": [
                    {
                        "Name": {
                            "Fn::Sub": "${AWS::StackName}-container"
                        },
                        "Image": "137112412989.dkr.ecr.us-east-1.amazonaws.com/
amazonlinux:latest",
                        "PortMappings": [
```

```
                        {
                            "Protocol": "tcp",
                            "ContainerPort": 8080,
                            "HostPort": 8080
                        }
                    ],
                    "LogConfiguration": {
                        "LogDriver": "awslogs",
                        "Options": {
                            "awslogs-region": {
                                "Ref": "AWS::Region"
                            },
                            "awslogs-group": {
                                "Ref": "LogGroup"
                            },
                            "awslogs-stream-prefix": "ecs"
                        }
                    }
                }
            ],
            "RequiresCompatibilities": [
                "FARGATE"
            ]
        }
    },
    "Service": {
        "Type": "AWS::ECS::Service",
        "Properties": {
            "ServiceName": {
                "Fn::Sub": "${AWS::StackName}-service"
            },
            "Cluster": {
                "Ref": "Cluster"
            },
            "TaskDefinition": {
                "Ref": "TaskDefinition"
            },
            "DesiredCount": 1,
            "LaunchType": "FARGATE",
            "PlatformVersion": "1.3.0",
            "NetworkConfiguration": {
                "AwsvpcConfiguration": {
                    "AssignPublicIp": "ENABLED",
                    "Subnets": {
```

```
                            "Ref": "SubnetIds"
                        },
                        "SecurityGroups": {
                            "Ref": "SecurityGroups"
                        }
                    }
                }
            }
        }
    }
}
```

2. Di command prompt Anda, gunakan AWS CLI perintah berikut untuk membuat tumpukan baru. Anda harus mengganti nilai `SecurityGroups` dan `SubnetIds` dengan nilai untuk grup keamanan dan subnet lingkungan Amazon MWAA Anda.

```
aws cloudformation create-stack \
--stack-name my-ecs-stack --template-body file://ecs-mwaa-cfn.json \
--parameters ParameterKey=SecurityGroups,ParameterValue=your-mwaa-security-group \
ParameterKey=SubnetIds,ParameterValue=your-mwaa-subnet-1\\,your-mwaa-subnet-1 \
--capabilities CAPABILITY_IAM
```

Atau, Anda dapat menggunakan skrip shell berikut. Skrip mengambil nilai yang diperlukan untuk grup keamanan lingkungan Anda, dan subnet menggunakan `get-environment` AWS CLI perintah, lalu membuat tumpukan yang sesuai. Untuk menjalankan skrip, lakukan hal berikut.

a. Salin, dan simpan skrip seperti `ecs-stack-helper.sh` di direktori yang sama dengan CloudFormation template Anda.

```
#!/bin/bash

joinByString() {
  local separator="$1"
  shift
  local first="$1"
  shift
  printf "%s" "$first" "${@/#/$separator}"
}

response=$(aws mwaa get-environment --name $1)
```

```
securityGroupId=$(echo "$response" | jq -r
 '.Environment.NetworkConfiguration.SecurityGroupIds[]')
subnetIds=$(joinByString '\,' $(echo "$response" | jq -r
 '.Environment.NetworkConfiguration.SubnetIds[]'))

aws cloudformation create-stack --stack-name $2 --template-body file://ecs-
cfn.json \
--parameters ParameterKey=SecurityGroups,ParameterValue=$securityGroupId \
ParameterKey=SubnetIds,ParameterValue=$subnetIds \
--capabilities CAPABILITY_IAM
```

b. Jalankan skrip menggunakan perintah berikut. Ganti `environment-name` dan `stack-name` dengan informasi Anda.

```
chmod +x ecs-stack-helper.sh
./ecs-stack-helper.bash environment-name stack-name
```

Jika berhasil, Anda akan merujuk ke output berikut yang menunjukkan ID CloudFormation tumpukan baru Anda.

```
{
  "StackId": "arn:aws:cloudformation:us-west-2:123456789012:stack/my-ecs-
stack/123456e7-8ab9-01cd-b2fb-36cce63786c9"
}
```

Setelah CloudFormation tumpukan Anda selesai dan AWS telah menyediakan sumber daya Amazon ECS Anda, Anda siap untuk membuat dan mengunggah DAG Anda.

Contoh kode

1. Buka prompt perintah, dan arahkan ke direktori tempat kode DAG Anda disimpan. Contoh:

```
cd dags
```

2. Salin konten contoh kode berikut dan simpan secara lokal sebagai`mwaa-ecs-operator.py`, lalu unggah DAG baru Anda ke Amazon S3.

```
from http import client
from airflow import DAG
```

```
from airflow.providers.amazon.aws.operators.ecs import ECSOperator
from airflow.utils.dates import days_ago
import boto3

CLUSTER_NAME="mwaa-ecs-test-cluster" #Replace value for CLUSTER_NAME with your
 information.
CONTAINER_NAME="mwaa-ecs-test-container" #Replace value for CONTAINER_NAME with
 your information.
LAUNCH_TYPE="FARGATE"

with DAG(
    dag_id = "ecs_fargate_dag",
    schedule_interval=None,
    catchup=False,
    start_date=days_ago(1)
) as dag:
    client=boto3.client('ecs')
    services=client.list_services(cluster=CLUSTER_NAME,launchType=LAUNCH_TYPE)

 service=client.describe_services(cluster=CLUSTER_NAME,services=services['serviceArns'])

    ecs_operator_task = ECSOperator(
        task_id = "ecs_operator_task",
        dag=dag,
        cluster=CLUSTER_NAME,
        task_definition=service['services'][0]['taskDefinition'],
        launch_type=LAUNCH_TYPE,
        overrides={
            "containerOverrides":[
                {
                    "name":CONTAINER_NAME,
                    "command":["ls", "-l", "/"],
                },
            ],
        },

        network_configuration=service['services'][0]['networkConfiguration'],
        awslogs_group="mwaa-ecs-zero",
        awslogs_stream_prefix=f"ecs/{CONTAINER_NAME}",
    )
```

> **Note**
>
> Dalam contoh DAG, for`awslogs_group`, Anda mungkin perlu memodifikasi grup log dengan nama untuk grup log tugas Amazon ECS Anda. Contoh mengasumsikan grup log bernama`mwaa-ecs-zero`. Untuk`awslogs_stream_prefix`, gunakan awalan aliran log tugas Amazon ECS. Contoh mengasumsikan awalan aliran log,. `ecs`

3. Jalankan AWS CLI perintah berikut untuk menyalin DAG ke bucket lingkungan Anda, lalu picu DAG menggunakan Apache Airflow UI.

```
aws s3 cp your-dag.py s3://your-environment-bucket/dags/
```

4. Jika berhasil, Anda akan mendapatkan output yang mirip dengan yang berikut di log tugas untuk `ecs_operator_task` di `ecs_fargate_dag` DAG:

```
[2022-01-01, 12:00:00 UTC] {{ecs.py:300}} INFO - Running ECS Task -
Task definition: arn:aws:ecs:us-west-2:123456789012:task-definition/mwaa-ecs-test-
task:1 - on cluster mwaa-ecs-test-cluster
[2022-01-01, 12:00:00 UTC] {{ecs-operator-test.py:302}} INFO - ECSOperator
 overrides:
{'containerOverrides': [{'name': 'mwaa-ecs-test-container', 'command': ['ls', '-l',
 '/']}]}
.
.
.
[2022-01-01, 12:00:00 UTC] {{ecs.py:379}} INFO - ECS task ID is:
 e012340b5e1b43c6a757cf012c635935
[2022-01-01, 12:00:00 UTC] {{ecs.py:313}} INFO - Starting ECS Task Log Fetcher
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] total
 52
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC]
 lrwxrwxrwx   1 root root    7 Jun 13 18:51 bin -> usr/bin
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] dr-xr-
xr-x   2 root root 4096 Apr  9  2019 boot
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   5 root root  340 Jul 19 17:54 dev
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   1 root root 4096 Jul 19 17:54 etc
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   2 root root 4096 Apr  9  2019 home
```

```
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC]
 lrwxrwxrwx   1 root root    7 Jun 13 18:51 lib -> usr/lib
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC]
 lrwxrwxrwx   1 root root    9 Jun 13 18:51 lib64 -> usr/lib64
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   2 root root 4096 Jun 13 18:51 local
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   2 root root 4096 Apr  9  2019 media
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   2 root root 4096 Apr  9  2019 mnt
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   2 root root 4096 Apr  9  2019 opt
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] dr-xr-
xr-x 103 root root    0 Jul 19 17:54 proc
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] dr-xr-
x-\-\-   2 root root 4096 Apr  9  2019 root
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   2 root root 4096 Jun 13 18:52 run
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC]
 lrwxrwxrwx   1 root root    8 Jun 13 18:51 sbin -> usr/sbin
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x   2 root root 4096 Apr  9  2019 srv
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] dr-xr-
xr-x  13 root root    0 Jul 19 17:54 sys
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC]
 drwxrwxrwt   2 root root 4096 Jun 13 18:51 tmp
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x  13 root root 4096 Jun 13 18:51 usr
[2022-01-01, 12:00:00 UTC] {{ecs.py:119}} INFO - [2022-07-19, 17:54:03 UTC] drwxr-
xr-x  18 root root 4096 Jun 13 18:52 var
.
.
.
[2022-01-01, 12:00:00 UTC] {{ecs.py:328}} INFO - ECS Task has been successfully
 executed
```

Menggunakan dbt dengan Amazon MWAA

Topik ini menunjukkan bagaimana Anda dapat menggunakan dbt dan Postgres dengan Amazon MWAA. Pada langkah-langkah berikut, Anda akan menambahkan dependensi yang diperlukan ke perangkat Anda`requirements.txt`, dan mengunggah contoh proyek dbt ke bucket Amazon S3

lingkungan Anda. Kemudian, Anda akan menggunakan sampel DAG untuk memverifikasi bahwa Amazon MWAA telah menginstal dependensi, dan akhirnya menggunakan `BashOperator` untuk menjalankan proyek dbt.

Topik

- Versi
- Prasyarat
- Dependensi
- Unggah proyek dbt ke Amazon S3
- Gunakan DAG untuk memverifikasi instalasi ketergantungan dbt
- Gunakan DAG untuk menjalankan proyek dbt

Versi

Anda dapat menggunakan contoh kode pada halaman ini dengan Apache Airflow v2 di Python 3.10 dan Apache Airflowv3 di Python 3.11.

Prasyarat

Sebelum Anda dapat menyelesaikan langkah-langkah berikut, Anda memerlukan yang berikut:

- Lingkungan Amazon MWAA menggunakan Apache Airflow v2.2.2. Sampel ini ditulis, dan diuji dengan v2.2.2. Anda mungkin perlu memodifikasi sampel untuk digunakan dengan versi Apache Airflow lainnya.
- Contoh proyek dbt. Untuk mulai menggunakan dbt dengan Amazon MWAA, Anda dapat membuat fork dan mengkloning proyek starter dbt dari repositori dbt-labs. GitHub

Dependensi

Untuk menggunakan Amazon MWAA dengan dbt, tambahkan skrip startup berikut ke lingkungan Anda. Untuk mempelajari lebih lanjut, lihat Menggunakan skrip startup dengan Amazon MWAA.

```
#!/bin/bash

  if [[ "${MWAA_AIRFLOW_COMPONENT}" != "worker" ]]
    then
      exit 0
```

```
fi

echo "------------------------------"
echo "Installing virtual Python env"
echo "------------------------------"

pip3 install --upgrade pip

echo "Current Python version:"
python3 --version
echo "..."

sudo pip3 install --user virtualenv
sudo mkdir python3-virtualenv
cd python3-virtualenv
sudo python3 -m venv dbt-env
sudo chmod -R 777 *

echo "------------------------------"
echo "Activating venv in"
$DBT_ENV_PATH
   echo "------------------------------"

source dbt-env/bin/activate
pip3 list

echo "------------------------------"
echo "Installing libraries..."
echo "------------------------------"

# do not use sudo, as it will install outside the venv
pip3 install dbt-redshift==1.6.1 dbt-postgres==1.6.1

echo "------------------------------"
echo "Venv libraries..."
echo "------------------------------"

pip3 list
dbt --version

echo "------------------------------"
echo "Deactivating venv..."
echo "------------------------------"
```

```
  deactivate
```

Di bagian berikut, Anda akan mengunggah direktori proyek dbt Anda ke Amazon S3 dan menjalankan DAG yang memvalidasi apakah Amazon MWAA telah berhasil menginstal dependensi dbt yang diperlukan.

Unggah proyek dbt ke Amazon S3

Untuk dapat menggunakan proyek dbt dengan lingkungan Amazon MWAA Anda, Anda dapat mengunggah seluruh direktori proyek ke folder lingkungan Anda. `dags` Saat lingkungan diperbarui, Amazon MWAA mengunduh direktori dbt ke folder lokal. `usr/local/airflow/dags/`

Untuk mengunggah proyek dbt ke Amazon S3

1. Arahkan ke direktori tempat Anda mengkloning proyek starter dbt.
2. Jalankan AWS CLI perintah Amazon S3 berikut untuk menyalin konten proyek secara rekursif ke `dags` folder lingkungan Anda menggunakan parameter. `--recursive` Perintah membuat sub-direktori bernama `dbt` yang dapat Anda gunakan untuk semua proyek dbt Anda. Jika sub-direktori sudah ada, file proyek disalin ke direktori yang ada, dan direktori baru tidak dibuat. Perintah ini juga membuat sub-direktori dalam `dbt` direktori untuk proyek pemula khusus ini.

   ```
   aws s3 cp dbt-starter-project s3://amzn-s3-demo-bucket/dags/dbt/dbt-starter-project
    --recursive
   ```

   Anda dapat menggunakan nama yang berbeda untuk sub-direktori proyek untuk mengatur beberapa proyek dbt dalam direktori induk. `dbt`

Gunakan DAG untuk memverifikasi instalasi ketergantungan dbt

DAG berikut menggunakan perintah `BashOperator` dan bash untuk memverifikasi apakah Amazon MWAA telah berhasil menginstal dependensi dbt yang ditentukan dalam. `requirements.txt`

```
from airflow import DAG
   from airflow.operators.bash_operator import BashOperator
   from airflow.utils.dates import days_ago

   with DAG(dag_id="dbt-installation-test", schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
   cli_command = BashOperator(
```

```
   task_id="bash_command",
   bash_command="/usr/local/airflow/python3-virtualenv/dbt-env/bin/dbt --version"
   )
```

Lakukan hal berikut untuk mengakses log tugas dan memverifikasi bahwa dbt dan dependensinya telah diinstal.

1. Arahkan ke konsol Amazon MWAA, lalu pilih Open Airflow UI dari daftar lingkungan yang tersedia.
2. Pada Apache Airflow UI, temukan `dbt-installation-test` DAG dari daftar, lalu pilih tanggal di kolom untuk membuka `Last` Run tugas terakhir yang berhasil.
3. Menggunakan Tampilan Grafik, pilih `bash_command` tugas untuk membuka detail instance tugas.
4. Pilih Log untuk membuka log tugas, lalu verifikasi bahwa log berhasil mencantumkan versi dbt yang kami tentukan. `requirements.txt`

Gunakan DAG untuk menjalankan proyek dbt

DAG berikut menggunakan a `BashOperator` untuk menyalin proyek dbt yang Anda unggah ke Amazon S3 dari direktori `usr/local/airflow/dags/` lokal ke direktori `/tmp` yang dapat diakses tulis, lalu jalankan proyek dbt. Perintah bash mengasumsikan proyek dbt starter berjudul. `dbt-starter-project` Ubah nama direktori sesuai dengan nama direktori proyek Anda.

```
from airflow import DAG
   from airflow.operators.bash_operator import BashOperator
   from airflow.utils.dates import days_ago

   import os

   DAG_ID = os.path.basename(__file__).replace(".py", "")

   # assumes all files are in a subfolder of DAGs called dbt

   with DAG(dag_id=DAG_ID, schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
   cli_command = BashOperator(
   task_id="bash_command",
   bash_command="source /usr/local/airflow/python3-virtualenv/dbt-env/bin/activate;\
```

```
   cp -R /usr/local/airflow/dags/dbt /tmp;\
   echo 'listing project files:';\
   ls -R /tmp;\
   cd /tmp/dbt/mwaa_dbt_test_project;\
   /usr/local/airflow/python3-virtualenv/dbt-env/bin/dbt run --project-dir /tmp/dbt/
mwaa_dbt_test_project --profiles-dir ..;\
   cat /tmp/dbt_logs/dbt.log;\
   rm -rf /tmp/dbt/mwaa_dbt_test_project"
   )
```

AWS Blog dan Tutorial

- Bekerja dengan Amazon EKS dan Amazon MWAA untuk Apache Airflow v2.x

Praktik terbaik untuk Alur Kerja Terkelola Amazon untuk Apache Airflow

Panduan ini menjelaskan praktik terbaik yang kami rekomendasikan saat menggunakan Alur Kerja Terkelola Amazon untuk Apache Airflow.

Topik

- Penyetelan kinerja untuk Apache Airflow di Amazon MWAA
- Mengelola dependensi Python di requirements.txt

Penyetelan kinerja untuk Apache Airflow di Amazon MWAA

Topik ini menjelaskan cara menyetel kinerja Alur Kerja Terkelola Amazon untuk lingkungan Apache Airflow menggunakan. Menggunakan opsi konfigurasi Apache Airflow di Amazon MWAA

Daftar Isi

- Menambahkan opsi konfigurasi Apache Airflow
- Penjadwal Apache Airflow
  - Parameter
  - Batas
- Folder DAG
  - Parameter
- File DAG
  - Parameter
- Tugas
  - Parameter

Menambahkan opsi konfigurasi Apache Airflow

Gunakan prosedur berikut untuk menambahkan opsi konfigurasi Aliran Udara ke lingkungan Anda.

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.

3. Pilih Edit.
4. Pilih Berikutnya.
5. Pilih Tambahkan konfigurasi khusus di panel Opsi konfigurasi Aliran Udara.
6. Pilih konfigurasi dari daftar dropdown dan masukkan nilai, atau masukkan konfigurasi khusus dan masukkan nilai.
7. Pilih Tambahkan konfigurasi khusus untuk setiap konfigurasi yang ingin Anda tambahkan.
8. Pilih Simpan.

Untuk mempelajari lebih lanjut, lihatMenggunakan opsi konfigurasi Apache Airflow di Amazon MWAA.

Penjadwal Apache Airflow

Penjadwal Apache Airflow adalah komponen inti dari Apache Airflow. Masalah dengan penjadwal dapat mencegah DAG diurai dan tugas dijadwalkan. Untuk informasi selengkapnya tentang penyetelan penjadwal Apache Airflow, lihat kinerja penjadwal Anda di situs Fine-tuning web dokumentasi Apache Airflow.

Parameter

Bagian ini menjelaskan opsi konfigurasi yang tersedia untuk penjadwal Apache Airflow (Apache Airflow v2 dan yang lebih baru) dan kasus penggunaannya.

Apache Airflow v3

Konfigurasi	Kasus penggunaan
celery.sync_parallelism Jumlah proses yang digunakan Celery Executor untuk menyinkronkan status tugas. Default: 1	Anda dapat menggunakan opsi ini untuk mencegah konflik antrian dengan membatasi proses yang digunakan Celery Executor. Secara default, nilai diatur 1 untuk mencegah kesalahan dalam mengirimkan log tugas ke CloudWatch Log. Menyetel nilai ke `0` berarti menggunakan jumlah maksimum proses, tetapi dapat menyebabkan kesalahan saat mengirimkan log tugas.

Konfigurasi	Kasus penggunaan
scheduler.scheduler_idle_sleep_time Jumlah detik untuk menunggu antara pemrosesan file DAG berturut-turut di “loop” scheduler. Default: 1	Anda dapat menggunakan opsi ini untuk membebaskan penggunaan CPU pada penjadwal dengan meningkatkan waktu penjadwal tidur setelah selesai mengambil hasil penguraian DAG, menemukan dan mengantri tugas, dan menjalankan tugas antrian di Pelaksana. Meningkatkan nilai ini menghabiskan jumlah thread scheduler yang dijalankan di lingkungan `dag_processor.parsing_processes` untuk Apache Airflow v2 dan Apache Airflow v3. Ini dapat mengurangi kapasitas penjadwal untuk mengurai DAG, dan meningkatkan waktu yang dibutuhkan DAG untuk mengisi di server web.
scheduler.max_dagruns_to_create_per_loop Jumlah maksimum DAG yang akan dibuat DagRunsuntuk per penjadwal “loop.” Default: 10	Anda dapat menggunakan opsi ini untuk membebaskan sumber daya untuk menjadwalkan tugas dengan mengurangi jumlah maksimum DagRunsuntuk “loop” penjadwal.
dag_processor.parsing_processes Jumlah thread penjadwal dapat berjalan secara paralel untuk menjadwalkan DAG. Default: Gunakan `(2 * number of vCPUs) - 1`	Anda dapat menggunakan opsi ini untuk membebaskan sumber daya dengan mengurangi jumlah proses yang dijalankan penjadwal secara paralel untuk mengurai DAG. Kami menyarankan agar angka ini tetap rendah jika penguraian DAG memengaruhi penjadwalan tugas. Anda harus menentukan nilai yang kurang dari jumlah vCPU di lingkungan Anda. Untuk mempelajari lebih lanjut, lihat Batas.

Apache Airflow v2

Konfigurasi	Kasus penggunaan
celery.sync_parallelism Jumlah proses yang digunakan Celery Executor untuk menyinkronkan status tugas. Default: 1	Anda dapat menggunakan opsi ini untuk mencegah konflik antrian dengan membatasi proses yang digunakan Celery Executor. Secara default, nilai diatur 1 untuk mencegah kesalahan dalam mengirimkan log tugas ke CloudWatch Log. Menyetel nilai ke `0` berarti menggunakan jumlah maksimum proses, tetapi dapat menyebabkan kesalahan saat mengirimkan log tugas.
scheduler.idle_sleep_time Jumlah detik untuk menunggu antara pemrosesan file DAG berturut-turut di “loop” scheduler. Default: 1	Anda dapat menggunakan opsi ini untuk membebaskan penggunaan CPU pada penjadwal dengan meningkatkan waktu penjadwal tidur setelah selesai mengambil hasil penguraian DAG, menemukan dan mengantri tugas, dan menjalankan tugas antrian di Pelaksana. Meningkatkan nilai ini menghabiskan jumlah thread scheduler yang dijalankan di lingkungan `scheduler .parsing_processes` untuk Apache Airflow v2 dan Apache Airflow v3. Ini dapat mengurangi kapasitas penjadwal untuk mengurai DAG, dan meningkatkan waktu yang dibutuhkan DAG untuk mengisi di server web.
scheduler.max_dagruns_to_create_per_loop Jumlah maksimum DAG yang akan dibuat DagRunsuntuk per penjadwal “loop.” Default: 10	Anda dapat menggunakan opsi ini untuk membebaskan sumber daya untuk menjadwal kan tugas dengan mengurangi jumlah maksimum DagRunsuntuk “loop” penjadwal.

Konfigurasi	Kasus penggunaan
scheduler.parsing_processes Jumlah thread penjadwal dapat berjalan secara paralel untuk menjadwalkan DAG. Default: Gunakan `(2 * number of vCPUs) - 1`	Anda dapat menggunakan opsi ini untuk membebaskan sumber daya dengan mengurangi jumlah proses yang dijalanka n penjadwal secara paralel untuk mengurai DAG. Kami menyarankan agar angka ini tetap rendah jika penguraian DAG memengaruhi penjadwalan tugas. Anda harus menentuka n nilai yang kurang dari jumlah vCPU di lingkungan Anda. Untuk mempelajari lebih lanjut, lihat Batas.

Batas

Bagian ini menjelaskan batasan yang perlu dipertimbangkan saat menyesuaikan parameter default untuk penjadwal.

scheduler.parsing_processes, scheduler.max_threads (hanya v2)

Dua utas diperbolehkan per vCPU untuk kelas lingkungan. Setidaknya satu thread harus disediakan untuk scheduler untuk kelas lingkungan. Jika Anda melihat keterlambatan dalam tugas yang dijadwalkan, Anda mungkin perlu meningkatkan kelas lingkungan Anda. Misalnya, lingkungan yang besar memiliki instance kontainer Fargate 4 vCPU untuk penjadwalnya. Ini berarti bahwa maksimum 7 total utas tersedia untuk digunakan untuk proses lain. Artinya, dua utas dikalikan empat vCPU, minus satu untuk penjadwal itu sendiri. Nilai yang Anda tentukan di `scheduler.max_threads` (hanya v2) dan tidak `scheduler.parsing_processes` boleh melebihi jumlah utas yang tersedia untuk kelas lingkungan, seperti yang tercantum:

- mw1.small — Tidak boleh melebihi 1 thread untuk proses lainnya. Utas yang tersisa dicadangkan untuk penjadwal.
- mw1.medium — Tidak boleh melebihi 3 thread untuk proses lainnya. Utas yang tersisa dicadangkan untuk penjadwal.
- mw1.large - Tidak boleh melebihi 7 utas untuk proses lainnya. Utas yang tersisa dicadangkan untuk penjadwal.

Folder DAG

Penjadwal Apache Airflow terus memindai folder DAG di lingkungan Anda. `plugins.zip`File apa pun yang berisi, atau file Python (`.py`) yang berisi pernyataan impor "aliran udara". Setiap objek Python DAG yang dihasilkan kemudian ditempatkan ke dalam file yang akan diproses oleh penjadwal untuk menentukan apa, jika ada, tugas yang perlu dijadwalkan. DagBag Penguraian file Dag terjadi terlepas dari apakah file berisi objek DAG yang layak.

Parameter

Bagian ini menjelaskan opsi konfigurasi yang tersedia untuk folder DAGS (Apache Airflow v2 dan yang lebih baru) dan kasus penggunaannya.

Apache Airflow v3

Konfigurasi	Kasus penggunaan
dag_processor.refresh_interval Jumlah detik folder DAG harus dipindai untuk file baru. Default: 300 detik	Anda dapat menggunakan opsi ini untuk membebaskan sumber daya dengan meningkatkan jumlah detik untuk mengurai folder DAG. Kami merekomendasikan untuk meningkatkan nilai ini jika Anda mengalami waktu penguraian yang lama`total_parse_time metrics` , yang mungkin disebabkan oleh sejumlah besar file di folder DAGS Anda.
dag_processor.min_file_process_interval Jumlah detik setelah penjadwal mem-parsing DAG dan pembaruan ke DAG tercermin. Default: 30 detik	Anda dapat menggunakan opsi ini untuk membebaskan sumber daya dengan meningkatkan jumlah detik yang menunggu penjadwal sebelum menguraikan DAG. Misalnya, jika Anda menentukan nilai`30`, file DAG diurai setelah setiap 30 detik. Kami menyarankan agar angka ini tetap tinggi untuk mengurangi penggunaan CPU di lingkungan Anda.

Apache Airflow v2

Konfigurasi	Kasus penggunaan
scheduler.dag_dir_list_interval Jumlah detik folder DAG harus dipindai untuk file baru. Default: 300 detik	Anda dapat menggunakan opsi ini untuk membebaskan sumber daya dengan meningkatkan jumlah detik untuk mengurai folder DAG. Kami merekomendasikan untuk meningkatkan nilai ini jika Anda mengalami waktu penguraian yang lama`total_parse_time metrics` , yang mungkin disebabkan oleh sejumlah besar file di folder DAGS Anda.
scheduler.min_file_process_interval Jumlah detik setelah penjadwal mem-parsing DAG dan pembaruan ke DAG tercermin. Default: 30 detik	Anda dapat menggunakan opsi ini untuk membebaskan sumber daya dengan meningkatkan jumlah detik yang menunggu penjadwal sebelum menguraikan DAG. Misalnya, jika Anda menentukan nilai`30`, file DAG diurai setelah setiap 30 detik. Kami menyarankan agar angka ini tetap tinggi untuk mengurangi penggunaan CPU di lingkungan Anda.

File DAG

Sebagai bagian dari loop scheduler Apache Airflow, file DAG individual diurai untuk mengekstrak objek DAG Python. Di Apache Airflow v2 dan yang lebih baru, scheduler mem-parsing maksimum jumlah proses parsing secara bersamaan. Jumlah detik yang ditentukan dalam `scheduler.min_file_process_interval` (v2) atau `dag_processor.min_file_process_interval` (v3) harus lewat sebelum file yang sama diurai lagi.

Parameter

Bagian ini menjelaskan opsi konfigurasi yang tersedia untuk file Apache Airflow DAG (Apache Airflow v2 dan yang lebih baru) dan kasus penggunaannya.

Apache Airflow v3

Konfigurasi	Kasus penggunaan
dag_processor.dag_file_processor_timeout Jumlah detik sebelum DagFilePr ocessorwaktu habis memproses file DAG. Default: 50 detik	Anda dapat menggunakan opsi ini untuk membebaskan sumber daya dengan meningkatkan waktu yang dibutuhkan sebelum DagFileProcessorwaktu habis. Kami merekomendasikan untuk meningkatkan nilai ini jika Anda mengalami batas waktu dalam log pemrosesan DAG Anda yang menghasil kan tidak ada DAG yang layak dimuat.
core.dagbag_import_timeout Jumlah detik sebelum mengimpor file Python habis waktu. Default: 30 detik	Anda dapat menggunakan opsi ini untuk membebaskan sumber daya dengan meningkatkan waktu yang diperlukan sebelum waktu penjadwal habis saat mengimpor file Python untuk mengekstr ak objek DAG. Opsi ini diproses sebagai bagian dari penjadwal “loop,” dan harus berisi nilai kurang dari nilai yang ditentukan dalam`dag_processor.dag_file_proc essor_timeout` .
core.min_serialized_dag_update_interval Jumlah minimum detik setelah DAG serial dalam database diperbarui. Default: 30	Anda dapat menggunakan opsi ini untuk membebaskan sumber daya dengan meningkatkan jumlah detik setelah DAG serial dalam database diperbarui. Kami merekomen dasikan untuk meningkatkan nilai ini jika Anda memiliki sejumlah besar DAG, atau DAG kompleks. Meningkatkan nilai ini mengurang

Konfigurasi	Kasus penggunaan
	i beban pada penjadwal dan database saat DAG diserialisasikan.
core.min_serialized_dag_fetch_interval Jumlah detik DAG serial diambil kembali dari database saat sudah dimuat di file. DagBag Default: 10	Anda dapat menggunakan opsi ini untuk membebaskan sumber daya dengan meningkatkan jumlah detik DAG serial diambil kembali. Nilai harus lebih besar dari nilai yang ditentukan `core.min_serialized_dag_update_interval` untuk mengurangi tingkat "tulis" database. Meningkatkan nilai ini mengurangi beban pada server web dan database saat DAG diserialisasikan.

Apache Airflow v2

Konfigurasi	Kasus penggunaan
core.dag_file_processor_timeout Jumlah detik sebelum DagFilePr ocessorwaktu habis memproses file DAG. Default: 50 detik	Anda dapat menggunakan opsi ini untuk membebaskan sumber daya dengan meningkatkan waktu yang dibutuhkan sebelum DagFileProcessorwaktu habis. Kami merekomendasikan untuk meningkatkan nilai ini jika Anda mengalami batas waktu dalam log pemrosesan DAG Anda yang menghasil kan tidak ada DAG yang layak dimuat.
core.dagbag_import_timeout Jumlah detik sebelum mengimpor file Python habis waktu. Default: 30 detik	Anda dapat menggunakan opsi ini untuk membebaskan sumber daya dengan meningkatkan waktu yang diperlukan sebelum waktu penjadwal habis saat mengimpor file Python untuk mengekstr ak objek DAG. Opsi ini diproses sebagai

Konfigurasi	Kasus penggunaan
	bagian dari penjadwal “loop,” dan harus berisi nilai kurang dari nilai yang ditentukan dalam`core.dag_file_processor_timeout` .
core.min_serialized_dag_update_interval Jumlah minimum detik setelah DAG serial dalam database diperbarui. Default: 30	Anda dapat menggunakan opsi ini untuk membebaskan sumber daya dengan meningkatkan jumlah detik setelah DAG serial dalam database diperbarui. Kami merekomendasikan untuk meningkatkan nilai ini jika Anda memiliki sejumlah besar DAG, atau DAG kompleks. Meningkatkan nilai ini mengurangi beban pada penjadwal dan database saat DAG diserialisasikan.
core.min_serialized_dag_fetch_interval Jumlah detik DAG serial diambil kembali dari database saat sudah dimuat di file. DagBag Default: 10	Anda dapat menggunakan opsi ini untuk membebaskan sumber daya dengan meningkatkan jumlah detik DAG serial diambil kembali. Nilai harus lebih besar dari nilai yang ditentukan `core.min_serialized_dag_update_interval` untuk mengurangi tingkat “tulis” database. Meningkatkan nilai ini mengurangi beban pada server web dan database saat DAG diserialisasikan.

Tugas

Penjadwal Apache Airflow dan pekerja keduanya terlibat dalam tugas antrian dan de-antrian. Penjadwal mengambil tugas yang diuraikan yang siap dijadwalkan dari status Tidak Ada ke status Terjadwal. Pelaksana, juga berjalan pada wadah penjadwal di Fargate, mengantri tugas-tugas itu dan menetapkan statusnya ke Antrian. Ketika pekerja memiliki kapasitas, ia mengambil tugas dari

antrian dan menetapkan status ke Running, yang kemudian mengubah statusnya menjadi Sukses atau Gagal berdasarkan apakah tugas berhasil atau gagal.

Parameter

Bagian ini menjelaskan opsi konfigurasi yang tersedia untuk tugas Apache Airflow dan kasus penggunaannya.

Opsi konfigurasi default yang diganti Amazon MWAA ditandai. *red*

Apache Airflow v3

Konfigurasi	Kasus penggunaan
inti.paralelisme Jumlah maksimum instance tugas yang dapat memiliki `Running` status. Default: Dinamis diatur berdasark an`(maxWorkers * maxCelery Workers) / schedulers * 1.5` .	Anda dapat menggunakan opsi ini untuk membebaskan sumber daya dengan meningkatkan jumlah instance tugas yang dapat dijalankan secara bersamaan. Nilai yang ditentukan harus berupa jumlah pekerja yang tersedia dikalikan dengan kepadatan tugas pekerja. Kami merekomendasikan untuk mengubah nilai ini hanya ketika Anda mengalami sejumlah besar tugas yang terjebak dalam status “Berlari” atau “Antrian”.
core.execute_tasks_new_python_penerjemah Menentukan apakah Apache Airflow menjalankan tugas dengan melakukan fork pada proses induk, atau dengan membuat proses Python baru. Default: `True`	Ketika diatur ke`True`, Apache Airflow mengenali perubahan yang Anda buat pada plugin Anda sebagai proses Python baru sehingga dibuat untuk menjalankan tugas.
celery.worker_concurrency Amazon MWAA mengesampingkan instalasi basis Airflow untuk opsi ini untuk menskalak	*Any value specified for this option is ignored.*

Konfigurasi	Kasus penggunaan
an pekerja sebagai bagian dari komponen penskalaan otomatisnya. Default: Tidak berlaku	
seledry.worker_autoscale Konkurensi tugas untuk pekerja. Default: • mw1.mikro - 3,0 • mw1.small - 5,0 • mw1.medium - 10,0 • mw1.large - 20,0 • mw1.xlarge - 40,0 • mw1.2xlarge - 80,0	Anda dapat menggunakan opsi ini untuk membebaskan sumber daya dengan mengurangi`maximum`, konkurensi `minimum` tugas pekerja. Pekerja menerima hingga tugas `maximum` bersamaan yang dikonfigu rasi, terlepas dari apakah ada sumber daya yang cukup untuk melakukannya. Jika tugas dijadwalkan tanpa sumber daya yang memadai, tugas segera gagal. Kami merekomendasikan mengubah nilai ini untuk tugas intensif sumber daya dengan mengurangi nilai menjadi kurang dari default untuk memungkinkan lebih banyak kapasitas per tugas.

Apache Airflow v2

Konfigurasi	Kasus penggunaan
inti.paralelisme Jumlah maksimum instance tugas yang dapat memiliki `Running` status. Default: Dinamis diatur berdasark an`(maxWorkers * maxCelery Workers) / schedulers * 1.5` .	Anda dapat menggunakan opsi ini untuk membebaskan sumber daya dengan meningkatkan jumlah instance tugas yang dapat dijalankan secara bersamaan. Nilai yang ditentukan harus berupa jumlah pekerja yang tersedia dikalikan dengan kepadatan tugas pekerja. Kami merekomendasikan untuk mengubah nilai ini hanya ketika Anda mengalami sejumlah besar tugas yang terjebak dalam status “Berlari” atau “Antrian”.

Konfigurasi	Kasus penggunaan
core.dag_concurrency Jumlah instance tugas yang diizinkan untuk dijalankan secara bersamaan untuk setiap DAG. Default: 10000	Anda dapat menggunakan opsi ini untuk membebaskan sumber daya dengan meningkatkan jumlah instance tugas yang diizinkan untuk dijalankan secara bersamaan . Misalnya, jika Anda memiliki seratus DAG dengan sepuluh tugas paralel, dan Anda ingin semua DAG berjalan secara bersamaan , Anda dapat menghitung paralelisme maksimum sebagai jumlah pekerja yang tersedia dikalikan dengan kepadatan tugas pekerja`celery.worker_concurrency` , dibagi dengan jumlah DAG.
core.execute_tasks_new_python_penerjemah Menentukan apakah Apache Airflow menjalankan tugas dengan melakukan fork pada proses induk, atau dengan membuat proses Python baru. Default: `True`	Ketika diatur ke`True`, Apache Airflow mengenali perubahan yang Anda buat pada plugin Anda sebagai proses Python baru sehingga dibuat untuk menjalankan tugas.
celery.worker_concurrency Amazon MWAA mengesampingkan instalasi basis Airflow untuk opsi ini untuk menskalak an pekerja sebagai bagian dari komponen penskalaan otomatisnya. Default: Tidak berlaku	*Any value specified for this option is ignored.*

Konfigurasi	Kasus penggunaan
seledry.worker_autoscale Konkurensi tugas untuk pekerja. Default: • mw1.mikro - 3,0 • mw1.small - 5,0 • mw1.medium - 10,0 • mw1.large - 20,0 • mw1.xlarge - 40,0 • mw1.2xlarge - 80,0	Anda dapat menggunakan opsi ini untuk membebaskan sumber daya dengan mengurangi`maximum`, konkurensi `minimum` tugas pekerja. Pekerja menerima hingga tugas `maximum` bersamaan yang dikonfigu rasi, terlepas dari apakah ada sumber daya yang cukup untuk melakukannya. Jika tugas dijadwalkan tanpa sumber daya yang memadai, tugas segera gagal. Kami merekomendasikan mengubah nilai ini untuk tugas intensif sumber daya dengan mengurangi nilai menjadi kurang dari default untuk memungkinkan lebih banyak kapasitas per tugas.

Mengelola dependensi Python di requirements.txt

Topik ini menjelaskan cara menginstal dan mengelola dependensi Python dalam file untuk lingkungan Alur Kerja Terkelola Amazon `requirements.txt` untuk Apache Airflow.

Daftar Isi

- Menguji DAG menggunakan utilitas Amazon MWAA CLI
- Menginstal dependensi Python menggunakan Format File Persyaratan PyPi.org
  - Opsi satu: dependensi Python dari Indeks Paket Python
  - Opsi dua: Roda Python (.whl)
    - Menggunakan file plugins.zip di bucket Amazon S3
    - Menggunakan file WHL yang dihosting di URL
    - Membuat file WHL dari DAG
  - Opsi tiga: Dependensi Python yang dihosting di Compliant Repo pribadi PyPi/PEP-503
- Mengaktifkan log di konsol Amazon MWAA
- Mengakses log di konsol CloudWatch Log

- Mengakses kesalahan di Apache Airflow UI
  - Masuk ke Apache Airflow
- Contoh skenario requirements.txt

Menguji DAG menggunakan utilitas Amazon MWAA CLI

- Utilitas antarmuka baris perintah (CLI) mereplikasi Alur Kerja Terkelola Amazon untuk lingkungan Apache Airflow secara lokal.
- CLI membangun image container Docker secara lokal yang mirip dengan image produksi Amazon MWAA. Anda dapat menggunakan ini untuk menjalankan lingkungan Apache Airflow lokal untuk mengembangkan dan menguji DAG, plugin khusus, dan dependensi sebelum menerapkan ke Amazon MWAA.
- Untuk menjalankan CLI, lihat aws-mwaa-docker-images on. GitHub

Menginstal dependensi Python menggunakan Format File Persyaratan PyPi.org

Bagian berikut menjelaskan berbagai cara untuk menginstal dependensi Python sesuai dengan Format File Persyaratan. PyPi.org

Opsi satu: dependensi Python dari Indeks Paket Python

Bagian berikut menjelaskan cara menentukan dependensi Python dari Indeks Paket Python dalam sebuah file. `requirements.txt`

Apache Airflow v3

1. Uji secara lokal. Tambahkan pustaka tambahan secara iteratif untuk menemukan kombinasi paket dan versinya yang tepat, sebelum membuat `requirements.txt` file. Untuk menjalankan utilitas CLI Amazon MWAA, lihat aws-mwaa-docker-images di. GitHub
2. Tinjau paket tambahan Apache Airflow. Untuk mengakses daftar paket yang diinstal untuk Apache Airflow v3 di Amazon MWAA, lihat aws-mwaa-docker-images di situs web. `requirements.txt` GitHub
3. Tambahkan pernyataan kendala. Tambahkan file kendala untuk lingkungan Apache Airflow v3 Anda di bagian atas file Anda. `requirements.txt` File batasan Apache Airflow menentukan versi penyedia yang tersedia pada saat rilis Apache Airflow.

Dalam contoh berikut, ganti *{environment-version}* dengan nomor versi lingkungan Anda, dan *{Python-version}* dengan versi Python yang kompatibel dengan lingkungan Anda.

Untuk informasi tentang versi Python yang kompatibel dengan lingkungan Apache Airflow Anda, lihat Apache Airflow Versions.

```
--constraint "https://raw.githubusercontent.com/apache/airflow/
constraints-{Airflow-version}/constraints-{Python-version}.txt"
```

Jika file kendala menentukan bahwa `xyz==1.0` paket tidak kompatibel dengan paket lain di lingkungan Anda, `pip3 install` gagal mencegah pustaka yang tidak kompatibel diinstal ke lingkungan Anda. Jika instalasi gagal untuk paket apa pun, Anda dapat mengakses log kesalahan untuk setiap komponen Apache Airflow (penjadwal, pekerja, dan server web) di aliran log yang sesuai di Log. CloudWatch Untuk informasi selengkapnya tentang jenis log, lihatthe section called “Mengakses log Aliran Udara”.

4. Paket Apache Airflow. Tambahkan paket ekstra dan versi (==). Ini membantu mencegah paket dengan nama yang sama, tetapi versi yang berbeda, diinstal di lingkungan Anda.

```
apache-airflow[package-extra]==2.5.1
```

5. Pustaka Python. Tambahkan nama paket dan versi (==) di `requirements.txt` file Anda. Ini membantu mencegah pembaruan yang melanggar di masa mendatang PyPi.orgagar tidak diterapkan secara otomatis.

```
library == version
```

Example Boto3 dan psycopg2-biner

Contoh ini disediakan untuk tujuan demonstrasi. Pustaka biner boto dan psycopg2 disertakan dengan instalasi dasar untuk Apache Airflow v3 dan tidak perlu ditentukan dalam file. `requirements.txt`

```
boto3==1.17.54
boto==2.49.0
botocore==1.20.54
psycopg2-binary==2.8.6
```

Jika paket ditentukan tanpa versi, Amazon MWAA menginstal versi terbaru paket dari. PyPi.org Versi ini dapat bertentangan dengan paket lain di Andarequirements.txt.

Apache Airflow v2

1. Uji secara lokal. Tambahkan pustaka tambahan secara iteratif untuk menemukan kombinasi paket dan versinya yang tepat, sebelum membuat `requirements.txt` file. Untuk menjalankan utilitas CLI Amazon MWAA, lihat aws-mwaa-docker-images di. GitHub
2. Tinjau paket tambahan Apache Airflow. Untuk mengakses daftar paket yang diinstal untuk Apache Airflow v2 di Amazon MWAA, akses aws-mwaa-docker-images di situs web. `requirements.txt` GitHub
3. Tambahkan pernyataan kendala. Tambahkan file kendala untuk lingkungan Apache Airflow v2 Anda di bagian atas file Anda. `requirements.txt` File batasan Apache Airflow menentukan versi penyedia yang tersedia pada saat rilis Apache Airflow.

   Dimulai dengan Apache Airflow v2.7.2, file persyaratan Anda harus menyertakan pernyataan. `--constraint` Jika Anda tidak memberikan kendala, Amazon MWAA akan menentukan satu untuk Anda untuk memastikan paket yang tercantum dalam persyaratan Anda kompatibel dengan versi Apache Airflow yang Anda gunakan.

   Dalam contoh berikut, ganti *`{environment-version}`* dengan nomor versi lingkungan Anda, dan *`{Python-version}`* dengan versi Python yang kompatibel dengan lingkungan Anda.

   Untuk informasi tentang versi Python yang kompatibel dengan lingkungan Apache Airflow Anda, lihat Apache Airflow Versions.

   ```
   --constraint "https://raw.githubusercontent.com/apache/airflow/
   constraints-{Airflow-version}/constraints-{Python-version}.txt"
   ```

   Jika file kendala menentukan bahwa `xyz==1.0` paket tidak kompatibel dengan paket lain di lingkungan Anda, `pip3 install` gagal mencegah pustaka yang tidak kompatibel diinstal ke lingkungan Anda. Jika instalasi gagal untuk paket apa pun, Anda dapat mengakses log kesalahan untuk setiap komponen Apache Airflow (penjadwal, pekerja, dan server web) di aliran log yang sesuai di Log. CloudWatch Untuk informasi selengkapnya tentang jenis log, lihatthe section called “Mengakses log Aliran Udara”.

4. Paket Apache Airflow. Tambahkan paket ekstra dan versi (==). Ini membantu mencegah paket dengan nama yang sama, tetapi versi yang berbeda, diinstal di lingkungan Anda.

```
apache-airflow[package-extra]==2.5.1
```

5. Pustaka Python. Tambahkan nama paket dan versi (==) di `requirements.txt` file Anda. Ini membantu mencegah pembaruan yang melanggar di masa mendatang PyPi.orgagar tidak diterapkan secara otomatis.

```
library == version
```

Example Boto3 dan psycopg2-biner

Contoh ini disediakan untuk tujuan demonstrasi. Pustaka biner boto dan psycopg2 disertakan dengan instalasi dasar Apache Airflow v2 dan tidak perlu ditentukan dalam file. `requirements.txt`

```
boto3==1.17.54
boto==2.49.0
botocore==1.20.54
psycopg2-binary==2.8.6
```

Jika paket ditentukan tanpa versi, Amazon MWAA menginstal versi terbaru paket dari. PyPi.org Versi ini dapat bertentangan dengan paket lain di Anda`requirements.txt`.

Opsi dua: Roda Python (.whl)

Roda Python adalah format paket yang dirancang untuk mengirimkan perpustakaan dengan artefak yang dikompilasi. Ada beberapa manfaat paket roda sebagai metode untuk menginstal dependensi di Amazon MWAA:

- Instalasi lebih cepat - file WHL disalin ke wadah sebagai satu ZIP, dan kemudian diinstal secara lokal, tanpa harus mengunduh masing-masing.
- Konflik yang lebih sedikit — Anda dapat menentukan kompatibilitas versi untuk paket Anda terlebih dahulu. Akibatnya, tidak perlu mengerjakan versi yang kompatibel `pip` secara rekursif.
- Ketahanan yang lebih besar — Dengan pustaka yang dihosting secara eksternal, persyaratan hilir dapat berubah, mengakibatkan ketidakcocokan versi antar kontainer di lingkungan Amazon

MWAA. Dengan tidak bergantung pada sumber eksternal untuk dependensi, setiap kontainer memiliki pustaka yang sama terlepas dari kapan setiap wadah dipakai.

Kami merekomendasikan metode berikut untuk menginstal dependensi Python dari arsip roda Python () di file Anda. `.whl` `requirements.txt`

Metode

- Menggunakan file plugins.zip di bucket Amazon S3
- Menggunakan file WHL yang dihosting di URL
- Membuat file WHL dari DAG

Menggunakan file **`plugins.zip`** di bucket Amazon S3

Penjadwal Apache Airflow, pekerja, dan server web (untuk Apache Airflow v2.2.2 dan yang lebih baru) mencari plugin khusus selama startup pada wadah Fargate yang dikelola untuk lingkungan Anda di. AWS`/usr/local/airflow/plugins/*` Proses ini dimulai sebelum Amazon MWAA untuk dependensi `pip3 install -r requirements.txt` Python dan startup layanan Apache Airflow. `plugins.zip`File dapat digunakan untuk file apa pun yang Anda tidak ingin terus berubah selama eksekusi lingkungan, atau Anda tidak ingin memberikan akses ke pengguna yang menulis DAG. Misalnya, file roda pustaka Python, file PEM sertifikat, dan file YAMAL konfigurasi.

Bagian berikut menjelaskan cara memasang roda yang ada di `plugins.zip` file di bucket Amazon S3 Anda.

1. Unduh file WHL yang diperlukan Anda dapat menggunakan `pip download`yang ada `requirements.txt` di Amazon MWAA aws-mwaa-docker-images atau wadah Amazon Linux 2 lainnya untuk menyelesaikan dan mengunduh file roda Python yang diperlukan.

   ```
   pip3 download -r "$AIRFLOW_HOME/dags/requirements.txt" -d "$AIRFLOW_HOME/plugins"
   cd "$AIRFLOW_HOME/plugins"
   zip "$AIRFLOW_HOME/plugins.zip" *
   ```

2. Tentukan jalur di Anda **`requirements.txt`**. Tentukan direktori plugin di bagian atas requirements.txt Anda menggunakan `--find-links`dan instruksikan untuk `pip` tidak menginstal dari sumber lain menggunakan `--no-index`, seperti yang tercantum dalam kode berikut:

   ```
   --find-links /usr/local/airflow/plugins
   ```

```
--no-index
```

Example roda di requirements.txt

Contoh berikut mengasumsikan Anda telah mengunggah roda dalam `plugins.zip` file di root bucket Amazon S3 Anda. Contoh:

```
--find-links /usr/local/airflow/plugins
--no-index

numpy
```

Amazon MWAA mengambil `numpy-1.20.1-cp37-cp37m-manylinux1_x86_64.whl` roda dari `plugins` folder dan menginstalnya di lingkungan Anda.

Menggunakan file WHL yang dihosting di URL

Bagian berikut menjelaskan cara memasang roda yang di-host di URL. URL harus dapat diakses publik, atau dapat diakses dari dalam VPC Amazon khusus yang Anda tentukan untuk lingkungan Amazon MWAA Anda.

- Berikan URL. Berikan URL ke roda di Anda`requirements.txt`.

  Example arsip roda pada URL publik

  Contoh berikut mengunduh roda dari situs publik.

  ```
  --find-links https://files.pythonhosted.org/packages/
  --no-index
  ```

  Amazon MWAA mengambil roda dari URL yang Anda tentukan dan menginstalnya di lingkungan Anda.

  > **Note**
  >
  > URL tidak dapat diakses dari server web pribadi yang menginstal persyaratan di Amazon MWAA v2.2.2 dan yang lebih baru.

Membuat file WHL dari DAG

Jika Anda memiliki lingkungan jaringan Pribadi menggunakan Apache Airflow v2.2.2 atau yang lebih baru dan Anda tidak dapat menginstal persyaratan karena lingkungan Anda tidak memiliki akses ke repositori eksternal, Anda dapat menggunakan DAG berikut untuk mengambil persyaratan MWAA Amazon yang ada dan mengemasnya di Amazon S3:

```
from airflow import DAG
 from airflow.operators.bash_operator import BashOperator
 from airflow.utils.dates import days_ago

 S3_BUCKET = 'my-s3-bucket'
 S3_KEY = 'backup/plugins_whl.zip'

 with DAG(dag_id="create_whl_file", schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
 cli_command = BashOperator(
 task_id="bash_command",
 bash_command=f"mkdir /tmp/whls;pip3 download -r /usr/local/airflow/requirements/
requirements.txt -d /tmp/whls;zip -j /tmp/plugins.zip /tmp/whls/*;aws s3 cp /tmp/
plugins.zip s3://amzn-s3-demo-bucket/{S3_KEY}"
)
```

Setelah menjalankan DAG, gunakan file baru ini sebagai Amazon MWAA Andaplugins.zip, opsional, dikemas dengan plugin lain. Kemudian, perbarui Anda requirements.txt sebelumnya dengan --find-links /usr/local/airflow/plugins dan --no-index tanpa menambahkan. --constraint

Metode ini dapat Anda gunakan untuk menggunakan pustaka yang sama secara offline.

Opsi tiga: Dependensi Python yang dihosting di Compliant Repo pribadi PyPi/PEP-503

Bagian berikut menjelaskan cara menginstal tambahan Apache Airflow yang di-host di URL pribadi dengan otentikasi.

1. Tambahkan nama pengguna dan kata sandi Anda sebagai opsi konfigurasi Apache Airflow. Contoh:

   - foo.user : *YOUR_USER_NAME*
   - foo.pass : *YOUR_PASSWORD*

2. Buat `requirements.txt` file Anda. Gantikan placeholder dalam contoh berikut dengan URL pribadi Anda, dan nama pengguna dan kata sandi yang telah Anda tambahkan sebagai opsi konfigurasi Apache Airflow. Contoh:

```
--index-url https://${AIRFLOW__FOO__USER}:${AIRFLOW__FOO__PASS}@my.privatepypi.com
```

3. Tambahkan pustaka tambahan apa pun ke `requirements.txt` file Anda. Contoh:

```
--index-url https://${AIRFLOW__FOO__USER}:${AIRFLOW__FOO__PASS}@my.privatepypi.com
my-private-package==1.2.3
```

Mengaktifkan log di konsol Amazon MWAA

Peran eksekusi untuk lingkungan Amazon MWAA Anda memerlukan izin untuk mengirim log ke CloudWatch Log. Untuk memperbarui izin peran eksekusi, lihat. Peran eksekusi Amazon MWAA

Anda dapat mengaktifkan log Apache Airflow di `INFO`,`WARNING`,`ERROR`, atau level. `CRITICAL` Saat Anda memilih level log, Amazon MWAA mengirimkan log untuk level tersebut dan semua tingkat keparahan yang lebih tinggi. Misalnya, jika Anda mengaktifkan log di `INFO` level tersebut, Amazon MWAA mengirimkan `INFO` log dan`WARNING`,`ERROR`, dan tingkat `CRITICAL` log ke CloudWatch Log. Sebaiknya aktifkan log Apache Airflow di `INFO` tingkat agar penjadwal mengakses log yang diterima untuk file. `requirements.txt`

Mengakses log di konsol CloudWatch Log

Anda dapat mengakses log Apache Airflow untuk penjadwal yang menjadwalkan alur kerja Anda dan mengurai folder Anda. `dags` Langkah-langkah berikut menjelaskan cara membuka grup log untuk penjadwal di konsol Amazon MWAA, dan mengakses log Apache Airflow di konsol Log. CloudWatch

Untuk mengakses log untuk **`requirements.txt`**

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.
3. Pilih grup log penjadwal aliran udara di panel Pemantauan.
4. Pilih `requirements_install_ip` log masuk Aliran log.

5. Lihat daftar paket yang diinstal pada lingkungan di`/usr/local/airflow/.local/bin`. Contoh:

```
Collecting appdirs==1.4.4 (from -r /usr/local/airflow/.local/bin (line 1))
Downloading https://files.pythonhosted.org/
packages/3b/00/2344469e2084fb28kjdsfiuyweb47389789vxbmnbjhsdgf5463acd6cf5e3db69324/
appdirs-1.4.4-py2.py3-none-any.whl
Collecting astroid==2.4.2 (from -r /usr/local/airflow/.local/bin (line 2))
```

6. Tinjau daftar paket dan apakah salah satu dari ini mengalami kesalahan selama instalasi. Jika terjadi kesalahan, Anda bisa mendapatkan kesalahan yang mirip dengan yang berikut ini:

```
2021-03-05T14:34:42.731-07:00
No matching distribution found for LibraryName==1.0.0 (from -r /usr/local/
airflow/.local/bin (line 4))
No matching distribution found for LibraryName==1.0.0 (from -r /usr/local/
airflow/.local/bin (line 4))
```

Mengakses kesalahan di Apache Airflow UI

Anda juga dapat memeriksa Apache Airflow UI untuk mengidentifikasi apakah kesalahan terkait dengan masalah lain. Kesalahan paling umum yang dapat Anda temui dengan Apache Airflow di Amazon MWAA adalah:

```
Broken DAG: No module named x
```

Jika Anda menemukan kesalahan ini di UI Apache Airflow Anda, Anda mungkin kehilangan ketergantungan yang diperlukan dalam file Anda. `requirements.txt`

Masuk ke Apache Airflow

Anda memerlukan Kebijakan akses Apache Airflow UI: AmazonMWAAWebServerAccess izin untuk Akun AWS in AWS Identity and Access Management (IAM) untuk mengakses UI Apache Airflow Anda.

Untuk mengakses UI Apache Airflow Anda

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.

3. Pilih Buka UI Aliran Udara.

Contoh skenario **`requirements.txt`**

Anda dapat mencampur dan mencocokkan berbagai format dalam format Anda`requirements.txt`. Contoh berikut menggunakan kombinasi berbagai cara untuk menginstal ekstra.

Example Ekstra PyPi.org dan URL publik

Anda perlu menggunakan `--index-url` opsi saat menentukan paket dari PyPi.org, selain paket pada URL publik, seperti URL repo yang sesuai dengan PEP 503 kustom.

```
aws-batch == 0.6
    phoenix-letter >= 0.3

    --index-url http://dist.repoze.org/zope2/2.10/simple
    zopelib
```

Pemantauan dan metrik untuk Alur Kerja Terkelola Amazon untuk Apache Airflow

Pemantauan merupakan bagian penting dalam menjaga keandalan, ketersediaan, dan kinerja Alur Kerja Terkelola Amazon untuk Apache Airflow dan solusi Anda. AWS Kami merekomendasikan untuk mengumpulkan data pemantauan dari semua bagian AWS solusi Anda sehingga Anda dapat lebih mudah men-debug kegagalan multi-titik jika terjadi. Topik ini menjelaskan sumber daya apa yang AWS disediakan untuk memantau lingkungan Amazon MWAA Anda dan menanggapi peristiwa potensial.

> **Note**
>
> Metrik dan pencatatan Apache Airflow tunduk pada harga Amazon standar. CloudWatch

Untuk informasi lebih lanjut tentang pemantauan Apache Airflow, lihat Logging & Monitoring di situs web dokumentasi Apache Airflow.

Bagian-bagian

- Ikhtisar pemantauan di Amazon MWAA
- Mengakses log audit di AWS CloudTrail
- Mengakses log Aliran Udara di Amazon CloudWatch
- Memantau dasbor dan alarm di Amazon MWAA
- Metrik lingkungan Apache Airflow di CloudWatch
- Metrik kontainer, antrian, dan database untuk Amazon MWAA

Ikhtisar pemantauan di Amazon MWAA

Halaman ini menjelaskan AWS layanan yang digunakan untuk memantau Alur Kerja Terkelola Amazon untuk lingkungan Apache Airflow.

Daftar Isi

- CloudWatch Ikhtisar Amazon
- AWS CloudTrail selayang pandang

CloudWatch Ikhtisar Amazon

CloudWatch adalah repositori metrik untuk AWS layanan yang dapat Anda gunakan untuk mengambil statistik berdasarkan metrik dan dimensi yang diterbitkan oleh layanan. Anda dapat menggunakan metrik ini untuk mengonfigurasi alarm, menghitung statistik, dan kemudian menyajikan data di dasbor yang membantu Anda menilai kesehatan lingkungan Anda di konsol Amazon CloudWatch .

Apache Airflow sudah diatur untuk mengirim metrik StatSD untuk Alur Kerja Terkelola Amazon untuk lingkungan Apache Airflow ke Amazon. CloudWatch

Untuk mempelajari lebih lanjut, lihat Apa itu Amazon CloudWatch? .

AWS CloudTrail selayang pandang

CloudTrail adalah layanan audit yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di Amazon MWAA. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat ke Amazon MWAA, alamat IP dari mana permintaan itu dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan yang tersedia di log audit.

Untuk mempelajari lebih lanjut, lihat Apa itu AWS CloudTrail? .

Mengakses log audit di AWS CloudTrail

AWS CloudTrail diaktifkan pada Anda Akun AWS saat Anda membuatnya. CloudTrail mencatat aktivitas yang diambil oleh entitas IAM atau AWS layanan, seperti Alur Kerja Terkelola Amazon untuk Apache Airflow, yang dicatat sebagai peristiwa. CloudTrail Anda dapat melihat, mencari, dan mengunduh riwayat acara 90 hari terakhir di CloudTrail konsol. CloudTrail menangkap semua peristiwa di konsol Amazon MWAA dan semua panggilan ke Amazon MWAA API. Itu tidak menangkap tindakan hanya-baca, sepertiGetEnvironment, atau tindakan. PublishMetrics Halaman ini menjelaskan cara menggunakan CloudTrail untuk memantau peristiwa untuk Amazon MWAA.

Daftar Isi

- Membuat jejak di CloudTrail
- Mengakses acara dengan Riwayat CloudTrail Acara
- Contoh jejak untuk CreateEnvironment
- Apa selanjutnya?

Membuat jejak di CloudTrail

Anda perlu membuat jejak untuk mengakses catatan acara yang sedang berlangsung di Anda Akun AWS, termasuk acara untuk Amazon MWAA. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Jika Anda tidak membuat jejak, Anda masih dapat mengakses riwayat acara yang tersedia di CloudTrail konsol. Misalnya, dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat ke Amazon MWAA, alamat IP dari mana permintaan itu dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan. Untuk mempelajari lebih lanjut, lihat Membuat jejak untuk Anda Akun AWS.

Mengakses acara dengan Riwayat CloudTrail Acara

Anda dapat memecahkan masalah insiden operasional dan keamanan selama 90 hari terakhir di CloudTrail konsol dengan melihat riwayat acara. Misalnya, Anda dapat mengakses peristiwa yang terkait dengan pembuatan, modifikasi, atau penghapusan sumber daya (seperti pengguna IAM atau AWS sumber daya lainnya) di basis Anda per Akun AWS wilayah. Untuk mempelajari lebih lanjut, lihat Mengakses Acara dengan Riwayat CloudTrail Acara.

1. Buka konsol CloudTrail.
2. Pilih Riwayat acara.
3. Pilih acara yang ingin Anda lihat, lalu pilih Bandingkan detail acara.

Contoh jejak untuk **`CreateEnvironment`**

Jejak adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket Amazon S3 yang telah Anda tentukan.

CloudTrail file log berisi satu atau lebih entri log. Peristiwa mewakili permintaan tunggal dari sumber manapun dan mencakup informasi tentang tindakan yang diminta, seperti tanggal dan waktu tindakan, atau parameter permintaan. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, dan tidak terdaftar dalam urutan tertentu. Contoh berikut adalah entri log untuk `CreateEnvironment` tindakan yang ditolak karena kurangnya izin. Nilai-nilai di `AirflowConfigurationOptions` telah disunting untuk privasi.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
```

```
    "principalId": "00123456ABC7DEF8HIJK",
    "arn": "arn:aws:sts::012345678901:assumed-role/root/myuser",
    "accountId": "012345678901",
    "accessKeyId": "",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "00123456ABC7DEF8HIJK",
        "arn": "arn:aws:iam::012345678901:role/user",
        "accountId": "012345678901",
        "userName": "user"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-10-07T15:51:52Z"
      }
    }
  },
  "eventTime": "2020-10-07T15:52:58Z",
  "eventSource": "airflow.amazonaws.com",
  "eventName": "CreateEnvironment",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "205.251.233.178",
  "userAgent": "PostmanRuntime/7.26.5",
  "errorCode": "AccessDenied",
  "requestParameters": {
    "SourceBucketArn": "arn:aws:s3:::my-bucket",
    "ExecutionRoleArn": "arn:aws:iam::012345678901:role/AirflowTaskRole",
    "AirflowConfigurationOptions": "***",
    "DagS3Path": "sample_dag.py",
    "NetworkConfiguration": {
      "SecurityGroupIds": [
      "sg-01234567890123456"
      ],
      "SubnetIds": [
        "subnet-01234567890123456",
        "subnet-65432112345665431"
      ]
    },
    "Name": "test-cloudtrail"
  },
  "responseElements": {
    "message": "Access denied."
```

```
  },
  "requestID": "RequestID",
  "eventID": "EventID",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "recipientAccountId": "012345678901"
}
```

Apa selanjutnya?

- Pelajari cara mengonfigurasi AWS layanan lain untuk data peristiwa yang dikumpulkan dalam CloudTrail log di Layanan dan Integrasi yang CloudTrail Didukung.
- Pelajari cara diberi tahu saat memublikasikan CloudTrail file log baru ke bucket Amazon S3 di Mengonfigurasi Notifikasi Amazon SNS untuk. CloudTrail

Mengakses log Aliran Udara di Amazon CloudWatch

Amazon MWAA dapat mengirim log Apache Airflow ke Amazon. CloudWatch Anda dapat mengakses log untuk beberapa lingkungan dari satu lokasi untuk dengan mudah mengidentifikasi penundaan tugas Apache Airflow atau kesalahan alur kerja tanpa alat pihak ketiga tambahan. Log Apache Airflow harus diaktifkan di Amazon Managed Workflows untuk konsol Apache Airflow untuk mengakses pemrosesan Apache Airflow DAG, tugas, server web, log masuk pekerja. CloudWatch

Daftar Isi

- Harga
- Sebelum Anda mulai
- Jenis log
- Mengaktifkan log Apache Airflow
- Mengakses log Apache Airflow
- Contoh log penjadwal
- Apa selanjutnya?

Harga

- Biaya CloudWatch Log Standar berlaku. Untuk informasi lebih lanjut, lihat CloudWatch harga.

Sebelum Anda mulai

- Anda harus memiliki peran yang dapat mengakses log in CloudWatch. Untuk informasi lebih lanjut, lihatMengakses lingkungan Amazon MWAA.

Jenis log

Amazon MWAA membuat grup log untuk setiap opsi pencatatan Aliran Udara yang Anda aktifkan, dan mendorong log ke grup Log yang CloudWatch terkait dengan lingkungan. Grup log diberi nama dalam format berikut:`YourEnvironmentName-`**`LogType`**. Misalnya, jika nama lingkungan Anda`Airflow-v202-Public`, log tugas Apache Airflow akan dikirim ke. `Airflow-v202-Public-`**`Task`**

Jenis log	Deskripsi
`YourEnvironmentName-` **`DAGProcesing`**	Log manajer prosesor DAG (bagian dari penjadwal yang memproses file DAG).
`YourEnvironmentName-` **`Scheduler`**	Log yang dihasilkan oleh penjadwal Aliran Udara.
`YourEnvironmentName-` **`Task`**	Tugas mencatat hasil DAG.
`YourEnvironmentName-` **`WebServer`**	Log yang dihasilkan antarmuka web Airflow.
`YourEnvironmentName-` **`Worker`**	Log yang dihasilkan sebagai bagian dari alur kerja dan eksekusi DAG.

Mengaktifkan log Apache Airflow

Anda dapat mengaktifkan log Apache Airflow di `INFO`,`WARNING`,`ERROR`, atau level. `CRITICAL` Saat Anda memilih level log, Amazon MWAA mengirimkan log untuk level tersebut dan semua tingkat keparahan yang lebih tinggi. Misalnya, jika Anda mengaktifkan log di `INFO` level tersebut, Amazon MWAA mengirimkan `INFO` log dan`WARNING`,`ERROR`, dan tingkat `CRITICAL` log ke CloudWatch Log.

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.

3. Pilih Edit.
4. Pilih Berikutnya.
5. Pilih satu atau beberapa opsi pencatatan berikut:
    a. Pilih grup log penjadwal aliran udara di panel Pemantauan.
    b. Pilih grup log server web Aliran udara di panel Pemantauan.
    c. Pilih grup log pekerja Aliran udara di panel Pemantauan.
    d. Pilih grup log pemrosesan Aliran Udara DAG di panel Pemantauan.
    e. Pilih grup log tugas Aliran udara di panel Pemantauan.
    f. Pilih level logging di level Log.
6. Pilih Berikutnya.
7. Pilih Simpan.

Mengakses log Apache Airflow

Bagian berikut menjelaskan cara mengakses log Apache Airflow di konsol. CloudWatch

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.
3. Pilih grup log di panel Monitoring.
4. Pilih log masuk Aliran log.

Contoh log penjadwal

Anda dapat mengakses log Apache Airflow untuk penjadwal yang menjadwalkan alur kerja Anda dan mengurai folder Anda. `dags` Langkah-langkah berikut menjelaskan cara membuka grup log untuk penjadwal di konsol Amazon MWAA, dan mengakses log Apache Airflow di konsol Log. CloudWatch

Untuk mengakses log untuk **`requirements.txt`**

1. Buka halaman Lingkungan di konsol Amazon MWAA.
2. Pilih lingkungan.
3. Pilih grup log penjadwal aliran udara di panel Pemantauan.
4. Pilih `requirements_install_ip` log masuk Aliran log.

5. Lihat daftar paket yang diinstal pada lingkungan di`/usr/local/airflow/.local/bin`. Contoh:

```
Collecting appdirs==1.4.4 (from -r /usr/local/airflow/.local/bin (line 1))
Downloading https://files.pythonhosted.org/
packages/3b/00/2344469e2084fb28kjdsfiuyweb47389789vxbmnbjhsdgf5463acd6cf5e3db69324/
appdirs-1.4.4-py2.py3-none-any.whl
Collecting astroid==2.4.2 (from -r /usr/local/airflow/.local/bin (line 2))
```

6. Tinjau daftar paket dan apakah salah satu dari ini mengalami kesalahan selama instalasi. Jika terjadi kesalahan, Anda mungkin mendapatkan kesalahan yang mirip dengan berikut ini:

```
2021-03-05T14:34:42.731-07:00
No matching distribution found for LibraryName==1.0.0 (from -r /usr/local/
airflow/.local/bin (line 4))
No matching distribution found for LibraryName==1.0.0 (from -r /usr/local/
airflow/.local/bin (line 4))
```

Apa selanjutnya?

- Pelajari cara mengonfigurasi CloudWatch alarm di Menggunakan CloudWatch alarm Amazon.
- Pelajari cara membuat CloudWatch dasbor di Menggunakan CloudWatch dasbor.

Memantau dasbor dan alarm di Amazon MWAA

Anda dapat membuat dasbor khusus di Amazon CloudWatch dan menambahkan alarm untuk metrik tertentu guna memantau status kesehatan Alur Kerja Terkelola Amazon untuk lingkungan Apache Airflow. Saat alarm ada di dasbor, alarm berubah menjadi merah saat berada di `ALARM` negara bagian, membuatnya lebih mudah untuk memantau kesehatan lingkungan Amazon MWAA secara proaktif.

Apache Airflow mengekspos metrik untuk beberapa proses, termasuk jumlah proses DAG, ukuran tas DAG, tugas yang sedang berjalan, kegagalan tugas, dan keberhasilan. Saat Anda membuat lingkungan, Airflow secara otomatis mengirimkan metrik untuk lingkungan Amazon MWAA. CloudWatch Halaman ini menjelaskan cara membuat dasbor status kesehatan untuk metrik Aliran Udara CloudWatch untuk lingkungan Amazon MWAA.

Daftar Isi

- Metrik-metrik
- Ikhtisar status alarm
- Contoh dasbor dan alarm khusus
  - Tentang metrik ini
  - Tentang dasbor
  - Penggunaan AWS tutorial
  - Penggunaan CloudFormation
- Menghapus metrik dan dasbor
- Apa selanjutnya?

Metrik-metrik

Anda dapat membuat dasbor dan alarm khusus untuk metrik apa pun yang tersedia untuk versi Apache Airflow Anda. Setiap metrik sesuai dengan indikator kinerja kunci Apache Airflow (KPI). Untuk mengakses daftar metrik, lihat:

- Metrik lingkungan Apache Airflow di CloudWatch

Ikhtisar status alarm

Sebuah alarm metrik mungkin saja berada dalam status berikut ini:

- `OK` – Metrik atau ekspresi berada dalam ambang batas yang telah ditetapkan sebelumnya.
- `ALARM` – Metrik atau ekspresi berada di luar ambang batas yang telah ditetapkan sebelumnya.
- `INSUFFICIENT_DATA` – Alarm baru saja dimulai, metrik tidak tersedia, atau tidak ada data yang memadai yang tersedia bagi metrik untuk menentukan status alarm.

Contoh dasbor dan alarm khusus

Anda dapat membuat dasbor pemantauan khusus yang menampilkan bagan metrik yang dipilih untuk lingkungan Amazon MWAA Anda.

Tentang metrik ini

Daftar berikut menjelaskan masing-masing metrik yang dibuat di dasbor kustom oleh tutorial dan definisi template di bagian ini.

- QueuedTasks- Jumlah tugas dengan status antrian. Sesuai dengan metrik `executor.queued_tasks` Apache Airflow.
- TasksPending- Jumlah tugas yang tertunda di pelaksana. Sesuai dengan metrik `scheduler.tasks.pending` Apache Airflow.

> **Note**
>
> Tidak berlaku untuk Apache Airflow v2.2 dan yang lebih baru.

- RunningTasks- Jumlah tugas yang berjalan di pelaksana. Sesuai dengan metrik `executor.running_tasks` Apache Airflow.
- SchedulerHeartbeat- Jumlah check-in Apache Airflow melakukan pada pekerjaan scheduler. Sesuai dengan metrik `scheduler_heartbeat` Apache Airflow.
- TotalParseTime- Jumlah detik yang diambil untuk memindai dan mengimpor semua file DAG sekali. Sesuai dengan metrik `dag_processing.total_parse_time` Apache Airflow.

Tentang dasbor

Gambar berikut menampilkan dasbor pemantauan yang dibuat oleh definisi tutorial dan template di bagian ini.

Penggunaan AWS tutorial

Anda dapat menggunakan AWS tutorial berikut untuk secara otomatis membuat dasbor status kesehatan untuk lingkungan Amazon MWAA yang saat ini digunakan. Ini juga menciptakan CloudWatch alarm untuk pekerja yang tidak sehat dan kegagalan detak jantung penjadwal di semua lingkungan Amazon MWAA.

- CloudWatch Otomatisasi Dasbor untuk Amazon MWAA

Penggunaan CloudFormation

Anda dapat menggunakan definisi CloudFormation templat di bagian ini untuk membuat dasbor pemantauan CloudWatch, lalu menambahkan alarm di CloudWatch konsol untuk menerima notifikasi saat metrik melampaui ambang batas tertentu. Untuk membuat tumpukan menggunakan definisi template ini, lihat Membuat tumpukan di CloudFormation konsol. Untuk menambahkan alarm ke dasbor, lihat Menggunakan alarm.

```
AWSTemplateFormatVersion: "2010-09-09"
Description: Creates MWAA Cloudwatch Dashboard
Parameters:
  DashboardName:
    Description: Enter the name of the CloudWatch Dashboard
    Type: String
  EnvironmentName:
    Description: Enter the name of the MWAA Environment
    Type: String
Resources:
  BasicDashboard:
    Type: AWS::CloudWatch::Dashboard
    Properties:
      DashboardName: !Ref DashboardName
      DashboardBody:
        Fn::Sub: '{
              "widgets": [
                  {
                      "type": "metric",
                      "x": 0,
                      "y": 0,
                      "width": 12,
                      "height": 6,
                      "properties": {
                          "view": "timeSeries",
                          "stacked": true,
                          "metrics": [
                              [
                                  "AmazonMWAA",
                                  "QueuedTasks",
                                  "Function",
                                  "Executor",
                                  "Environment",
                                  "${EnvironmentName}"
                              ]
```

```
                ],
                "region": "${AWS::Region}",
                "title": "QueuedTasks ${EnvironmentName}",
                "period": 300
            }
        },
        {
            "type": "metric",
            "x": 0,
            "y": 6,
            "width": 12,
            "height": 6,
            "properties": {
                "view": "timeSeries",
                "stacked": true,
                "metrics": [
                    [
                        "AmazonMWAA",
                        "RunningTasks",
                        "Function",
                        "Executor",
                        "Environment",
                        "${EnvironmentName}"
                    ]
                ],
                "region": "${AWS::Region}",
                "title": "RunningTasks ${EnvironmentName}",
                "period": 300
            }
        },
        {
            "type": "metric",
            "x": 12,
            "y": 6,
            "width": 12,
            "height": 6,
            "properties": {
                "view": "timeSeries",
                "stacked": true,
                "metrics": [
                    [
                        "AmazonMWAA",
                        "SchedulerHeartbeat",
                        "Function",
```

```
                        "Scheduler",
                        "Environment",
                        "${EnvironmentName}"
                    ]
                ],
                "region": "${AWS::Region}",
                "title": "SchedulerHeartbeat ${EnvironmentName}",
                "period": 300
            }
        },
        {
            "type": "metric",
            "x": 12,
            "y": 0,
            "width": 12,
            "height": 6,
            "properties": {
                "view": "timeSeries",
                "stacked": true,
                "metrics": [
                    [
                        "AmazonMWAA",
                        "TasksPending",
                        "Function",
                        "Scheduler",
                        "Environment",
                        "${EnvironmentName}"
                    ]
                ],
                "region": "${AWS::Region}",
                "title": "TasksPending ${EnvironmentName}",
                "period": 300
            }
        },
        {
            "type": "metric",
            "x": 0,
            "y": 12,
            "width": 24,
            "height": 6,
            "properties": {
                "view": "timeSeries",
                "stacked": true,
                "region": "${AWS::Region}",
```

```
                        "metrics": [
                            [
                                "AmazonMWAA",
                                "TotalParseTime",
                                "Function",
                                "DAG Processing",
                                "Environment",
                                "${EnvironmentName}"
                            ]
                        ],
                        "title": "TotalParseTime  ${EnvironmentName}",
                        "period": 300
                    }
                }
            ]
        }'
```

Menghapus metrik dan dasbor

Jika Anda menghapus lingkungan Amazon MWAA, dasbor yang sesuai juga dihapus. CloudWatch metrik disimpan selama lima belas (15) bulan dan tidak dapat dihapus. CloudWatch Konsol membatasi pencarian metrik hingga dua (2) minggu setelah metrik terakhir dicerna untuk memastikan bahwa instance terbaru ditampilkan untuk lingkungan Amazon MWAA Anda. Untuk mempelajari lebih lanjut, lihat CloudWatch FAQ Amazon.

Apa selanjutnya?

- Pelajari cara membuat DAG yang menanyakan database metadata Amazon Aurora PostgreSQL untuk lingkungan Anda dan menerbitkan metrik kustom ke dalam. CloudWatch Menggunakan DAG untuk menulis metrik khusus CloudWatch

Metrik lingkungan Apache Airflow di CloudWatch

Apache Airflow v2 dan v3 sudah disiapkan untuk mengumpulkan dan mengirim metrik StatSD untuk Alur Kerja Terkelola Amazon untuk lingkungan Apache Airflow ke Amazon. CloudWatch Daftar lengkap metrik yang dikirim Apache Airflow tersedia di halaman Metrik di panduan referensi Apache Airflow. Halaman ini menjelaskan metrik Apache Airflow yang tersedia dan cara mengakses metrik CloudWatch di konsol. CloudWatch

Daftar Isi

- Istilah
- Dimensi
- Mengakses metrik di konsol CloudWatch
- Metrik Apache Airflow tersedia di CloudWatch
  - Penghitung Aliran Udara Apache
  - Pengukur Aliran Udara Apache
  - Pengatur Waktu Aliran Udara Apache
- Memilih metrik mana yang dilaporkan
- Apa selanjutnya?

Istilah

Namespace

Namespace adalah wadah untuk CloudWatch metrik layanan. AWS Untuk Amazon MWAA, namespace adalah AmazonMWAA.

CloudWatch metrik

CloudWatch Metrik mewakili kumpulan titik data yang diurutkan waktu yang spesifik untuk CloudWatch.

Metrik Aliran Udara Apache

Metrik khusus untuk Apache Airflow.

Dimensi

Dimensi adalah name/value pasangan yang merupakan bagian dari identitas metrik.

Unit

Statistik memiliki satuan ukuran. Untuk Amazon MWAA, unit termasuk Hitung, Detik, dan Millidetik. Untuk Amazon MWAA, unit ditetapkan berdasarkan unit dalam metrik Aliran Udara asli.

Dimensi

Bagian ini menjelaskan pengelompokan CloudWatch Dimensi untuk metrik Apache Airflow di. CloudWatch

Dimensi	Deskripsi
HARI	Menunjukkan nama Apache Airflow DAG tertentu.
DAG Nama file	Menunjukkan nama file Apache Airflow DAG tertentu.
Fungsi	Dimensi ini digunakan untuk meningkatkan pengelompokan metrik di. CloudWatch
Pekerjaan	Menunjukkan pekerjaan Apache Airflow yang dijalankan oleh penjadwal . Selalu memiliki nilai`Job`.
Operator	Menunjukkan operator Apache Airflow tertentu.
Kolam	Menunjukkan kumpulan pekerja Apache Airflow tertentu.
Tugas	Menunjukkan tugas Apache Airflow tertentu.
HostName	Menunjukkan nama host untuk proses Apache Airflow tertentu yang berjalan.

Mengakses metrik di konsol CloudWatch

Bagian ini menjelaskan cara mengakses metrik kinerja CloudWatch untuk DAG tertentu.

Untuk mengakses metrik kinerja untuk dimensi

1. Buka halaman Metrik di CloudWatch konsol.
2. Pilih Anda Wilayah AWS.
3. Pilih namespace AmazonMwaa.
4. Di tab Semua metrik, pilih dimensi. Misalnya, DAG, Lingkungan.
5. Pilih CloudWatch metrik untuk dimensi. Misalnya, TaskInstanceSuccesses atau TaskInstanceDuration. Pilih Grafik semua hasil pencarian.
6. Pilih tab Graphed metrics untuk mengakses statistik performa metrik Apache Airflow, seperti DAG, Environment, Task.

Metrik Apache Airflow tersedia di CloudWatch

Bagian ini menjelaskan metrik dan dimensi Apache Airflow yang dikirim. CloudWatch

Penghitung Aliran Udara Apache

Metrik Apache Airflow di bagian ini berisi data tentang Apache Airflow Counters.

CloudWatch metrik	Metrik Aliran Udara Apache	Unit	Dimensi	
Membanting **Note** Hanya tersedia untuk Apache Airflow v2.4.3 hingga v2.10.3.	sla_terlewatkan	Hitungan	Fungsi, Penjadwal	
FailedSlaCallback	sla_callback_notification_failure	Hitungan	Fungsi, Penjadwal	

CloudWatch metrik	Metrik Aliran Udara Apache	Unit	Dimensi	
Note Hanya tersedia untuk Apache Airflow v2.4.3 hingga v2.10.3.				
Pembaruan **Note** Tersedia untuk Apache Airflow v2.6.3 dan yang lebih baru.	dataset.updates	Hitungan	Fungsi, Penjadwal	
Yatim piatu **Note** Tersedia untuk Apache Airflow v2.6.3 dan yang lebih baru.	dataset.yatim piatu	Hitungan	Fungsi, Penjadwal	
FailedCeleryTaskExecution **Note** Tersedia untuk Apache Airflow v2.4.3 dan yang lebih baru.	celery.execute_command.failure	Hitungan	Fungsi, Seledri	

CloudWatch metrik	Metrik Aliran Udara Apache	Unit	Dimensi	
FilePathQueueUpdateCount Note Tersedia untuk Apache Airflow v2.6.3 dan yang lebih baru.	dag_processing.file_path_queue_update_count	Hitungan	Fungsi, Penjadwal	
CriticalSectionBusy	scheduler.critical_section_busy	Hitungan	Fungsi, Penjadwal	
DagBagSize	dagbag_size	Hitungan	Fungsi, Pemrosesan DAG	
DagCallbackExceptions	dag.callback_exceptions	Hitungan	DAG, Semua	
FailedSLAEmailAttempts Note Tidak tersedia untuk Apache Airflow v3.0.6 dan yang lebih baru.	sla_email_notification_failure	Hitungan	Fungsi, Penjadwal	

CloudWatch metrik	Metrik Aliran Udara Apache	Unit	Dimensi	
TaskInstanceFinished	ti.finish.{dag_id}.{task_id}.{negara}	Hitungan	DAG, {dag_id} Tugas, {task_id} Negara, {state}	
JobEnd	{job_name}_akhir	Hitungan	Job, {job_name}	
JobHeartbeatFailure	{job_name}_heartbeat_failure	Hitungan	Job, {job_name}	
JobStart	{job_name}_mulai	Hitungan	Job, {job_name}	
ManagerStalls	dag_processing.manager_stalls	Hitungan	Fungsi, Pemrosesan DAG	
OperatorFailures	operator_failures_{operator_name}	Hitungan	Operator, {operator_name}	

CloudWatch metrik	Metrik Aliran Udara Apache	Unit	Dimensi	
OperatorSuccesses	operator_successes_{operator_name}	Hitungan	Operator, {operator_name}	
OtherCallbackCount **Note** Tersedia dalam Apache Airflow v2.6.3 dan yang lebih baru.	dag_processing.other_callback_count	Hitungan	Fungsi, Penjadwal	
Proses	dag_processing.processes	Hitungan	Fungsi, Pemrosesan DAG	
SchedulerHeartbeat	scheduler_heartbeat	Hitungan	Fungsi, Penjadwal	
StartedTaskInstances	ti.start.{dag_id}.{task_id}	Hitungan	DAG, Semua Tugas, Semua	

CloudWatch metrik	Metrik Aliran Udara Apache	Unit	Dimensi	
SlaCallbackCount	dag_processing.sla_callback_count **Note** Tersedia untuk Apache Airflow v2.6.3 dan yang lebih baru.	Hitungan	Fungsi, Penjadwal	
TasksKilledExternally	scheduler.tasks.killed_externally	Hitungan	Fungsi, Penjadwal	
TaskTimeoutError	celery.task_timeout_error	Hitungan	Fungsi, Seledri	
TaskInstanceCreatedUsingOperator	task_instance_created- {operator_name}	Hitungan	Operator, {operator_name}	

CloudWatch metrik	Metrik Aliran Udara Apache	Unit	Dimensi	
TaskInstancePreviouslySucceeded	sebelumnya_berhasil	Hitungan	DAG, Semua Tugas, Semua	
TaskInstanceFailures	ti_failure	Hitungan	DAG, Semua Tugas, Semua	
TaskInstanceSuccesses	ti_sukses	Hitungan	DAG, Semua Tugas, Semua	
TaskRemovedFromDAG	task_removed_from_dag. {dag_id}	Hitungan	DAG, {dag_id}	
TaskRestoredToDAG	task_restored_to_dag. {dag_id}	Hitungan	DAG, {dag_id}	
TriggersSucceeded **Note** Tersedia untuk Apache Airflow v2.7.2 dan yang lebih baru.	pemicu.berhasil	Hitungan	Fungsi, Pemicu	

CloudWatch metrik	Metrik Aliran Udara Apache	Unit	Dimensi	
TriggersFailed Note Tersedia untuk Apache Airflow v2.7.2 dan yang lebih baru.	pemicu.gagal	Hitungan	Fungsi, Pemicu	
TriggersBlockedMainThread Note Tersedia untuk Apache Airflow v2.7.2 dan yang lebih baru.	triggers.blocked_main_thread	Hitungan	Fungsi, Pemicu	
TriggerHeartbeat Note Tersedia untuk Apache Airflow v2.8.1 dan yang lebih baru.	triggerer_detak jantung	Hitungan	Fungsi, Pemicu	

CloudWatch metrik	Metrik Aliran Udara Apache	Unit	Dimensi	
TaskInstanceCreatedUsingOperator	airflow.task_instance_created_`{operator_name}` **Note** Tersedia untuk Apache Airflow v2.7.2 dan yang lebih baru.	Hitungan	Operator, `{operator_name}`	
ZombiesKilled	zombie_dibunuh	Hitungan	DAG, Semua Tugas, Semua	

Pengukur Aliran Udara Apache

Metrik Apache Airflow di bagian ini berisi data tentang Apache Airflow Gauges.

CloudWatch metrik	Metrik Aliran Udara Apache	Unit	Dimensi	

CloudWatch metrik	Metrik Aliran Udara Apache	Unit	Dimensi	
DAGFileRefreshError	dag_file_refresh_error	Hitungan	Fungsi, Pemrosesan DAG	
ImportErrors	dag_processing.import_errors	Hitungan	Fungsi, Pemrosesan DAG	
Exception Failures	smart_sensor_operator.exception_failure	Hitungan	Fungsi, Operator Sensor Cerdas	
ExecutedTasks	smart_sensor_operator.executed_tasks	Hitungan	Fungsi, Operator Sensor Cerdas	
InfraFailures	smart_sensor_operator.infra_failure	Hitungan	Fungsi, Operator Sensor Cerdas	
LoadedTasks	smart_sensor_operator.loaded_tasks	Hitungan	Fungsi, Operator Sensor Cerdas	
TotalParseTime	dag_processing.total_parse_time	Detik	Fungsi, Pemrosesan DAG	

CloudWatch metrik	Metrik Aliran Udara Apache	Unit	Dimensi	
Triggered DagRuns **Note** Tersedia dalam Apache Airflow v2.6.3 dan yang lebih baru.	dataset.triggered_dagruns	Hitungan	Fungsi, Penjadwal	
TriggersRunning **Note** Tersedia dalam Apache Airflow v2.7.2 dan yang lebih baru.	pemicu.running.*{hostname}*	Hitungan	Fungsi, Pemicu HostName, *{hostname}*	

CloudWatch metrik	Metrik Aliran Udara Apache	Unit	Dimensi
PoolDeferredSlots Note Tersedia dalam Apache Airflow v2.7.2 dan yang lebih baru.	pool.deferred_slots.{pool_name}	Hitungan	Kolam renang, {pool_name}
DAGFileProcessingLastRunSecondsAgo	dag_processing.last_run.seconds_ago.{dag_nama file}	Detik	DAG Nama file, {dag_file name}
OpenSlots	executor.open_slots	Hitungan	Fungsi, Pelaksana
OrphanedTasksAdopted	scheduler.orphaned_tasks.adopted	Hitungan	Fungsi, Penjadwal
OrphanedTasksCleared	scheduler.orphaned_tasks.cleared	Hitungan	Fungsi, Penjadwal

CloudWatch metrik	Metrik Aliran Udara Apache	Unit	Dimensi	
PokedExceptions	smart_sensor_operator.poked_exception	Hitungan	Fungsi, Operator Sensor Cerdas	
PokedSuccess	smart_sensor_operator.poked_success	Hitungan	Fungsi, Operator Sensor Cerdas	
PokedTasks	smart_sensor_operator.poked_tasks	Hitungan	Fungsi, Operator Sensor Cerdas	
PoolFailures	pool.open_slots.{pool_name}	Hitungan	Kolam renang, {pool_name}	
PoolStarvingTasks	pool.starving_tasks.{pool_name}	Hitungan	Kolam renang, {pool_name}	
PoolOpenSlots	pool.open_slots.{pool_name}	Hitungan	Kolam renang, {pool_name}	
PoolQueuedSlots	pool.queued_slots.{pool_name}	Hitungan	Kolam renang, {pool_name}	
PoolRunningSlots	pool.running_slots.{pool_name}	Hitungan	Kolam renang, {pool_name}	
Processor Timeouts	dag_processing.processor_timeouts	Hitungan	Fungsi, Pemrosesan DAG	

CloudWatch metrik	Metrik Aliran Udara Apache	Unit	Dimensi
QueuedTasks	executor.queued_tasks	Hitungan	Fungsi, Pelaksana
RunningTasks	executor.running_tasks	Hitungan	Fungsi, Pelaksana
TasksExecutable	scheduler.tasks.executable	Hitungan	Fungsi, Penjadwal
TasksPending **Note** Tidak berlaku untuk Apache Airflow v2.2 dan yang lebih baru.	scheduler.tasks.pending	Hitungan	Fungsi, Penjadwal
TasksRunning	scheduler.tasks.running	Hitungan	Fungsi, Penjadwal
TasksStarving	scheduler.tasks.starving	Hitungan	Fungsi, Penjadwal

CloudWatch metrik	Metrik Aliran Udara Apache	Unit	Dimensi
TasksWithoutDagRun	scheduler.tasks.without_dagrun	Hitungan	Fungsi, Penjadwal
DAGFileProcessingLastNumOfDbQueries **Note** Tersedia dalam Apache Airflow v2.10.1 dan yang lebih baru.	dag_processing.last_num_of_db_queries.{dag_nama file}	Hitungan	DAG Nama file, {dag_file name}

CloudWatch metrik	Metrik Aliran Udara Apache	Unit	Dimensi	
PoolScheduledSlots Note Tersedia dalam Apache Airflow v2.10.1 dan yang lebih baru.	pool.scheduled_slots. {pool_name}	Hitungan	Kolam renang, {pool_name}	
TaskCpuUsage Note Tersedia dalam Apache Airflow v2.10.1 dan yang lebih baru.	cpu.usage. {dag_id}. {task_id}	Persen	DAG, {dag_id} Tugas, {task_id}	

CloudWatch metrik	Metrik Aliran Udara Apache	Unit	Dimensi	
TaskMemoryUsage Note Tersedia dalam Apache Airflow v2.10.1 dan yang lebih baru.	mem.penggunaan. {dag_id}. {task_id}	Persen	DAG, {dag_id} Tugas, {task_id}	

Pengatur Waktu Aliran Udara Apache

Metrik Apache Airflow di bagian ini berisi data tentang Apache Airflow Timers.

CloudWatch metrik	Metrik Aliran Udara Apache	Unit	Dimensi	
CollectdBdags	collect_db_dags	Milidetik	Fungsi, Pemrosesan DAG	
CriticalSectionDuration	scheduler .critical_section_duration	Milidetik	Fungsi, Penjadwal	
		Milidetik		

CloudWatch metrik	Metrik Aliran Udara Apache	Unit	Dimensi	
CriticalSectionQueryDuration Note Tersedia untuk Apache Airflow v2.5.1 dan yang lebih baru.	scheduler.critical_section_query_duration		Fungsi, Penjadwal	
DAGDependencyCheck	dagrun.dependency-check.{dag_id}	Milidetik	DAG, {dag_id}	
DAGDurationFailed	dagrun.duration.failed.{dag_id}	Milidetik	DAG, {dag_id}	
DAGDurationSuccess	dagrun.duration.success.{dag_id}	Milidetik	DAG, {dag_id}	
DAGFileProcessingLastDuration	dag_processing.last_duration.{dag_nama file}	Detik	DAG Nama file, {dag_filename}	

CloudWatch metrik	Metrik Aliran Udara Apache	Unit	Dimensi	
DAGSchedu leDelay	dagrun.sc hedule_delay. {dag_id}	Milidetik	DAG, {dag_id}	
FirstTask SchedulingDelay	dagrun. {dag_id} .first_ta sk_schedu ling_delay	Milidetik	DAG, {dag_id}	
Scheduler LoopDuration Note Tersedia untuk Apache Airflow v2.5.1 dan yang lebih baru.	scheduler .schedule r_loop_duration	Milidetik	Fungsi, Penjadwal	
TaskInsta nceDuration	hari ini. {dag_id}. {task_id} .durasi	Milidetik	DAG, {dag_id} Tugas, {task_id}	

CloudWatch metrik	Metrik Aliran Udara Apache	Unit	Dimensi	
TaskInstanceQueuedDuration	hari ini. `{dag_id}`. `{task_id}` .antrian_durasi **Note** Tersedia untuk Apache Airflow v2.7.2 dan yang lebih baru.	Milidetik	DAG, {dag_id} Tugas, {task_id}	
TaskInstanceScheduledDuration **Note** Tersedia untuk Apache Airflow v2.7.2 dan yang lebih baru.	hari ini. `{dag_id}`. `{task_id}` .scheduled_duration	Milidetik	DAG, {dag_id} Tugas, {task_id}	

Memilih metrik mana yang dilaporkan

Anda dapat memilih metrik Apache Airflow mana yang dipancarkan, CloudWatch atau diblokir oleh Apache Airflow, menggunakan opsi konfigurasi Amazon MWAA berikut:

- **`metrics.metrics_allow_list`**— Daftar awalan dipisahkan koma yang dapat Anda gunakan untuk memilih metrik mana yang dipancarkan oleh lingkungan Anda. CloudWatch Gunakan opsi ini jika Anda ingin Apache Airflow tidak mengirim semua metrik yang tersedia dan sebagai gantinya pilih subset elemen. Misalnya, `scheduler,executor,dagrun`.
- **`metrics.metrics_block_list`**— Daftar awalan yang dipisahkan koma untuk menyaring metrik yang dimulai dengan elemen daftar. Misalnya, `scheduler,executor,dagrun`.

Jika Anda mengonfigurasi keduanya `metrics.metrics_allow_list` dan`metrics.metrics_block_list`, Apache Airflow mengabaikan. `metrics.metrics_block_list` Jika Anda mengonfigurasi `metrics.metrics_block_list` tetapi tidak`metrics.metrics_allow_list`, Apache Airflow memfilter elemen yang Anda tentukan. `metrics.metrics_block_list`

> **Note**
>
> Opsi `metrics.metrics_allow_list` dan `metrics.metrics_block_list` konfigurasi hanya berlaku untuk Apache Airflow v2.6.3 dan yang lebih baru. Untuk versi sebelumnya dari Apache Airflow `metrics.statsd_allow_list` gunakan dan sebagai gantinya. `metrics.statsd_block_list`

Apa selanjutnya?

- Jelajahi operasi Amazon MWAA API yang digunakan untuk mempublikasikan metrik kesehatan lingkungan di. PublishMetrics

Metrik kontainer, antrian, dan database untuk Amazon MWAA

Selain metrik Apache Airflow, Anda dapat memantau komponen dasar Alur Kerja Terkelola Amazon untuk CloudWatch lingkungan Apache Airflow menggunakan, yang mengumpulkan data mentah dan memproses data menjadi metrik yang dapat dibaca dan mendekati waktu nyata. Dengan metrik

lingkungan ini, Anda akan memiliki visibilitas yang lebih besar ke dalam indikator kinerja utama untuk membantu Anda mengukur lingkungan Anda dengan tepat dan masalah debug dengan alur kerja Anda. Metrik ini berlaku untuk semua versi Apache Airflow yang didukung di Amazon MWAA.

Amazon MWAA akan menyediakan pemanfaatan CPU dan memori untuk setiap wadah Amazon Elastic Container Service (Amazon ECS) Container Service (Amazon ECS) dan instance Amazon Aurora PostgreSQL, dan Amazon Simple Queue Service (Amazon SQS) untuk jumlah pesan dan usia pesan tertua, Amazon Relational Database Service Metrik Database Service (Amazon RDS) untuk koneksi database, kedalaman antrian disk, operasi tulis, latensi, dan throughput, dan metrik Amazon RDS Proxy. Metrik ini juga mencakup jumlah pekerja dasar, pekerja tambahan, penjadwal, dan server web.

Statistik ini disimpan selama 15 bulan, sehingga Anda dapat mengakses informasi historis dan mendapatkan perspektif yang lebih baik tentang mengapa jadwal gagal, dan memecahkan masalah mendasar. Anda juga dapat mengatur alarm yang memantau ambang batas tertentu, dan mengirim pemberitahuan atau mengambil tindakan ketika ambang batas tersebut terpenuhi. Untuk informasi selengkapnya, lihat Panduan CloudWatch Pengguna Amazon.

Topik

- Istilah
- Dimensi
- Mengakses metrik di konsol CloudWatch
- Daftar metrik

Istilah

Namespace

Namespace adalah wadah untuk CloudWatch metrik layanan. AWS Untuk Amazon MWAA, namespace adalah. `AWS/MWAA`

CloudWatch metrik

CloudWatch Metrik mewakili kumpulan titik data yang diurutkan waktu yang spesifik untuk CloudWatch.

Dimensi

Dimensi adalah name/value pasangan yang merupakan bagian dari identitas metrik.

Unit

Sebuah statistik memiliki satuan ukuran. Untuk Amazon MWAA, unit termasuk Count.

Dimensi

Bagian ini menjelaskan pengelompokan CloudWatch dimensi untuk metrik Amazon MWAA di. CloudWatch

Dimensi	Deskripsi
Kluster	Metrik untuk minimum tiga wadah Amazon ECS yang digunakan lingkungan Amazon MWAA untuk menjalankan komponen Apache Airflow: scheduler, worker, dan server web.
Antrean	Metrik untuk antrian Amazon SQS yang memisahkan penjadwal dari pekerja. Ketika pekerja membaca pesan, mereka dianggap dalam penerbangan dan tidak tersedia untuk pekerja lain. Pesan menjadi tersedia bagi pekerja lain untuk dibaca jika tidak dihapus sebelum batas waktu visibilitas 12 jam.
Basis Data	Metrik cluster Aurora yang digunakan oleh Amazon MWAA. Ini termasuk metrik untuk instance database utama dan replika baca untuk mendukung operasi baca. Amazon MWAA menerbitkan metrik database untuk instans READER dan WRITER.

Mengakses metrik di konsol CloudWatch

Bagian ini menjelaskan cara mengakses metrik Amazon MWAA Anda di. CloudWatch

Untuk mengakses metrik kinerja untuk dimensi

1. Buka halaman Metrik di CloudWatch konsol.

2. Pilih Anda Wilayah AWS.
3. Pilih AWS/MWAAnamespace.
4. Di tab Semua metrik, pilih dimensi. Misalnya, Cluster.
5. Pilih CloudWatch metrik untuk dimensi. Misalnya, NumSchedulersatau CPUUutilization. Kemudian, pilih Grafik semua hasil pencarian.
6. Pilih tab Metrik grafik untuk mengakses metrik kinerja.

Daftar metrik

Tabel berikut mencantumkan metrik layanan cluster, antrian, dan database untuk Amazon MWAA. Untuk mengakses deskripsi metrik yang dipancarkan langsung dari Amazon ECS, Amazon SQS, atau Amazon RDS, pilih tautan dokumentasi masing-masing.

Topik

- Metrik klaster
- Metrik basis data
- Metrik antrian
- Metrik Application Load Balancer

Metrik klaster

Metrik berikut berlaku untuk setiap penjadwal, pekerja dasar, pekerja tambahan, dan server web. Untuk informasi dan deskripsi selengkapnya tentang setiap metrik klaster, lihat Metrik dan dimensi yang tersedia di Panduan Pengembang Amazon ECS.

Namespace	Metrik	Unit
AWS/MWAA	CPUUtilization	Persen
AWS/MWAA	MemoryUtilization	Persen

Mengevaluasi jumlah pekerja tambahan dan wadah server web

Anda dapat menggunakan metrik komponen yang disediakan dalam dimensi Cluster, seperti yang dijelaskan dalam prosedur berikut, untuk menilai berapa banyak pekerja tambahan, atau server web, lingkungan yang digunakan pada titik waktu tertentu. Anda dapat melakukan ini dengan membuat grafik CPUUutilization atau MemoryUtilizationmetrik dan menyetel tipe statistik ke Hitungan Sampel. Nilai yang dihasilkan adalah jumlah total `RUNNING` tugas untuk `AdditionalWorker` komponen. Memahami jumlah contoh pekerja tambahan yang digunakan oleh lingkungan Anda dapat membantu Anda mengukur bagaimana skala lingkungan Anda dan dapat Anda gunakan untuk mengoptimalkan jumlah pekerja tambahan.

Workers

Untuk mengevaluasi jumlah pekerja tambahan yang menggunakan Konsol Manajemen AWS

1. Pilih AWS/MWAAnamespace.
2. Di tab Semua metrik, pilih dimensi Cluster.
3. Dalam dimensi Cluster, untuk AdditionalWorker, pilih CPUUutilization atau metrik. MemoryUtilization
4. Pada tab Metrik Grafik, atur Periode menjadi 1 Menit dan Statistik ke Hitungan Sampel.

webservers

Untuk mengevaluasi jumlah server web tambahan menggunakan Konsol Manajemen AWS

1. Pilih AWS/MWAAnamespace.
2. Di tab Semua metrik, pilih dimensi Cluster.
3. Dalam dimensi Cluster, untuk AdditionalWebservers, pilih CPUUutilization atau metrik. MemoryUtilization
4. Pada tab Metrik Grafik, atur Periode menjadi 1 Menit dan Statistik ke Hitungan Sampel.

Untuk informasi selengkapnya, lihat Penghitungan `RUNNING` tugas layanan di Panduan Pengembang Layanan Kontainer Elastis Amazon.

Metrik basis data

Metrik berikut berlaku untuk setiap instance database yang terkait dengan lingkungan Amazon MWAA.

Namespace	Metrik	Unit
AWS/MWAA	CPUUtilization	Persen
AWS/MWAA	DatabaseConnections	Hitungan
AWS/MWAA	DiskQueueDepth	Hitungan
AWS/MWAA	FreeableMemory	Byte
AWS/MWAA	VolumeWriteIOPS	Hitung per lima menit
AWS/MWAA	WriteIOPS	Hitungan per detik
AWS/MWAA	WriteLatency	Detik
AWS/MWAA	WriteThroughput	Byte per detik

Metrik antrian

Untuk informasi selengkapnya tentang unit dan deskripsi untuk metrik antrian berikut, lihat Metrik yang tersedia untuk CloudWatch Amazon SQS di Panduan Pengembang Layanan Antrian Sederhana Amazon.

Namespace	Metrik	Unit

Namespace	Metrik	Unit
AWS/MWAA	ApproximateAgeOfOldestTask	Detik
AWS/MWAA	RunningTasks	Hitungan
AWS/MWAA	QueuedTasks	Hitungan

Metrik Application Load Balancer

Metrik Application Load Balancer berlaku untuk server web yang berjalan di lingkungan Anda. Amazon MWAA menggunakan metrik ini untuk penskalaan server web Anda berdasarkan jumlah lalu lintas. Untuk informasi selengkapnya tentang unit dan deskripsi metrik penyeimbang beban berikut, lihat metrik untuk CloudWatch Application Load Balancer Anda di Panduan Pengguna Application Load Balancers.

Namespace	Metrik	Unit
AWS/MWAA	ActiveConnectionCount	Hitungan

Keamanan di Alur Kerja Terkelola Amazon untuk Apache Airflow

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara AWS dan Anda (pelanggan). Model tanggung jawab bersama menjelaskan hal ini sebagai keamanan dari cloud dan keamanan dalam cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan AWS layanan di AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Third-party auditor secara teratur menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari Program Kepatuhan Program AWS Kepatuhan Program AWS . Untuk mempelajari tentang program kepatuhan yang berlaku untuk Amazon MWAA, lihat AWS Layanan dalam Lingkup oleh AWS Layanan Program Kepatuhan dalam Lingkup oleh Program .
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, yang mencakup sensitivitas data Anda, persyaratan perusahaan Anda, serta undang-undang dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan Alur Kerja Terkelola Amazon untuk Apache Airflow. Gunakan untuk mengonfigurasi Amazon MWAA untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga mempelajari cara menggunakan AWS layanan lain yang membantu Anda memantau dan mengamankan sumber daya Amazon MWAA Anda.

Di bagian ini:

- Perlindungan Data di Alur Kerja Terkelola Amazon untuk Apache Airflow
- AWS Identity and Access Management
- Validasi Kepatuhan untuk Alur Kerja Terkelola Amazon untuk Apache Airflow
- Ketahanan dalam Alur Kerja Terkelola Amazon untuk Apache Airflow
- Keamanan Infrastruktur di Amazon MWAA
- Analisis Konfigurasi dan Kerentanan di Amazon MWAA
- Praktik terbaik keamanan di Amazon MWAA

Perlindungan Data di Alur Kerja Terkelola Amazon untuk Apache Airflow

Model tanggung jawab AWS bersama model tanggung jawab berlaku untuk perlindungan data di Alur Kerja Terkelola Amazon untuk Apache Airflow. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk menjaga kontrol atas konten yang dihosting di infrastruktur ini. Konten ini mencakup konfigurasi keamanan dan tugas manajemen untuk yang Layanan AWS Anda gunakan. Untuk informasi lebih lanjut tentang privasi data, lihat FAQ Privasi Data. Untuk informasi tentang perlindungan data di Eropa, lihat Model Tanggung Jawab AWS Bersama dan posting blog GDPR di Blog AWS Keamanan.

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensi dan menyiapkan akun pengguna individu dengan AWS Identity and Access Management (IAM). Dengan cara ini, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugas mereka. Kami juga merekomendasikan agar Anda mengamankan data Anda dengan cara-cara berikut ini:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan AWS sumber daya. Kami merekomendasikan TLS 1.2 atau versi yang lebih baru.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default dalam AWS layanan.
- Gunakan layanan keamanan terkelola lanjutan seperti Amazon Macie, yang membantu menemukan dan mengamankan data pribadi yang disimpan di Amazon S3.

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan Amazon MWAA atau AWS layanan lain menggunakan konsol, API AWS CLI, atau AWS SDK. Data apa pun yang Anda masukkan ke dalam tag atau kolom formulir bebas yang digunakan untuk nama dapat digunakan untuk penagihan atau log diagnostik. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan supaya Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

Enkripsi di Amazon MWAA

Topik berikut menjelaskan bagaimana Amazon MWAA melindungi data Anda saat istirahat, dan dalam perjalanan. Gunakan informasi ini untuk mempelajari cara Amazon MWAA terintegrasi dengan AWS KMS mengenkripsi data saat istirahat, dan cara data dienkripsi menggunakan protokol Transport Layer Security (TLS) dalam perjalanan.

Topik

- Enkripsi saat diam
- Enkripsi saat bergerak

Enkripsi saat diam

Di Amazon MWAA, data saat istirahat adalah data yang disimpan layanan ke media persisten.

Anda dapat menggunakan kunci yang AWS dimiliki untuk enkripsi data saat istirahat, atau secara opsional memberikan Customer-managed kunci untuk enkripsi tambahan saat Anda membuat lingkungan. Jika Anda memilih untuk menggunakan kunci KMS yang dikelola pelanggan, kunci tersebut harus berada di akun yang sama dengan AWS sumber daya dan layanan lain yang Anda gunakan dengan lingkungan Anda.

Untuk menggunakan kunci KMS yang dikelola pelanggan, Anda harus melampirkan pernyataan kebijakan yang diperlukan untuk CloudWatch akses ke kebijakan utama Anda. Saat Anda menggunakan kunci KMS yang dikelola pelanggan untuk lingkungan Anda, Amazon MWAA melampirkan empat hibah atas nama Anda. Untuk informasi selengkapnya tentang hibah Amazon MWAA yang dilampirkan ke kunci KMS yang dikelola pelanggan, lihat kunci untuk enkripsi data. Customer-managed

Jika Anda tidak menentukan kunci KMS yang dikelola pelanggan, secara default, Amazon MWAA menggunakan kunci KMS yang AWS dimiliki untuk mengenkripsi dan mendekripsi data Anda. Kami merekomendasikan menggunakan kunci KMS yang AWS dimiliki untuk mengelola enkripsi data di Amazon MWAA.

> **Note**
>
> Anda membayar penyimpanan dan penggunaan kunci KMS yang AWS dimiliki, atau dikelola pelanggan di Amazon MWAA. Untuk informasi lebih lanjut, lihat AWS KMS Harga.

Artefak enkripsi

Anda menentukan artefak enkripsi yang digunakan untuk enkripsi saat istirahat dengan menentukan kunci atau Customer-managed kunci yang AWS dimiliki saat Anda membuat lingkungan Amazon MWAA Anda. Amazon MWAA menambahkan hibah yang diperlukan ke kunci yang Anda tentukan.

Amazon S3 - Data Amazon S3 dienkripsi pada tingkat objek Server-Side menggunakan Enkripsi (SSE). Enkripsi dan dekripsi Amazon S3 berlangsung di bucket Amazon S3 tempat kode DAG dan file pendukung Anda disimpan. Objek dienkripsi saat diunggah ke Amazon S3 dan didekripsi saat diunduh ke lingkungan Amazon MWAA Anda. Secara default, jika Anda menggunakan kunci KMS yang dikelola pelanggan, Amazon MWAA menggunakannya untuk membaca dan mendekripsi data di bucket Amazon S3 Anda.

CloudWatch Log — Jika Anda menggunakan kunci KMS yang AWS dimiliki, log Apache Airflow yang dikirim CloudWatch ke Log dienkripsi menggunakan CloudWatch SSE dengan kunci KMS milik Log. AWS Jika Anda menggunakan kunci KMS yang dikelola pelanggan, Anda harus menambahkan kebijakan kunci ke kunci KMS Anda untuk mengizinkan CloudWatch Log menggunakan kunci Anda.

Amazon SQS - Amazon MWAA membuat satu antrian Amazon SQS untuk lingkungan Anda. Amazon MWAA menangani enkripsi data yang diteruskan ke dan dari antrian menggunakan SSE dengan kunci KMS yang AWS dimiliki, atau kunci KMS yang dikelola pelanggan yang Anda tentukan. Anda harus menambahkan izin Amazon SQS ke peran eksekusi terlepas dari apakah Anda menggunakan kunci KMS yang AWS dimiliki atau dikelola pelanggan.

Aurora PostgreSQL — Amazon MWAA membuat satu cluster PostgreSQL untuk lingkungan Anda. Aurora PostgreSQL mengenkripsi konten dengan kunci KMS yang dimiliki atau dikelola pelanggan menggunakan SSE. AWS Jika Anda menggunakan kunci KMS yang dikelola pelanggan, Amazon RDS menambahkan setidaknya dua hibah ke kunci: satu untuk cluster dan satu untuk instance database. Amazon RDS dapat membuat hibah tambahan jika Anda memilih untuk menggunakan kunci KMS yang dikelola pelanggan di beberapa lingkungan. Untuk informasi selengkapnya, lihat Perlindungan data di Amazon RDS.

Enkripsi saat bergerak

Data dalam perjalanan disebut sebagai data yang dapat dicegat saat melakukan perjalanan jaringan.

Transport Layer Security (TLS) mengenkripsi objek Amazon MWAA dalam perjalanan antara komponen Apache Airflow lingkungan Anda dan layanan lain AWS yang terintegrasi dengan Amazon MWAA, seperti Amazon S3. Untuk informasi selengkapnya tentang enkripsi Amazon S3, lihat Melindungi data menggunakan enkripsi.

Menggunakan kunci yang dikelola pelanggan untuk enkripsi

Anda secara opsional dapat memberikan Customer-managed kunci untuk enkripsi data di lingkungan Anda. Anda harus membuat kunci KMS yang dikelola pelanggan di Wilayah yang sama dengan instans lingkungan Amazon MWAA dan bucket Amazon S3 tempat Anda menyimpan sumber daya untuk alur kerja. Jika kunci KMS yang dikelola pelanggan yang Anda tentukan berada di akun yang berbeda dari yang Anda gunakan untuk mengonfigurasi lingkungan, Anda harus menentukan kunci menggunakan ARN untuk akses lintas akun. Untuk informasi selengkapnya tentang membuat kunci, lihat Membuat Kunci di Panduan AWS Key Management Service Pengembang.

Apa yang didukung

AWS KMS fitur	Didukung	
ID AWS KMS kunci atau ARN.	Ya	
Alias AWS KMS kunci.	Tidak	
Kunci AWS KMS multi-wil ayah.	Tidak	

Menggunakan Hibah untuk Enkripsi

Topik ini menjelaskan hibah Amazon MWAA yang dilampirkan ke kunci KMS yang dikelola pelanggan atas nama Anda untuk mengenkripsi dan mendekripsi data Anda.

Cara kerjanya

Ada dua mekanisme kontrol akses berbasis sumber daya yang didukung oleh AWS KMS kunci KMS yang dikelola pelanggan: kebijakan utama dan hibah.

Kebijakan kunci digunakan ketika izin sebagian besar statis dan digunakan dalam mode layanan sinkron. Hibah digunakan ketika izin yang lebih dinamis dan terperinci diperlukan, seperti ketika layanan perlu menentukan izin akses yang berbeda untuk dirinya sendiri atau akun lain.

Amazon MWAA menggunakan dan melampirkan empat kebijakan hibah ke kunci KMS yang dikelola pelanggan Anda. Hal ini disebabkan oleh izin granular yang diperlukan untuk lingkungan untuk

mengenkripsi data saat istirahat dari CloudWatch Log, antrian Amazon SQS, database database PostgreSQL Aurora, rahasia Secrets Manager, bucket Amazon S3, dan tabel DynamoDB.

Saat Anda membuat lingkungan Amazon MWAA dan menentukan kunci KMS yang dikelola pelanggan, Amazon MWAA melampirkan kebijakan hibah ke kunci KMS yang dikelola pelanggan. Kebijakan ini memungkinkan Amazon MWAA menggunakan kunci KMS yang dikelola pelanggan `airflow.us-east-1.amazonaws.com` untuk mengenkripsi sumber daya atas nama Anda yang dimiliki oleh Amazon MWAA.

Amazon MWAA membuat, dan melampirkan, hibah tambahan ke kunci KMS tertentu atas nama Anda. Ini termasuk kebijakan untuk menghentikan hibah jika Anda menghapus lingkungan Anda, untuk menggunakan kunci KMS yang dikelola pelanggan untuk Client-Side Enkripsi (CSE), dan untuk peran AWS Fargate eksekusi yang perlu mengakses rahasia yang dilindungi oleh kunci yang dikelola pelanggan Anda di Secrets Manager.

Kebijakan hibah

Amazon MWAA menambahkan hibah kebijakan berbasis sumber daya berikut atas nama Anda ke kunci KMS yang dikelola pelanggan. Kebijakan ini memungkinkan penerima hibah dan prinsipal (Amazon MWAA) untuk melakukan tindakan yang ditentukan dalam kebijakan.

Hibah 1: digunakan untuk membuat sumber daya bidang data

```
{
  "Name": "mwaa-grant-for-env-mgmt-role-environment name",
  "GranteePrincipal": "airflow.us-east-1.amazonaws.com",
  "RetiringPrincipal": "airflow.us-east-1.amazonaws.com",
  "Operations": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:CreateGrant",
    "kms:DescribeKey",
    "kms:RetireGrant"
  ]
}
```

Hibah 2: digunakan untuk **`ControllerLambdaExecutionRole`**akses

```
{
```

```
  "Name": "mwaa-grant-for-lambda-exec-environment name",
  "GranteePrincipal": "airflow.us-east-1.amazonaws.com",
  "RetiringPrincipal": "airflow.us-east-1.amazonaws.com",
  "Operations": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey",
    "kms:RetireGrant"
  ]
}
```

Hibah 3: digunakan untuk **`CfnManagementLambdaExecutionRole`**akses

```
{
  "Name": " mwaa-grant-for-cfn-mgmt-environment name",
  "GranteePrincipal": "airflow.us-east-1.amazonaws.com",
  "RetiringPrincipal": "airflow.us-east-1.amazonaws.com",
  "Operations": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ]
}
```

Grant 4: digunakan untuk peran eksekusi Fargate untuk mengakses rahasia backend

```
{
  "Name": "mwaa-fargate-access-for-environment name",
  "GranteePrincipal": "airflow.us-east-1.amazonaws.com",
  "RetiringPrincipal": "airflow.us-east-1.amazonaws.com",
  "Operations": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey",
    "kms:RetireGrant"
  ]
```

```
}
```

Melampirkan kebijakan utama ke kunci yang dikelola pelanggan

Jika Anda memilih untuk menggunakan kunci KMS yang dikelola pelanggan Anda sendiri dengan Amazon MWAA, Anda harus melampirkan kebijakan berikut ke kunci untuk mengizinkan Amazon MWAA menggunakannya untuk mengenkripsi data Anda.

Jika kunci KMS yang dikelola pelanggan yang Anda gunakan untuk lingkungan Amazon MWAA belum dikonfigurasi untuk berfungsi CloudWatch, Anda harus memperbarui kebijakan kunci untuk mengizinkan Log terenkripsi. CloudWatch Untuk informasi selengkapnya, lihat Enkripsi data log dalam CloudWatch menggunakan AWS Key Management Service layanan.

Contoh berikut merupakan kebijakan kunci untuk CloudWatch Log. Gantikan nilai sampel yang disediakan untuk wilayah tersebut.

```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "logs.us-east-1.amazonaws.com"
  },
  "Action": [
    "kms:Encrypt*",
    "kms:Decrypt*",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:Describe*"
  ],
  "Resource": "*",
  "Condition": {
    "ArnLike": {
      "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:us-east-1:*:*"
    }
  }
}
```

AWS Identity and Access Management

AWS Identity and Access Management (IAM) adalah AWS layanan yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. Administrator IAM mengontrol siapa yang

dapat diautentikasi (masuk) dan diberi wewenang (memiliki izin) untuk menggunakan Alur Kerja Terkelola Amazon untuk sumber daya Apache Airflow. IAM adalah AWS layanan yang dapat Anda gunakan tanpa biaya tambahan.

Topik ini memberikan gambaran dasar tentang bagaimana Amazon MWAA menggunakan AWS Identity and Access Management (IAM). Untuk mempelajari tentang mengelola akses ke Amazon MWAA, lihat. Mengelola akses ke lingkungan Amazon MWAA

Daftar Isi

- Audiens
- Mengautentikasi Menggunakan Identitas
- Mengelola Akses Menggunakan Kebijakan
- Memungkinkan pengguna untuk mengakses izin mereka sendiri
- Memecahkan Masalah Alur Kerja Terkelola Amazon untuk identitas dan akses Apache Airflow
- Bagaimana Amazon MWAA bekerja dengan IAM

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda berdasarkan peran Anda:

- Pengguna layanan - minta izin dari administrator Anda jika Anda tidak dapat mengakses fitur (lihat Memecahkan Masalah Alur Kerja Terkelola Amazon untuk identitas dan akses Apache Airflow)
- Administrator layanan - tentukan akses pengguna dan mengirimkan permintaan izin (lihat Bagaimana Amazon MWAA bekerja dengan IAM)
- Administrator IAM - tulis kebijakan untuk mengelola akses (lihat Contoh kebijakan berbasis identitas Amazon MWAA)

Mengautentikasi Menggunakan Identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensyal identitas Anda. Anda harus diautentikasi sebagai Pengguna root akun AWS, pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk sebagai identitas federasi menggunakan kredensyal dari sumber identitas seperti AWS IAM Identity Center (Pusat Identitas IAM), autentikasi masuk tunggal, atau kredensyal. Google/

Facebook Untuk informasi selengkapnya tentang cara masuk, lihat Cara masuk ke Akun AWS Anda dalam Panduan Pengguna AWS Sign-In .

Untuk akses terprogram, AWS sediakan SDK dan CLI untuk menandatangani permintaan secara kriptografis. Untuk informasi selengkapnya, lihat AWS Signature Version 4 untuk permintaan API dalam Panduan Pengguna IAM.

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang disebut pengguna Akun AWS root yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Untuk tugas yang memerlukan kredensial pengguna root, lihat Tugas yang memerlukan kredensial pengguna root dalam Panduan Pengguna IAM.

Pengguna dan Grup IAM

Pengguna IAM adalah identitas dengan izin khusus untuk satu orang atau aplikasi. Sebaiknya gunakan kredensial sementara alih-alih pengguna IAM dengan kredensial jangka panjang. Untuk informasi selengkapnya, lihat Mewajibkan pengguna manusia untuk menggunakan federasi dengan penyedia identitas untuk mengakses AWS menggunakan kredensyal sementara di Panduan Pengguna IAM.

Grup IAM menentukan kumpulan pengguna IAM dan mempermudah pengelolaan izin untuk pengguna dalam jumlah besar. Untuk mempelajari selengkapnya, lihat Kasus penggunaan untuk pengguna IAM dalam Panduan Pengguna IAM.

Peran IAM

Peran IAM adalah identitas dengan izin khusus yang menyediakan kredensial sementara. Anda dapat mengambil peran dengan beralih dari pengguna ke peran IAM (konsol) atau dengan memanggil operasi AWS CLI atau AWS API. Untuk informasi selengkapnya, lihat Metode untuk mengambil peran dalam Panduan Pengguna IAM.

Peran IAM berguna untuk akses pengguna terfederasi, izin pengguna IAM sementara, akses lintas akun, akses lintas layanan, dan aplikasi yang berjalan di Amazon EC2. Untuk informasi selengkapnya, lihat Akses sumber daya lintas akun di IAM dalam Panduan Pengguna IAM.

Mengelola Akses Menggunakan Kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan menentukan izin saat dikaitkan dengan identitas atau sumber daya. AWS mengevaluasi kebijakan ini ketika kepala sekolah membuat permintaan. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang dokumen kebijakan JSON, lihat Gambaran umum kebijakan JSON dalam Panduan Pengguna IAM.

Menggunakan kebijakan, administrator menentukan siapa yang memiliki akses ke apa dengan mendefinisikan principal mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Administrator IAM membuat kebijakan IAM dan menambahkannya ke peran, yang kemudian dapat diambil oleh pengguna. Kebijakan IAM mendefinisikan izin terlepas dari metode yang Anda gunakan untuk melakukan operasinya.

Identity-Based Kebijakan

Identity-based kebijakan adalah dokumen kebijakan izin JSON yang Anda lampirkan ke identitas (pengguna, grup, atau peran). Kebijakan ini mengontrol tindakan apa yang bisa dilakukan oleh identitas tersebut, terhadap sumber daya yang mana, dan dalam kondisi apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Tentukan izin IAM kustom dengan kebijakan yang dikelola pelanggan dalam Panduan Pengguna IAM.

Identity-based kebijakan dapat berupa kebijakan inline (disematkan langsung ke dalam satu identitas) atau kebijakan terkelola (kebijakan mandiri yang dilampirkan pada beberapa identitas). Untuk mempelajari cara memilih antara kebijakan terkelola dan kebijakan inline, lihat Pilih antara kebijakan terkelola dan kebijakan inline dalam Panduan Pengguna IAM.

Resource-Based Kebijakan

Resource-based kebijakan adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contohnya termasuk kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Anda harus menentukan principal dalam kebijakan berbasis sumber daya.

Resource-based kebijakan adalah kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola dari IAM dalam kebijakan berbasis sumber daya.

Daftar Kontrol Akses (ACL)

Daftar kontrol akses (ACL) mengendalikan principal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun kebijakan tersebut tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung ACL. Untuk mempelajari ACL selengkapnya, lihat Gambaran umum daftar kontrol akses (ACL) dalam Panduan Developer Amazon Simple Storage Service.

Tipe Kebijakan Lainnya

AWS mendukung jenis kebijakan tambahan yang dapat menetapkan izin maksimum yang diberikan oleh jenis kebijakan yang lebih umum:

- Batasan izin – Menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM. Untuk informasi selengkapnya, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCP) – Menentukan izin maksimum untuk organisasi atau unit organisasi di AWS Organizations. Untuk informasi selengkapnya, lihat Kebijakan kontrol layanan dalam Panduan Pengguna AWS Organizations .
- Kebijakan kontrol sumber daya (RCP) – Menetapkan izin maksimum yang tersedia untuk sumber daya di akun Anda. Untuk informasi selengkapnya, lihat Kebijakan kontrol sumber daya (RCP) dalam Panduan Pengguna AWS Organizations .
- Kebijakan sesi – Kebijakan lanjutan yang diteruskan sebagai parameter saat membuat sesi sementara untuk peran atau pengguna terfederasi. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Berbagai Tipe Kebijakan

Ketika beberapa jenis kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat Logika evaluasi kebijakan di Panduan Pengguna IAM.

Memungkinkan pengguna untuk mengakses izin mereka sendiri

Contoh ini menunjukkan cara membuat kebijakan yang mengizinkan pengguna IAM melihat kebijakan inline dan terkelola yang dilampirkan ke identitas pengguna mereka. Kebijakan ini

mencakup izin untuk menyelesaikan tindakan ini di konsol atau menggunakan API atau secara terprogram. AWS CLI 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": "*"
        }
    ]
}
```

Memecahkan Masalah Alur Kerja Terkelola Amazon untuk identitas dan akses Apache Airflow

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang dapat Anda temui saat bekerja dengan Amazon MWAA dan IAM.

Saya tidak berwenang untuk melakukan tindakan di Amazon MWAA

Jika Konsol Manajemen AWS memberitahu Anda bahwa Anda tidak berwenang untuk melakukan tindakan, maka Anda harus menghubungi administrator Anda untuk bantuan. Administrator Anda adalah orang yang memberikan nama pengguna dan kata sandi Anda.

Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan bahwa Anda tidak diizinkan untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran ke Amazon MWAA.

Beberapa Layanan AWS memungkinkan Anda untuk meneruskan peran yang ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait layanan. Untuk melakukannya, Anda harus memiliki izin untuk meneruskan peran ke layanan.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama `marymajor` mencoba menggunakan konsol untuk melakukan tindakan di Amazon MWAA. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui agar dia mendapatkan izin untuk melakukan tindakan `iam:PassRole` tersebut.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses sumber daya Amazon MWAA saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACL), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa referensi berikut:

- Untuk mengetahui apakah Amazon MWAA mendukung fitur-fitur ini, lihat. Bagaimana Amazon MWAA bekerja dengan IAM
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh sumber daya Akun AWS yang Anda miliki, lihat Menyediakan akses ke pengguna IAM di pengguna lain Akun AWS yang Anda miliki di Panduan Pengguna IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda kepada pihak ketiga Akun AWS, lihat Menyediakan akses yang Akun AWS dimiliki oleh pihak ketiga dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat Menyediakan akses ke pengguna terautentikasi eksternal (federasi identitas) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara menggunakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Akses sumber daya lintas akun di IAM di Panduan Pengguna IAM.

Bagaimana Amazon MWAA bekerja dengan IAM

Amazon MWAA menggunakan kebijakan berbasis identitas IAM untuk memberikan izin ke tindakan dan sumber daya Amazon MWAA. Untuk contoh kebijakan IAM kustom yang direkomendasikan yang dapat Anda gunakan untuk mengontrol akses ke sumber daya Amazon MWAA Anda, lihat. the section called “Mengakses lingkungan Amazon MWAA”

Untuk mendapatkan akses tingkat tinggi tentang cara Amazon MWAA dan AWS layanan lainnya bekerja dengan IAM, lihat AWS Layanan yang Bekerja dengan IAM di Panduan Pengguna IAM.

Kebijakan berbasis identitas Amazon MWAA

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan tindakan dan sumber daya yang diizinkan atau ditolak, serta kondisi di mana tindakan diizinkan atau ditolak. Amazon MWAA mendukung tindakan, sumber daya, dan kunci kondisi tertentu.

Langkah-langkah berikut menunjukkan bagaimana Anda dapat membuat kebijakan JSON baru menggunakan konsol IAM. Kebijakan ini menyediakan akses hanya-baca ke sumber daya Amazon MWAA Anda.

Cara menggunakan editor kebijakan JSON untuk membuat kebijakan

1. Masuk ke Konsol Manajemen AWS dan buka konsol IAM di https://console.aws.amazon.com/iam/.

2. Pada panel navigasi di sebelah kiri, pilih Kebijakan.

   Jika ini pertama kalinya Anda memilih Kebijakan, akan muncul halaman Selamat Datang di Kebijakan Terkelola. Pilih Memulai.

3. Di bagian atas halaman, pilih Buat kebijakan.
4. Di bagian Editor kebijakan, pilih opsi JSON.
5. Masukkan dokumen kebijakan JSON berikut:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "airflow:ListEnvironments",
        "airflow:GetEnvironment",
        "airflow:ListTagsForResource"
      ],
      "Resource": "*"
    }
  ]
}
```

6. Pilih Berikutnya.

> **Note**
>
> Anda dapat beralih antara opsi editor Visual dan JSON kapan saja. Namun, jika Anda melakukan perubahan atau memilih Berikutnya di editor Visual, IAM dapat merestrukturisasi kebijakan Anda untuk mengoptimalkannya bagi editor visual. Untuk informasi selengkapnya, lihat Restrukturisasi kebijakan dalam Panduan Pengguna IAM.

7. Pada halaman Tinjau dan buat, masukkan Nama kebijakan dan Deskripsi (opsional) untuk kebijakan yang Anda buat. Tinjau Izin yang ditentukan dalam kebijakan ini untuk melihat izin yang diberikan oleh kebijakan Anda.
8. Pilih Buat kebijakan untuk menyimpan kebijakan baru Anda.

Untuk mempelajari semua elemen yang Anda gunakan dalam kebijakan JSON, lihat Referensi Elemen Kebijakan IAM JSON di Panduan Pengguna IAM.

Tindakan

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, di mana utama dapat melakukan tindakan pada sumber daya, dan dalam kondisi apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Sertakan tindakan dalam kebijakan untuk memberikan izin untuk melakukan operasi terkait.

Pernyataan kebijakan harus mencakup `Action` elemen atau `NotAction` elemen. `Action`Elemen mencantumkan tindakan yang diizinkan oleh kebijakan. `NotAction`Elemen mencantumkan tindakan yang tidak diizinkan.

Tindakan yang ditentukan untuk Amazon MWAA mencerminkan tugas yang dapat Anda lakukan menggunakan Amazon MWAA. Tindakan kebijakan di Detektif memiliki awalan berikut:. `airflow:`

Anda juga dapat menggunakan wildcard (*) untuk menentukan beberapa tindakan. Alih-alih mencantumkan tindakan ini secara terpisah, Anda dapat memberikan akses ke semua tindakan yang diakhiri dengan kata, misalnya,`environment`.

Untuk mendapatkan daftar tindakan Amazon MWAA, lihat Tindakan yang Ditentukan oleh Alur Kerja Terkelola Amazon untuk Apache Airflow di Panduan Pengguna IAM.

Contoh kebijakan berbasis identitas Amazon MWAA

Untuk mengakses kebijakan Amazon MWAA, lihat. Mengelola akses ke lingkungan Amazon MWAA

Secara default, pengguna dan peran IAM tidak memiliki izin untuk membuat atau memodifikasi sumber daya Amazon MWAA. Mereka juga tidak dapat melakukan tugas menggunakan Konsol Manajemen AWS, AWS CLI, atau AWS API.

Administrator IAM harus membuat kebijakan IAM yang memberikan izin kepada pengguna dan peran untuk melakukan operasi API tertentu pada sumber daya yang diperlukan. Administrator kemudian melampirkan kebijakan tersebut ke pengguna IAM atau grup yang memerlukan izin tersebut.

> **Important**
>
> Sebaiknya gunakan peran IAM dan kredensyal sementara untuk menyediakan akses ke sumber daya Amazon MWAA Anda. Menghindari melampirkan poicies izin langsung ke pengguna IAM Anda.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM menggunakan contoh dokumen kebijakan JSON ini, lihat Membuat Kebijakan di Tab JSON di Panduan Pengguna IAM.

Topik

- Praktik terbaik kebijakan
- Menggunakan konsol Amazon MWAA
- Memungkinkan pengguna untuk mengakses izin mereka sendiri

Praktik terbaik kebijakan

Identity-based kebijakan menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya Amazon MWAA di akun Anda. Tindakan ini membuat Akun AWS Anda dikenai biaya. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulailah dengan kebijakan AWS terkelola dan beralih ke izin hak istimewa paling sedikit — Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan AWS terkelola yang memberikan izin untuk banyak kasus penggunaan umum. Mereka tersedia di Anda Akun AWS. Kami menyarankan Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola AWS pelanggan yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat Kebijakan yang dikelola AWS atau Kebijakan yang dikelola AWS untuk fungsi tugas dalam Panduan Pengguna IAM.
- Menerapkan izin dengan hak akses paling rendah – Ketika Anda menetapkan izin dengan kebijakan IAM, hanya berikan izin yang diperlukan untuk melakukan tugas. Anda melakukannya dengan mendefinisikan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, yang juga dikenal sebagai izin dengan hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan IAM untuk mengajukan izin, lihat Kebijakan dan izin dalam IAM dalam Panduan Pengguna IAM.
- Gunakan kondisi dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan suatu kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Sebagai contoh, Anda dapat menulis kondisi kebijakan untuk menentukan bahwa semua permintaan harus dikirim menggunakan SSL. Anda juga dapat menggunakan ketentuan untuk memberikan akses ke tindakan layanan jika digunakan melalui yang spesifik Layanan AWS, seperti CloudFormation. Untuk informasi selengkapnya, lihat Elemen kebijakan JSON IAM: Kondisi dalam Panduan Pengguna IAM.

- Gunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda untuk memastikan izin yang aman dan fungsional – IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan tersebut mematuhi bahasa kebijakan IAM (JSON) dan praktik terbaik IAM. IAM Access Analyzer menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk informasi selengkapnya, lihat Validasi kebijakan dengan IAM Access Analyzer dalam Panduan Pengguna IAM.
- Memerlukan otentikasi multi-faktor (MFA) - Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di Anda, Akun AWS aktifkan MFA untuk keamanan tambahan. Untuk meminta MFA ketika operasi API dipanggil, tambahkan kondisi MFA pada kebijakan Anda. Untuk informasi selengkapnya, lihat Amankan akses API dengan MFA dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat Praktik terbaik keamanan di IAM dalam Panduan Pengguna IAM.

Menggunakan konsol Amazon MWAA

Untuk menggunakan konsol Amazon MWAA, pengguna atau peran harus memiliki akses ke tindakan yang relevan, yang cocok dengan tindakan terkait di API.

Untuk mengakses kebijakan Amazon MWAA, lihat. Mengelola akses ke lingkungan Amazon MWAA

Memungkinkan pengguna untuk mengakses izin mereka sendiri

Contoh ini menunjukkan cara membuat kebijakan yang mengizinkan pengguna IAM melihat kebijakan inline dan terkelola yang dilampirkan ke identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan ini di konsol atau menggunakan API atau secara terprogram. AWS CLI 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": "*"
        }
    ]
}
```

Validasi Kepatuhan untuk Alur Kerja Terkelola Amazon untuk Apache Airflow

Untuk mempelajari apakah an Layanan AWS berada dalam lingkup program kepatuhan tertentu, lihat Layanan AWS di Lingkup oleh Program Kepatuhan Layanan AWS dan pilih program kepatuhan yang Anda minati. Untuk informasi umum, lihat Program AWS Kepatuhan Program AWS .

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifact .

Tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. Untuk informasi selengkapnya tentang tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS, lihat Dokumentasi AWS Keamanan.

Ketahanan dalam Alur Kerja Terkelola Amazon untuk Apache Airflow

Infrastruktur AWS global dibangun di sekitar Wilayah AWS dan Availability Zones. Wilayah memberikan beberapa Zona Ketersediaan yang terpisah dan terisolasi secara fisik, yang terkoneksi melalui jaringan latensi rendah, throughput tinggi, dan sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang serta mengoperasikan aplikasi dan basis data yang secara otomatis melakukan fail over di antara zona tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur pusat data tunggal atau multi tradisional.

Untuk informasi selengkapnya tentang Wilayah AWS dan Availability Zone, lihat Infrastruktur AWS Global.

Keamanan Infrastruktur di Amazon MWAA

Sebagai layanan terkelola, Alur Kerja Terkelola Amazon untuk Apache Airflow dilindungi oleh AWS keamanan jaringan global. Untuk informasi tentang layanan AWS keamanan dan cara AWS melindungi infrastruktur, lihat Keamanan AWS Cloud. Untuk mendesain AWS lingkungan Anda menggunakan praktik terbaik untuk keamanan infrastruktur, lihat Perlindungan Infrastruktur dalam Kerangka Kerja yang AWS Diarsiteksikan dengan Baik Pilar Keamanan.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses Amazon MWAA melalui jaringan. Klien harus mendukung hal-hal berikut:

- Keamanan Lapisan Pengangkutan (TLS). Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Cipher suite dengan perfect forward secrecy (PFS) seperti DHE (Ephemeral) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Diffie-Hellman Sebagian besar sistem modern seperti Java 7 dan versi lebih baru mendukung mode-mode ini.

Analisis Konfigurasi dan Kerentanan di Amazon MWAA

Konfigurasi dan kontrol TI adalah tanggung jawab bersama antara AWS dan Anda, pelanggan kami.

Alur Kerja Terkelola Amazon untuk Apache Airflow secara berkala menambal dan meningkatkan Apache Airflow di lingkungan Anda. Pastikan bahwa kebijakan akses yang sesuai digunakan untuk VPC Anda.

Untuk detail selengkapnya, lihat sumber daya berikut:

- Validasi Kepatuhan untuk Alur Kerja Terkelola Amazon untuk Apache Airflow
- Model Tanggung Jawab Bersama
- Amazon Web Services: Gambaran Umum Proses Keamanan
- Keamanan Infrastruktur di Amazon MWAA
- Praktik terbaik keamanan di Amazon MWAA

Praktik terbaik keamanan di Amazon MWAA

Amazon MWAA menyediakan sejumlah fitur keamanan untuk dipertimbangkan saat Anda mengembangkan dan menerapkan kebijakan keamanan Anda sendiri. Praktik terbaik berikut adalah pedoman umum dan tidak mewakili solusi keamanan yang lengkap. Karena praktik terbaik ini mungkin tidak sesuai atau tidak memadai untuk lingkungan Anda, perlakukan itu sebagai pertimbangan yang bermanfaat, bukan sebagai resep.

- Gunakan kebijakan izin yang paling tidak permisif. Berikan izin hanya pada sumber daya atau tindakan yang dibutuhkan pengguna untuk melakukan tugas.
- Gunakan AWS CloudTrail untuk memantau aktivitas pengguna di akun Anda.
- Pastikan kebijakan bucket Amazon S3 dan ACL objek memberikan izin kepada pengguna dari lingkungan Amazon MWAA terkait untuk memasukkan objek ke dalam bucket. Ini memastikan bahwa pengguna dengan izin untuk menambahkan alur kerja ke bucket juga memiliki izin untuk menjalankan alur kerja di Airflow.
- Gunakan bucket Amazon S3 yang terkait dengan lingkungan Amazon MWAA. Bucket Amazon S3 Anda bisa berupa nama apa saja. Jangan menyimpan benda lain di ember, atau gunakan ember dengan layanan lain.

Praktik terbaik keamanan di Apache Airflow

Apache Airflow bukan multi-tenant. Meskipun ada langkah-langkah kontrol akses untuk membatasi beberapa fitur untuk pengguna tertentu, yang diterapkan Amazon MWAA, pembuat DAG memang memiliki kemampuan untuk menulis DAG yang dapat mengubah hak pengguna Apache Airflow dan berinteraksi dengan metadatabase yang mendasarinya.

Kami merekomendasikan langkah-langkah berikut saat bekerja dengan Apache Airflow di Amazon MWAA untuk memastikan metadatabase dan DAG lingkungan Anda aman.

- Gunakan lingkungan terpisah untuk tim terpisah dengan akses penulisan DAG, atau kemampuan untuk menambahkan file ke `/dags` folder Amazon S3 Anda, dengan asumsi apa pun yang dapat diakses oleh Peran Eksekusi Amazon MWAA atau koneksi Apache Airflow juga akan dapat diakses oleh pengguna yang dapat menulis ke lingkungan.
- Jangan berikan akses folder Amazon S3 DAG langsung. Sebagai gantinya, gunakan CI/CD alat untuk menulis DAG ke Amazon S3, dengan langkah validasi yang memastikan bahwa kode DAG memenuhi pedoman keamanan tim Anda.
- Cegah akses pengguna ke bucket Amazon S3 lingkungan Anda. Sebagai gantinya, gunakan pabrik DAG yang menghasilkan DAG berdasarkan YAMAL, JSON, atau file definisi lain yang disimpan di lokasi terpisah dari bucket Amazon S3 Amazon MWAA Amazon S3 tempat Anda menyimpan DAG.
- Simpan rahasia di Secrets Manager. Meskipun ini tidak akan mencegah pengguna yang dapat menulis DAG dari membaca rahasia, ini akan mencegah mereka memodifikasi rahasia yang digunakan lingkungan Anda.

Mendeteksi perubahan pada hak istimewa pengguna Apache Airflow

Anda dapat menggunakan Wawasan CloudWatch Log untuk mendeteksi kejadian DAG yang mengubah hak istimewa pengguna Apache Airflow. Untuk melakukannya, Anda dapat menggunakan aturan EventBridge terjadwal, fungsi Lambda, dan Wawasan CloudWatch Log untuk mengirimkan notifikasi ke CloudWatch metrik setiap kali salah satu DAG Anda mengubah hak istimewa pengguna Apache Airflow.

Prasyarat

Untuk menyelesaikan langkah-langkah berikut, Anda memerlukan yang berikut:

- Lingkungan Amazon MWAA dengan semua jenis log Apache Airflow diaktifkan pada tingkat log. `INFO` Untuk informasi lebih lanjut, lihatthe section called “Mengakses log Aliran Udara”.

Untuk mengonfigurasi notifikasi untuk perubahan hak istimewa pengguna Apache Airflow

1. Buat fungsi Lambda yang menjalankan string kueri CloudWatch Log Insights berikut terhadap lima grup log lingkungan Amazon MWAA (`DAGProcessing`,,,`Scheduler`, `Task` dan). `WebServer` `Worker`

```
fields @log, @timestamp, @message | filter @message like "add-role" | stats count()
 by @log
```

2. Buat EventBridge aturan yang berjalan sesuai jadwal, dengan fungsi Lambda yang Anda buat di langkah sebelumnya sebagai target aturan. Konfigurasikan jadwal Anda menggunakan ekspresi cron atau rate untuk dijalankan secara berkala.

Versi Apache Airflow di Amazon Managed Workflows for Apache Airflow

Topik ini menjelaskan versi Apache Airflow Amazon Managed Workflows untuk Apache Airflow mendukung, dan praktik terbaik untuk meningkatkan ke versi terbaru.

Topik

- Tentang versi Amazon MWAA
- Versi terbaru
- Versi Apache Airflow
- Komponen Apache Airflow
- Memutakhirkan versi Apache Airflow
- Menurunkan versi Apache Airflow
- Apache Airflow versi usang
- Dukungan dan FAQ versi Apache Airflow

Tentang versi Amazon MWAA

Amazon MWAA membuat gambar kontainer yang menggabungkan rilis Apache Airflow dengan binari umum lainnya dan pustaka Python. Gambar menggunakan instalasi dasar Apache Airflow untuk versi yang Anda tentukan. Saat membuat lingkungan, Anda menentukan versi gambar yang akan digunakan. Setelah lingkungan dibuat, ia terus menggunakan versi gambar yang ditentukan sampai Anda memutakhirkannya ke versi yang lebih baru.

Versi terbaru

Amazon MWAA mendukung lebih dari satu versi Apache Airflow. Jika Anda tidak menentukan versi gambar saat membuat lingkungan, Amazon MWAA membuat lingkungan menggunakan Apache Airflow versi terbaru yang didukung.

Versi Apache Airflow

Versi Apache Airflow berikut didukung di Alur Kerja Terkelola Amazon untuk Apache Airflow.

Note

- Efektif 30 Desember 2025, Amazon MWAA akan mengakhiri dukungan untuk Apache Airflow versi v2.4.3, v2.5.1, dan v2.6.3. Untuk informasi lebih lanjut, lihatDukungan dan FAQ versi Apache Airflow.
- Dimulai dengan Apache Airflow v2.2.2, Amazon MWAA mendukung pemasangan persyaratan Python, paket penyedia, dan plugin khusus langsung di server web Apache Airflow.
- Dimulai dengan Apache Airflow v2.7.2, file persyaratan Anda harus menyertakan pernyataan. `--constraint` Jika Anda tidak memberikan batasan, Amazon MWAA akan menentukan satu untuk Anda untuk memastikan paket yang tercantum dalam persyaratan Anda kompatibel dengan versi Apache Airflow yang Anda gunakan.

  Untuk informasi selengkapnya tentang pengaturan batasan dalam file persyaratan Anda, lihat Menginstal dependensi Python.

Versi Apache Airflow	Tanggal rilis Apache Airflow	Tanggal ketersediaan Amazon MWAA	Kendala Apache Airflow	Versi Python
v3.2.1	April 22, 2026	19 Mei 2026	v3.2.1 kendala file	Python 3.12
v2.11.0	Mei 20, 2025	Januari 7, 2026	v2.11.0 kendala file	Python 3.12
v3.0.6	Agustus 29, 2025	Oktober 1, 2025	v3.0.6 kendala file	Python 3.12
v2.10.3	November 4, 2024	Desember 18, 2024	v2.10.3 kendala file	Python 3.11
v2.10.1	September 5, 2024	September 26, 2024	v2.10.1 kendala file	Python 3.11

Versi Apache Airflow	Tanggal rilis Apache Airflow	Tanggal ketersediaan Amazon MWAA	Kendala Apache Airflow	Versi Python
v2.9.2	10 Juni 2024	9 Juli 2024	v2.9.2 kendala file	Python 3.11
v2.8.1	Januari 19, 2024	Februari 23, 2024	v2.8.1 kendala file	Python 3.11
v2.7.2	Oktober 12, 2023	6 November 2023	v2.7.2 kendala file	Python 3.11

Untuk informasi selengkapnya tentang memigrasi penerapan Apache Airflow yang dikelola sendiri, atau memigrasi lingkungan Amazon MWAA yang ada, termasuk petunjuk untuk membuat cadangan basis data metadata Anda, lihat Panduan Migrasi Amazon MWAA.

Komponen Apache Airflow

Bagian ini menjelaskan jumlah penjadwal dan pekerja Apache Airflow yang tersedia untuk setiap versi Apache Airflow di Amazon MWAA, dan menyediakan daftar fitur utama Apache Airflow, yang menunjukkan versi yang mendukung setiap fitur.

Penjadwal

Penjadwal untuk Apache Airflow v2 dan yang lebih baru:

Penjadwal (default)	Penjadwal (min)	Penjadwal (maks)
2	2	5

Pekerja

Pekerja untuk Apache Airflow v2 dan yang lebih baru:

Pekerja (default)	Pekerja (min)	Pekerja (maks)
10	1	25

Memutakhirkan versi Apache Airflow

Amazon MWAA mendukung peningkatan versi minor. Ini berarti Anda dapat meningkatkan lingkungan Anda dari versi x**.1.**z kex**.2.**z, tetapi tidak ke versi utama baru, misalnya, dari **1.**y**.**z ke**2.**y**.**z.

Untuk informasi selengkapnya, dan petunjuk terperinci tentang memperbarui sumber daya alur kerja Anda, dan meningkatkan lingkungan ke versi baru, lihat. the section called “Mengubah versi”

Menurunkan versi Apache Airflow

Amazon MWAA mendukung penurunan versi minor ke versi sebelumnya yang masih didukung pada saat downgrade. Ini berarti Anda dapat menurunkan versi lingkungan Anda dari versi x**.2.**z kex**.1.**z, tetapi tidak ke versi utama sebelumnya, misalnya, dari **2.**y**.**z ke**1.**y**.**z.

Untuk informasi selengkapnya, dan petunjuk terperinci tentang memperbarui sumber daya alur kerja Anda, dan meningkatkan lingkungan ke versi baru, lihat. the section called “Mengubah versi”

Apache Airflow versi usang

Tabel berikut mencantumkan versi Apache Airflow yang tidak digunakan lagi di Amazon MWAA, bersama dengan rilis awal dan tanggal akhir dukungan untuk setiap versi. Untuk informasi selengkapnya tentang migrasi ke versi yang lebih baru, lihat Panduan Migrasi Amazon MWAA.

Versi Apache Airflow	Tanggal rilis Apache Airflow	Tanggal ketersediaan Amazon MWAA	Tanggal akhir dukungan Amazon MWAA
v1.10.12	25 Agustus 2020	24 November 2020	Februari 21, 2024
v2.0.2	19 April 2021	25 Mei 2021	April 29, 2024

Versi Apache Airflow	Tanggal rilis Apache Airflow	Tanggal ketersediaan Amazon MWAA	Tanggal akhir dukungan Amazon MWAA
v2.2.2	15 November 2021	Januari 27,2022	27 Juni 2024
v2.4.3	November 14, 2022	Januari 05, 2023	Desember 30, 2025
v2.5.1	20 Januari 2023	11 April 2023	Desember 30, 2025
v2.6.3	Juli 10, 2023	Agustus 09, 2023	Desember 30, 2025

Dukungan dan FAQ versi Apache Airflow

Sesuai dengan proses rilis komunitas Apache Airflow dan kebijakan versi, Amazon MWAA berkomitmen untuk mendukung setidaknya tiga versi minor Apache Airflow pada waktu tertentu. Kami akan mengumumkan akhir tanggal dukungan dari versi minor Apache Airflow yang diberikan setidaknya 180 hari sebelum akhir tanggal dukungan.

Pertanyaan umum

T: Berapa lama Amazon MWAA mendukung versi Apache Airflow?

J: Amazon MWAA mendukung versi patch Apache Airflow selama minimal 12 bulan setelah pertama kali tersedia.

T: Apakah saya diberi tahu ketika dukungan berakhir untuk versi Apache Airflow di Amazon MWAA?

J: Ya. Jika ada lingkungan Amazon MWAA di akun Anda yang menjalankan versi mendekati akhir dukungan, Amazon MWAA mengirimkan pemberitahuan melalui Dasbor Health dengan akhir tanggal dukungan.

T: Apa yang terjadi pada tanggal akhir dukungan?

J: Di akhir tanggal dukungan, Anda tidak dapat lagi menggunakan versi usang untuk membuat lingkungan Amazon MWAA baru. Anda dapat terus mengakses lingkungan Amazon MWAA yang ada yang menjalankan versi Apache Airflow yang terkait dan tidak digunakan lagi dengan risiko Anda sendiri. Untuk meningkatkan ke versi Apache Airflow yang lebih baru di Amazon MWAA, lihat Panduan Migrasi Amazon MWAA.

Important

Anda bertanggung jawab untuk menjaga versi Amazon MWAA Anda tetap terkini. AWS mendesak semua pelanggan untuk meningkatkan lingkungan Amazon MWAA mereka ke versi terbaru untuk mendapatkan manfaat dari perlindungan keamanan, privasi, dan ketersediaan terkini. Jika Anda mengoperasikan lingkungan Anda pada versi atau perangkat lunak yang tidak didukung setelah tanggal penghentian, yang disebut sebagai versi lama, Anda menghadapi kemungkinan risiko keamanan, privasi, dan operasional yang lebih besar, termasuk peristiwa downtime. Dengan mengoperasikan lingkungan Amazon MWAA Anda pada versi lama, Anda mengonfirmasi bahwa Anda memahami dan secara sadar menanggung risiko ini, dan Anda setuju untuk menyelesaikan peningkatan ke versi terbaru sesegera mungkin. Pengoperasian lingkungan Anda yang berkelanjutan pada versi lama tunduk pada perjanjian yang mengatur penggunaan layanan oleh Anda. AWS
Versi lama tidak dianggap tersedia secara umum, dan AWS tidak lagi menyediakan dukungan untuk versi lama. Akibatnya, AWS dapat membatasi akses atau penggunaan versi lama kapan saja, jika AWS menentukan bahwa versi lama menimbulkan risiko keamanan atau kewajiban, atau risiko bahaya, terhadap layanan, Afiliasinya AWS, atau pihak ketiga lainnya. Keputusan Anda untuk terus menjalankan beban kerja Anda pada versi lama dapat mengakibatkan konten Anda menjadi tidak tersedia, rusak, atau tidak dapat dipulihkan.
Lingkungan yang berjalan pada versi lama tunduk pada pengecualian Perjanjian Tingkat Layanan (SLA).
Lingkungan, dan perangkat lunak terkait, yang berjalan pada versi lama mungkin mengandung bug, kesalahan, cacat, dan komponen berbahaya. Dengan demikian, dan terlepas dari informasi apa pun yang bertentangan dalam perjanjian, atau ketentuan layanan, AWS memberikan versi warisan apa adanya.
Untuk informasi lebih lanjut tentang AWS model tanggung jawab bersama, lihat Tanggung jawab bersama dalam AWS Well-Architected Kerangka Kerja.

Alur Kerja Terkelola Amazon untuk titik akhir dan kuota layanan Apache Airflow

Alur Kerja Terkelola Amazon untuk Apache Airflow memiliki kuota layanan dan titik akhir berikut. Kuota layanan, juga disebut sebagai batas, adalah jumlah maksimum sumber daya layanan atau operasi untuk Anda Akun AWS.

Daftar Isi

- Titik akhir layanan
- Kuota layanan
- Meningkatkan kuota

Titik akhir layanan

Untuk mengakses daftar titik akhir Amazon MWAA, lihat Alur Kerja Terkelola Amazon untuk titik akhir dan kuota Apache Airflow.

Kuota layanan

Nama kuota	Deskripsi	Kuota default	Dapat Disesuaikan
Lingkungan	Jumlah maksimum lingkungan Amazon MWAA per akun per Wilayah.	10	Ya
Pekerja per lingkungan	Jumlah maksimum pekerja per lingkungan Amazon MWAA.	25	Ya
Server web per lingkungan	Jumlah maksimum server web per lingkungan Amazon MWAA.	5	Ya

Meningkatkan kuota

Anda dapat meminta kenaikan ke kuota yang dapat disesuaikan dengan mengirimkan permintaan kenaikan kuota.

Amazon MWAA pertanyaan yang sering diajukan

Halaman ini menjelaskan pertanyaan umum yang dapat Anda temui saat menggunakan Alur Kerja Terkelola Amazon untuk Apache Airflow.

Daftar Isi

- Versi yang didukung
  - Apa yang didukung Amazon MWAA untuk Apache Airflow v2?
  - Versi Python apa yang bisa saya gunakan?
- Kasus penggunaan
  - Bisakah saya menggunakan Amazon MWAA dengan Amazon SageMaker Unified Studio?
  - Kapan saya bisa menggunakan AWS Step Functions vs Amazon MWAA?
- Spesifikasi lingkungan
  - Berapa banyak penyimpanan tugas yang tersedia untuk setiap lingkungan?
  - Apa sistem operasi default yang digunakan untuk lingkungan Amazon MWAA?
  - Dapatkah saya menggunakan gambar khusus untuk lingkungan Amazon MWAA saya?
  - Apakah Amazon MWAA HIPAA sesuai?
  - Apakah Amazon MWAA mendukung Instans Spot?
  - Apakah Amazon MWAA mendukung domain khusus?
  - Bisakah saya SSH ke lingkungan saya?
  - Mengapa aturan referensi mandiri diperlukan pada grup keamanan VPC?
  - Dapatkah saya menyembunyikan lingkungan dari grup yang berbeda di IAM?
  - Dapatkah saya menyimpan data sementara pada pekerja Apache Airflow?
  - Bisakah saya menentukan lebih dari 25 pekerja Apache Airflow?
  - Apakah Amazon MWAA mendukung VPC Amazon bersama atau subnet bersama?
  - Dapatkah saya membuat atau mengintegrasikan antrian Amazon SQS khusus untuk mengelola eksekusi tugas dan orkestrasi alur kerja di Apache Airflow?
- Metrik-metrik
  - Metrik apa yang digunakan untuk menentukan apakah akan menskalakan pekerja?
  - Bisakah saya membuat metrik khusus? CloudWatch
- DAG, Operator, Koneksi, dan pertanyaan lainnya

- Dapatkah saya menggunakan PythonVirtualenvOperator?
- Berapa lama waktu yang dibutuhkan Amazon MWAA untuk mengenali file DAG baru?
- Mengapa file DAG saya tidak diambil oleh Apache Airflow?
- Bisakah saya menghapus plugins.zip atau requirements.txt dari lingkungan?
- Mengapa plugin saya tidak muncul di menu Apache Airflow v2.0.2 Admin Plugins?
- Dapatkah saya menggunakan AWS Operator Database Migration Service (DMS)?
- Saat saya mengakses Airflow REST API menggunakan AWS kredensialnya, dapatkah saya meningkatkan batas throttling menjadi lebih dari 10 transaksi per detik (TPS)?
- Di mana server API Eksekusi Tugas Airflow berjalan di Amazon MWAA?

Versi yang didukung

Apa yang didukung Amazon MWAA untuk Apache Airflow v2?

Untuk mempelajari apa yang didukung Amazon MWAA, lihat. Versi Apache Airflow di Amazon Managed Workflows for Apache Airflow

Versi Python apa yang bisa saya gunakan?

Versi Apache Airflow berikut didukung di Alur Kerja Terkelola Amazon untuk Apache Airflow.

> **Note**
>
> - Efektif 30 Desember 2025, Amazon MWAA akan mengakhiri dukungan untuk Apache Airflow versi v2.4.3, v2.5.1, dan v2.6.3. Untuk informasi lebih lanjut, lihatDukungan dan FAQ versi Apache Airflow.
> - Dimulai dengan Apache Airflow v2.2.2, Amazon MWAA mendukung pemasangan persyaratan Python, paket penyedia, dan plugin khusus langsung di server web Apache Airflow.
> - Dimulai dengan Apache Airflow v2.7.2, file persyaratan Anda harus menyertakan pernyataan. `--constraint` Jika Anda tidak memberikan batasan, Amazon MWAA akan menentukan satu untuk Anda untuk memastikan paket yang tercantum dalam persyaratan Anda kompatibel dengan versi Apache Airflow yang Anda gunakan.

Untuk informasi selengkapnya tentang pengaturan batasan dalam file persyaratan Anda, lihat Menginstal dependensi Python.

Versi Apache Airflow	Tanggal rilis Apache Airflow	Tanggal ketersediaan Amazon MWAA	Kendala Apache Airflow	Versi Python
v3.2.1	April 22, 2026	19 Mei 2026	v3.2.1 kendala file	Python 3.12
v2.11.0	Mei 20, 2025	Januari 7, 2026	v2.11.0 kendala file	Python 3.12
v3.0.6	Agustus 29, 2025	Oktober 1, 2025	v3.0.6 kendala file	Python 3.12
v2.10.3	November 4, 2024	Desember 18, 2024	v2.10.3 kendala file	Python 3.11
v2.10.1	September 5, 2024	September 26, 2024	v2.10.1 kendala file	Python 3.11
v2.9.2	Juni 10, 2024	9 Juli 2024	v2.9.2 kendala file	Python 3.11
v2.8.1	Januari 19, 2024	Februari 23, 2024	v2.8.1 kendala file	Python 3.11
v2.7.2	Oktober 12, 2023	6 November 2023	v2.7.2 kendala file	Python 3.11

Untuk informasi selengkapnya tentang memigrasi penerapan Apache Airflow yang dikelola sendiri, atau memigrasi lingkungan Amazon MWAA yang ada, termasuk petunjuk untuk membuat cadangan basis data metadata Anda, lihat Panduan Migrasi Amazon MWAA.

Kasus penggunaan

Bisakah saya menggunakan Amazon MWAA dengan Amazon SageMaker Unified Studio?

Ya. Dengan alur kerja Amazon SageMaker Unified Studio, Anda dapat mengatur dan menjalankan serangkaian tugas di Amazon SageMaker Unified Studio. Alur kerja Amazon SageMaker Unified Studio menggunakan Apache Airflow untuk memodelkan prosedur pemrosesan data dan mengatur artefak kode Amazon Unified Studio Anda. SageMaker Untuk informasi selengkapnya, lihat bagian Alur Kerja. Untuk mempelajari lebih lanjut tentang Amazon SageMaker, lihat Apa itu Amazon SageMaker?

Kapan saya bisa menggunakan AWS Step Functions vs Amazon MWAA?

1. Anda dapat menggunakan Step Functions untuk memproses pesanan pelanggan individual, karena Step Functions dapat menskalakan untuk memenuhi permintaan untuk satu pesanan atau satu juta pesanan.
2. Jika Anda menjalankan alur kerja semalam yang memproses pesanan hari sebelumnya, Anda dapat menggunakan Step Functions atau Amazon MWAA. Amazon MWAA memberi Anda opsi open source untuk mengabstraksi alur kerja dari AWS sumber daya yang Anda gunakan.

Spesifikasi lingkungan

Berapa banyak penyimpanan tugas yang tersedia untuk setiap lingkungan?

Penyimpanan tugas dibatasi hingga 20 GB, dan ditentukan oleh Amazon ECS Fargate 1.4. Jumlah RAM ditentukan oleh kelas lingkungan yang Anda tentukan. Untuk informasi lebih lanjut tentang kelas lingkungan, lihatMengkonfigurasi kelas lingkungan Amazon MWAA.

Apa sistem operasi default yang digunakan untuk lingkungan Amazon MWAA?

Lingkungan Amazon MWAA dibuat pada instans yang menjalankan Amazon Linux 2 untuk versi 2.6 dan yang lebih lama, dan pada instans yang menjalankan Amazon Linux 2023 untuk versi 2.7 dan yang lebih baru.

Dapatkah saya menggunakan gambar khusus untuk lingkungan Amazon MWAA saya?

Gambar kustom tidak didukung. Amazon MWAA menggunakan gambar yang dibangun di Amazon Linux AMI. Amazon MWAA menginstal persyaratan tambahan dengan menjalankan `pip3 -r install` persyaratan yang ditentukan dalam file requirements.txt yang Anda tambahkan ke bucket Amazon S3 untuk lingkungan.

Apakah Amazon MWAA HIPAA sesuai?

Amazon MWAA memenuhi syarat Undang-Undang Portabilitas dan Akuntabilitas Asuransi Kesehatan (HIPAA). Jika Anda memiliki Adendum Rekanan Bisnis HIPAA (BAA) AWS, Anda dapat menggunakan Amazon MWAA untuk alur kerja yang menangani Informasi Kesehatan Terproteksi (PHI) pada lingkungan yang dibuat pada, atau setelah, 14 November 2022.

Apakah Amazon MWAA mendukung Instans Spot?

Amazon MWAA saat ini tidak mendukung jenis Instans Spot Amazon EC2 sesuai permintaan untuk Apache Airflow. Namun, lingkungan Amazon MWAA dapat memicu Instans Spot, misalnya, Amazon EMR dan Amazon EC2.

Apakah Amazon MWAA mendukung domain khusus?

Untuk dapat menggunakan domain khusus untuk nama host Amazon MWAA Anda, lakukan salah satu hal berikut:

- Untuk penerapan Amazon MWAA dengan akses server web publik, Anda dapat menggunakan Amazon dengan CloudFront Lambda @Edge untuk mengarahkan lalu lintas ke lingkungan Anda, dan memetakan nama domain khusus ke. CloudFront Untuk informasi selengkapnya dan contoh pengaturan domain kustom untuk lingkungan publik, lihat contoh domain kustom Amazon MWAA untuk server web publik di repositori contoh Amazon MWAA. GitHub
- Untuk penerapan Amazon MWAA dengan akses server web pribadi, lihat. the section called “Menyiapkan domain khusus”

Bisakah saya SSH ke lingkungan saya?

Meskipun SSH tidak didukung di lingkungan Amazon MWAA, dimungkinkan untuk menggunakan DAG untuk menjalankan perintah bash menggunakan file. `BashOperator` Contoh:

```
from airflow import DAG
    from airflow.operators.bash_operator import BashOperator
    from airflow.utils.dates import days_ago
    with DAG(dag_id="any_bash_command_dag", schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
    cli_command = BashOperator(
    task_id="bash_command",
    bash_command="{{ dag_run.conf['command'] }}"
    )
```

Untuk memicu DAG di Apache Airflow UI, gunakan:

```
{ "command" : "your bash command"}
```

Mengapa aturan referensi mandiri diperlukan pada grup keamanan VPC?

Dengan membuat aturan referensi diri, Anda membatasi sumber ke grup keamanan yang sama di VPC, dan itu tidak terbuka untuk semua jaringan. Untuk mempelajari lebih lanjut, lihatthe section called “Keamanan di VPC Anda”.

Dapatkah saya menyembunyikan lingkungan dari grup yang berbeda di IAM?

Anda dapat membatasi akses dengan menentukan nama lingkungan AWS Identity and Access Management, namun, pemfilteran akses tidak tersedia di AWS konsol—jika pengguna dapat mengakses satu lingkungan, mereka dapat mengakses semua lingkungan.

Dapatkah saya menyimpan data sementara pada pekerja Apache Airflow?

Operator Apache Airflow Anda dapat menyimpan data sementara pada pekerja. Pekerja Apache Airflow dapat mengakses file sementara `/tmp` di wadah Fargate untuk lingkungan Anda.

> **Note**
>
> Total penyimpanan tugas dibatasi hingga 20 GB, menurut Amazon ECS Fargate 1.4. Tidak ada jaminan bahwa tugas berikutnya berjalan pada instance kontainer Fargate yang sama, yang dapat menggunakan folder yang berbeda`/tmp`.

Bisakah saya menentukan lebih dari 25 pekerja Apache Airflow?

Ya. Meskipun Anda dapat menentukan hingga 25 pekerja Apache Airflow di konsol Amazon MWAA, Anda dapat mengonfigurasi hingga 50 pada lingkungan dengan meminta peningkatan kuota. Untuk informasi lebih lanjut, lihat Meminta kenaikan kuota.

Apakah Amazon MWAA mendukung VPC Amazon bersama atau subnet bersama?

Amazon MWAA tidak mendukung VPC Amazon bersama atau subnet bersama. VPC Amazon yang Anda pilih saat membuat lingkungan harus dimiliki oleh akun yang mencoba menciptakan lingkungan. Namun, Anda dapat merutekan lalu lintas dari VPC Amazon di akun Amazon MWAA ke VPC bersama. Untuk informasi selengkapnya dan contoh perutean lalu lintas ke VPC Amazon bersama, lihat Perutean keluar terpusat ke internet di Panduan Gateway Transit VPC Amazon.

Dapatkah saya membuat atau mengintegrasikan antrian Amazon SQS khusus untuk mengelola eksekusi tugas dan orkestrasi alur kerja di Apache Airflow?

Tidak, Anda tidak dapat membuat, memodifikasi, atau menggunakan antrian Amazon SQS khusus dalam Amazon MWAA. Ini karena Amazon MWAA secara otomatis menyediakan dan mengelola antrian Amazon SQS sendiri untuk setiap lingkungan Amazon MWAA.

Metrik-metrik

Metrik apa yang digunakan untuk menentukan apakah akan menskalakan pekerja?

Amazon MWAA memantau QueuedTasksdan RunningTasksmasuk CloudWatch untuk menentukan apakah akan menskalakan pekerja Apache Airflow di lingkungan Anda. Untuk mempelajari lebih lanjut, lihatPemantauan dan metrik.

Bisakah saya membuat metrik khusus? CloudWatch

Tidak di CloudWatch konsol. Namun, Anda dapat membuat DAG yang menulis metrik khusus. CloudWatch Untuk informasi lebih lanjut, lihatthe section called “Menggunakan DAG untuk menulis metrik khusus”.

DAG, Operator, Koneksi, dan pertanyaan lainnya

Dapatkah saya menggunakan **`PythonVirtualenvOperator`**?

`PythonVirtualenvOperator`Ini tidak didukung secara eksplisit di Amazon MWAA, tetapi Anda dapat membuat plugin khusus yang menggunakan. `PythonVirtualenvOperator` Untuk kode sampel, lihatthe section called “Plugin kustom untuk menambal PythonVirtualenvOperator ”.

Berapa lama waktu yang dibutuhkan Amazon MWAA untuk mengenali file DAG baru?

DAG disinkronkan secara berkala dari bucket Amazon S3 ke lingkungan Anda. Jika Anda menambahkan file DAG baru, dibutuhkan sekitar 300 detik bagi Amazon MWAA untuk mulai menggunakan file baru. Jika Anda memperbarui DAG yang ada, Amazon MWAA membutuhkan waktu sekitar 30 detik untuk mengenali pembaruan Anda.

Nilai-nilai ini, 300 detik untuk DAG baru, dan 30 detik untuk pembaruan ke DAG yang ada, sesuai dengan opsi konfigurasi Apache Airflow, dan masing-masing. `dag_dir_list_interval` `min_file_process_interval`

Mengapa file DAG saya tidak diambil oleh Apache Airflow?

Berikut ini adalah solusi yang mungkin untuk masalah ini:

1. Pastikan peran eksekusi Anda memiliki izin yang cukup untuk bucket Amazon S3 Anda. Untuk mempelajari lebih lanjut, lihatPeran eksekusi Amazon MWAA.
2. Periksa apakah bucket Amazon S3 telah mengonfigurasi Blokir Akses Publik, dan Pembuatan Versi diaktifkan. Untuk mempelajari lebih lanjut, lihatBuat bucket Amazon S3 untuk Amazon MWAA.
3. Verifikasi file DAG itu sendiri. Misalnya, pastikan bahwa setiap DAG memiliki ID DAG yang unik.

Bisakah saya menghapus **`plugins.zip`** atau **`requirements.txt`** dari lingkungan?

Saat ini, tidak ada cara untuk menghapus plugins.zip atau requirements.txt dari lingkungan setelah ditambahkan, tetapi kami sedang mengerjakan masalah ini. Untuk sementara, solusinya

adalah menunjuk ke teks kosong atau file zip, masing-masing. Untuk mempelajari lebih lanjut, lihatMenghapus file di Amazon S3.

Mengapa plugin saya tidak muncul di menu Apache Airflow v2.0.2 Admin Plugins?

Untuk alasan keamanan, server web Apache Airflow di Amazon MWAA memiliki jalan keluar jaringan yang terbatas, dan tidak menginstal plugin atau dependensi Python langsung di server web Apache Airflow untuk lingkungan versi 2.0.2. Plugin yang terdaftar memungkinkan Amazon MWAA untuk mengautentikasi pengguna Apache Airflow Anda di (IAM). AWS Identity and Access Management

Untuk dapat menginstal plugin dan dependensi Python langsung di server web, sebaiknya buat environemnt baru dengan Apache Airflow v2.2 dan yang lebih baru. Amazon MWAA menginstal dependensi Python dan dan plugin khusus langsung di server web untuk Apache Airflow v2.2 dan yang lebih baru.

Dapatkah saya menggunakan AWS Operator Database Migration Service (DMS)?

Amazon MWAA mendukung Operator DMS. Namun, operator ini tidak dapat digunakan untuk melakukan tindakan pada database metadata Amazon Aurora PostgreSQL yang terkait dengan lingkungan Amazon MWAA.

Saat saya mengakses Airflow REST API menggunakan AWS kredensialnya, dapatkah saya meningkatkan batas throttling menjadi lebih dari 10 transaksi per detik (TPS)?

Ya, Anda bisa. Untuk meningkatkan batas throttling, silakan hubungi Customer AWS Support.

Di mana server API Eksekusi Tugas Airflow berjalan di Amazon MWAA?

Amazon MWAA menjalankan Server API Eksekusi Tugas Airflow di komponen Webserver. API Eksekusi Tugas hanya tersedia di Apache Airflow v3 dan yang lebih baru. Untuk informasi lebih lanjut tentang arsitektur Amazon MWAA, lihat. Arsitektur

Memecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow

Bab ini menjelaskan masalah umum dan kesalahan yang dapat Anda temui saat menggunakan Apache Airflow di Alur Kerja Terkelola Amazon untuk Apache Airflow dan langkah-langkah yang disarankan untuk mengatasi kesalahan ini.

Daftar Isi

- Pemecahan masalah: DAGs, Operator, Koneksi, dan masalah lainnya
  - Koneksi
    - Saya tidak dapat terhubung ke Secrets Manager
    - Bagaimana cara mengonfigurasi kondisi pengelola secretsmanager:ResourceTag/<tag-key> rahasia atau pembatasan sumber daya dalam kebijakan peran eksekusi saya?
    - Saya tidak dapat terhubung ke Snowflake
    - Saya tidak dapat menemukan koneksi saya di Airflow UI
  - Server Web
    - Saya mendapatkan kesalahan 5xx saat mengakses server web
    - Saya mendapatkan The scheduler does not seem to be running kesalahan
  - Tugas
    - Saya membuat tugas saya macet atau tidak selesai
    - Saya mendapatkan kegagalan tugas tanpa log di Airflow v3
  - CLI
    - Saya mendapatkan kesalahan '503' saat memicu DAG di CLI
    - Mengapa perintah dags backfill Apache Airflow CLI gagal? Apakah ada solusinya?
  - Operator
    - Saya menerima PermissionError: [Errno 13] Permission denied kesalahan menggunakan operator S3Transform
- Pemecahan masalah: Membuat dan memperbarui lingkungan Amazon MWAA
  - Memperbarui requirements.txt
    - Saya menentukan versi baru saya requirements.txt dan butuh lebih dari 20 menit untuk memperbarui lingkungan saya

- Plugin
  - Apakah Amazon MWAA mendukung penerapan UI khusus?
- Buat ember
  - Saya tidak dapat memilih opsi untuk pengaturan Akses Publik Blok S3
- Ciptakan lingkungan
  - Saya mencoba menciptakan lingkungan dan macet di Creating negara bagian
  - Saya mencoba membuat lingkungan tetapi menampilkan status sebagai Create failed
  - Saya mencoba memilih VPC dan menerima kesalahan Network Failure
  - Saya mencoba membuat lingkungan dan menerima kesalahan layanan, partisi, atau sumber daya “harus dilewati”
  - Saya mencoba membuat lingkungan dan itu menampilkan status sebagai Available tetapi ketika saya mencoba mengakses UI Aliran Udara, 502 Bad Gateway kesalahan Empty Reply from Server atau ditampilkan
  - Saya mencoba membuat lingkungan dan nama pengguna saya adalah sekelompok nama karakter acak
- Perbarui lingkungan
  - Saya mencoba mengubah kelas lingkungan tetapi pembaruan gagal
- Lingkungan akses
  - Saya tidak dapat mengakses UI Apache Airflow
- Pemecahan masalah: CloudWatch Log dan kesalahan CloudTrail
  - Beberapa catatan
    - Saya tidak dapat menemukan log tugas saya, atau saya menerima Reading remote log from Cloudwatch log_group kesalahan
    - Tugas gagal tanpa log
    - Saya mendapatkan ResourceAlreadyExistsException kesalahan di CloudTrail
    - Saya mendapatkan Invalid request kesalahan di CloudTrail
    - Saya Cannot locate a 64-bit Oracle Client library: "libclntsh.so: cannot open shared object file: No such file or directory masuk log Apache Airflow
    - Saya mendapatkan psycopg2 'server menutup koneksi secara tidak terduga' di log penjadwal saya
    - Saya Executor reports task instance %s finished (%s) although the task says its %s masuk ke log pemrosesan DAG saya

- Saya Could not read remote logs from log_group: airflow-*{*environmentName}-Task log_stream:* {*DAG_ID}/*{*TASK_ID}/*{*time}/*{*n}.log. masuk ke log tugas saya

Pemecahan masalah: DAGs, Operator, Koneksi, dan masalah lainnya

Topik di halaman ini menjelaskan resolusi untuk dependensi Apache Airflow v2 dan v3 Python, plugin DAGs khusus,, Operator, Koneksi, tugas, dan masalah server web yang dapat Anda temui di lingkungan Alur Kerja Terkelola Amazon untuk Apache Airflow.

Daftar Isi

- Koneksi
  - Saya tidak dapat terhubung ke Secrets Manager
  - Bagaimana cara mengonfigurasi kondisi pengelola secretsmanager:ResourceTag/<tag-key> rahasia atau pembatasan sumber daya dalam kebijakan peran eksekusi saya?
  - Saya tidak dapat terhubung ke Snowflake
  - Saya tidak dapat menemukan koneksi saya di Airflow UI
- Server Web
  - Saya mendapatkan kesalahan 5xx saat mengakses server web
  - Saya mendapatkan The scheduler does not seem to be running kesalahan
- Tugas
  - Saya membuat tugas saya macet atau tidak selesai
  - Saya mendapatkan kegagalan tugas tanpa log di Airflow v3
- CLI
  - Saya mendapatkan kesalahan '503' saat memicu DAG di CLI
  - Mengapa perintah dags backfill Apache Airflow CLI gagal? Apakah ada solusinya?
- Operator
  - Saya menerima PermissionError: [Errno 13] Permission denied kesalahan menggunakan operator S3Transform

Koneksi

Topik berikut menjelaskan kesalahan yang mungkin Anda terima saat menggunakan koneksi Apache Airflow, atau menggunakan database lain. AWS

Saya tidak dapat terhubung ke Secrets Manager

Kami merekomendasikan langkah-langkah berikut:

1. Pelajari cara membuat kunci rahasia untuk koneksi dan variabel Apache Airflow Anda. the section called “Mengkonfigurasi Secrets Manager”
2. Pelajari cara menggunakan kunci rahasia untuk variabel `test-variable` Apache Airflow () di. Menggunakan kunci rahasia AWS Secrets Manager untuk variabel Apache Airflow
3. Pelajari cara menggunakan kunci rahasia untuk koneksi `myconn` Apache Airflow () di. Menggunakan kunci rahasia AWS Secrets Manager untuk koneksi Apache Airflow

Bagaimana cara mengonfigurasi kondisi pengelola **`secretsmanager:ResourceTag/<tag-key>`** rahasia atau pembatasan sumber daya dalam kebijakan peran eksekusi saya?

> **Note**
>
> Berlaku untuk Apache Airflow versi 2.0 dan sebelumnya.

Saat ini, Anda tidak dapat membatasi akses ke rahasia Secrets Manager dengan menggunakan kunci kondisi atau batasan sumber daya lainnya dalam peran eksekusi lingkungan Anda, karena masalah yang diketahui di Apache Airflow.

Saya tidak dapat terhubung ke Snowflake

Kami merekomendasikan langkah-langkah berikut:

1. Uji DAGs, plugin kustom, dan dependensi Python Anda secara lokal menggunakan on. aws-mwaa-docker-images GitHub
2. Tambahkan entri berikut ke requirements.txt untuk lingkungan Anda.

```
apache-airflow-providers-snowflake==1.3.0
```

3. Tambahkan impor berikut ke DAG Anda:

```
from airflow.providers.snowflake.operators.snowflake import SnowflakeOperator
```

Pastikan objek koneksi Apache Airflow menyertakan pasangan kunci-nilai berikut:

1. Id Conn: snowflake_conn
2. Jenis Conn: Kepingan Salju
3. Tuan rumah:<my account>. <my region if not us-west-2>.snowflakecomputing.com
4. Skema: <my schema>
5. Masuk: <my user name>
6. Kata sandi: ********
7. Pelabuhan: <port, if any>
8. Ekstra:

```
{
      "account": "<my account>",
      "warehouse": "<my warehouse>",
      "database": "<my database>",
      "region": "<my region if not using us-west-2 otherwise omit this line>"
      }
```

Contoh:

```
>>> import json
    >>> from airflow.models.connection import Connection
    >>> myconn = Connection(
    ...    conn_id='snowflake_conn',
    ...    conn_type='Snowflake',
    ...    host='123456789012.us-east-1.snowflakecomputing.com',
    ...    schema='YOUR_SCHEMA'
    ...    login='YOUR_USERNAME',
    ...    password='YOUR_PASSWORD',
    ...    port='YOUR_PORT'
    ...    extra=json.dumps(dict(account='123456789012', warehouse='YOUR_WAREHOUSE',
 database='YOUR_DB_OPTION', region='us-east-1')),
    ... )
```

Saya tidak dapat menemukan koneksi saya di Airflow UI

Apache Airflow menyediakan template koneksi di Apache Airflow UI. Ini menggunakan ini untuk menghasilkan string URI koneksi, terlepas dari jenis koneksi. Jika template koneksi tidak tersedia di Apache Airflow UI, template koneksi alternatif dapat digunakan untuk menghasilkan string URI koneksi, seperti menggunakan template koneksi HTTP.

Kami merekomendasikan langkah-langkah berikut:

1. Akses jenis koneksi yang disediakan Amazon MWAA di Apache Airflow UI di. Paket penyedia Apache Airflow diinstal di lingkungan Amazon MWAA
2. Akses perintah untuk membuat koneksi Apache Airflow di CLI di. Referensi perintah CLI Apache Airflow
3. Pelajari cara menggunakan templat koneksi di Apache Airflow UI secara bergantian untuk jenis koneksi yang tidak tersedia di Apache Airflow UI di Amazon MWAA di. Ikhtisar jenis koneksi

Server Web

Topik berikut menjelaskan kesalahan yang mungkin Anda terima untuk server web Apache Airflow Anda di Amazon MWAA.

Saya mendapatkan kesalahan 5xx saat mengakses server web

Kami merekomendasikan langkah-langkah berikut:

1. Periksa opsi konfigurasi Apache Airflow. Verifikasi bahwa pasangan nilai kunci yang Anda tentukan sebagai opsi konfigurasi Apache Airflow, seperti AWS Secrets Manager, telah dikonfigurasi dengan benar. Untuk mempelajari lebih lanjut, lihatthe section called “Saya tidak dapat terhubung ke Secrets Manager”.
2. Periksa`requirements.txt`. Verifikasi paket “ekstra” Aliran Udara dan pustaka lain yang tercantum dalam Anda `requirements.txt` kompatibel dengan versi Apache Airflow Anda.
3. Jelajahi cara untuk menentukan dependensi Python dalam `requirements.txt` file, lihat. Mengelola dependensi Python di requirements.txt

Saya mendapatkan **`The scheduler does not seem to be running`** kesalahan

Jika penjadwal tampaknya tidak berjalan, atau “detak jantung” terakhir diterima beberapa jam yang lalu, Anda DAGs mungkin tidak terdaftar di Apache Airflow, dan tugas baru tidak akan dijadwalkan.

Kami merekomendasikan langkah-langkah berikut:

1. Konfirmasikan bahwa grup keamanan VPC Anda mengizinkan akses masuk ke port. `5432` Port ini diperlukan untuk terhubung ke database metadata Amazon Aurora PostgreSQL untuk lingkungan Anda. Setelah aturan ini ditambahkan, berikan Amazon MWAA beberapa menit, dan kesalahan dapat hilang. Untuk mempelajari lebih lanjut, lihatthe section called “Keamanan di VPC Anda”.

   > **Note**
   >
   > - Metadatabase Aurora PostgreSQL adalah bagian dari arsitektur layanan Amazon MWAA dan tidak tersedia di Anda. Akun AWS
   > - Kesalahan terkait database biasanya merupakan gejala kegagalan penjadwal dan bukan akar penyebabnya.

2. Jika scheduler tidak berjalan, mungkin karena sejumlah faktor seperti kegagalan instalasi dependensi, atau penjadwal kelebihan beban. Konfirmasikan bahwa DAGs, plugin, dan persyaratan Anda berfungsi dengan benar dengan mengakses grup log yang sesuai di CloudWatch Log. Untuk mempelajari lebih lanjut, lihatPemantauan dan metrik.

Tugas

Topik berikut menjelaskan kesalahan yang mungkin Anda terima untuk tugas Apache Airflow di suatu lingkungan.

Saya membuat tugas saya macet atau tidak selesai

Jika tugas Apache Airflow Anda “macet” atau tidak selesai, kami merekomendasikan langkah-langkah berikut:

1. Mungkin ada sejumlah besar yang DAGs didefinisikan. Kurangi jumlah DAGs dan lakukan pembaruan lingkungan (seperti mengubah level log) untuk memaksa reset.

   a. Aliran udara DAGs mem-parsing apakah mereka diaktifkan atau tidak. Jika Anda menggunakan lebih dari 50% kapasitas lingkungan Anda, Anda mungkin mulai membanjiri penjadwal Apache Airflow. Hal ini menyebabkan Total Parse Time yang besar dalam CloudWatch Metrik atau waktu pemrosesan DAG yang lama di CloudWatch Log. Ada

cara lain untuk mengoptimalkan konfigurasi Apache Airflow yang berada di luar cakupan panduan ini.

b. Untuk mempelajari lebih lanjut tentang praktik terbaik yang kami sarankan untuk menyesuaikan kinerja lingkungan Anda, lihatthe section called “Penyetelan kinerja untuk Apache Airflow”.

2. Mungkin ada banyak tugas dalam antrian. Ini sering ditampilkan sebagai jumlah tugas yang besar—dan berkembang—di `None` negara bagian, atau sebagai sejumlah besar tugas di dalam. `Queued Tasks` and/or `Tasks Pending` CloudWatch Ini dapat terjadi karena alasan berikut:

   a. Jika ada lebih banyak tugas untuk dijalankan daripada lingkungan yang memiliki kapasitas untuk dijalankan, and/or sejumlah besar tugas yang antri sebelum penskalaan otomatis memiliki waktu untuk mendeteksi tugas dan menyebarkan pekerja tambahan.

   b. Jika ada lebih banyak tugas untuk dijalankan daripada lingkungan yang memiliki kapasitas untuk dijalankan, kami sarankan untuk mengurangi jumlah tugas yang Anda DAGs jalankan secara bersamaan, and/or meningkatkan pekerja Apache Airflow minimum.

   c. Jika ada sejumlah besar tugas yang antri sebelum penskalaan otomatis memiliki waktu untuk mendeteksi dan menyebarkan pekerja tambahan, kami merekomendasikan penyebaran tugas yang mengejutkan untuk meningkatkan pekerja Apache Airflow minimum. and/or

   d. Anda dapat menggunakan perintah update-environment di AWS Command Line Interface (AWS CLI) untuk mengubah jumlah minimum atau maksimum pekerja yang berjalan di lingkungan Anda.

      ```
      aws mwaa update-environment --name MyEnvironmentName --min-workers 2 --max-
      workers 10
      ```

   e. Untuk mempelajari lebih lanjut tentang praktik terbaik yang kami sarankan untuk menyesuaikan kinerja lingkungan Anda, lihatthe section called “Penyetelan kinerja untuk Apache Airflow”.

3. Jika tugas Anda macet dalam status “berjalan”, Anda juga dapat menghapus tugas atau menandainya sebagai berhasil atau gagal. Hal ini memungkinkan komponen penskalaan otomatis untuk lingkungan Anda untuk mengurangi jumlah pekerja yang berjalan di lingkungan Anda. Gambar berikut menggambarkan contoh tugas yang terdampar.

   - Pilih lingkaran untuk tugas yang terdampar, lalu pilih Hapus (seperti yang ditunjukkan). Hal ini memungkinkan Amazon MWAA untuk menurunkan skala pekerja; jika tidak, Amazon

MWAA tidak dapat menentukan mana yang DAGs diaktifkan atau dinonaktifkan, dan tidak dapat menurunkan skala, jika masih ada tugas antrian.

4. Pelajari lebih lanjut tentang siklus hidup tugas Apache Airflow di Concepts dalam panduan referensi Apache Airflow.

Saya mendapatkan kegagalan tugas tanpa log di Airflow v3

Jika tugas Apache Airflow 3 Anda gagal tanpa log, ikuti langkah-langkah berikut:

- Jika log pekerja menunjukkan kesalahan seperti `Task handler raised error: WorkerLostError('Worker exited prematurely: exitcode 15 Job: 12.')` sekitar waktu tugas gagal, ini menunjukkan bahwa proses pekerja bercabang yang ditugaskan ke tugas kemungkinan dihentikan secara tidak terduga.

  Untuk mengatasi hal ini, pertimbangkan untuk mengonfigurasi celery.worker_autoscale dengan nilai minimum dan maksimum yang sama. Contoh:

  ```
  celery.worker_autoscale=5,5  # for mw1.small
  celery.worker_autoscale=10,10 # for mw1.medium
  celery.worker_autoscale=20,20 # for mw1.large
  ```

  Ini memastikan bahwa ukuran kumpulan pekerja tetap tetap, mencegah penghentian pekerja yang tidak terduga.

CLI

Topik berikut menjelaskan kesalahan yang mungkin Anda terima saat menjalankan perintah CLI Airflow di. AWS Command Line Interface

Saya mendapatkan kesalahan '503' saat memicu DAG di CLI

Airflow CLI berjalan di server web Apache Airflow, yang memiliki konkurensi terbatas. Biasanya maksimal 4 perintah CLI dapat berjalan secara bersamaan.

Mengapa perintah `dags backfill` Apache Airflow CLI gagal? Apakah ada solusinya?

> **Note**
>
> Berikut ini hanya berlaku untuk lingkungan Apache Airflow v2.0.2.

`backfill`Perintah, seperti perintah Apache Airflow CLI lainnya, mem-parsing DAGs semua secara lokal sebelum diproses, terlepas dari DAGs DAG mana operasi CLI berlaku. Di lingkungan Amazon MWAA menggunakan Apache Airflow v2.0.2, karena plugin dan persyaratan belum diinstal pada server web pada saat perintah CLI berjalan, operasi parsing gagal, dan operasi tidak dipanggil. `backfill` Jika Anda tidak memiliki persyaratan atau plugin di lingkungan Anda, `backfill` operasi akan berhasil.

Untuk dapat menjalankan perintah `backfill` CLI, kami sarankan untuk memanggilnya di operator bash. Dalam operator bash, `backfill` dimulai dari pekerja, memungkinkan DAGs untuk mengurai dengan sukses karena semua persyaratan dan plguin yang diperlukan tersedia dan diinstal. Gunakan contoh berikut untuk membuat DAG dengan `BashOperator` to run`backfill`.

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from airflow.utils.dates import days_ago

with DAG(dag_id="backfill_dag", schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
    cli_command = BashOperator(
        task_id="bash_command",
        bash_command="airflow dags backfill my_dag_id"
    )
```

Operator

Topik berikut menjelaskan kesalahan yang mungkin Anda terima saat menggunakan Operator.

Saya menerima `PermissionError: [Errno 13] Permission denied` kesalahan menggunakan operator S3Transform

Kami merekomendasikan langkah-langkah berikut jika Anda mencoba menjalankan skrip shell dengan operator S3Transform dan Anda menerima kesalahan. `PermissionError: [Errno 13]`

`Permission denied` Langkah-langkah berikut mengasumsikan Anda memiliki file plugins.zip yang ada. Jika Anda membuat plugins.zip baru, lihatMenginstal plugin kustom.

1. Uji DAGs, plugin kustom, dan dependensi Python Anda secara lokal menggunakan on. aws-mwaa-docker-images GitHub
2. Buat skrip “transformasi” Anda.

```
#!/bin/bash
cp $1 $2
```

3. (opsional) pengguna macOS dan Linux mungkin perlu menjalankan perintah berikut untuk memastikan skrip dapat dieksekusi.

```
chmod 777 transform_test.sh
```

4. Tambahkan skrip ke plugins.zip Anda.

```
zip plugins.zip transform_test.sh
```

5. Ikuti langkah-langkah di Unggah plugins.zip ke Amazon S3.
6. Ikuti langkah-langkah dalam Menentukan versi plugins.zip di konsol Amazon MWAA.
7. Buat DAG berikut.

```
from airflow import DAG
      from airflow.providers.amazon.aws.operators.s3_file_transform import
 S3FileTransformOperator
      from airflow.utils.dates import days_ago
      import os

      DAG_ID = os.path.basename(__file__).replace(".py", "")

      with DAG (dag_id=DAG_ID, schedule_interval=None, catchup=False,
 start_date=days_ago(1)) as dag:
      file_transform = S3FileTransformOperator(
      task_id='file_transform',
      transform_script='/usr/local/airflow/plugins/transform_test.sh',
      source_s3_key='s3://amzn-s3-demo-bucket/files/input.txt',
      dest_s3_key='s3://amzn-s3-demo-bucket/files/output.txt'
      )
```

8. Ikuti langkah-langkah di Mengunggah kode DAG ke Amazon S3.

Pemecahan masalah: Membuat dan memperbarui lingkungan Amazon MWAA

Topik di halaman ini berisi kesalahan yang dapat Anda temui saat membuat dan memperbarui Alur Kerja Terkelola Amazon untuk lingkungan Apache Airflow dan cara mengatasi kesalahan ini.

Daftar Isi

- Memperbarui requirements.txt
  - Saya menentukan versi baru saya requirements.txt dan butuh lebih dari 20 menit untuk memperbarui lingkungan saya
- Plugin
  - Apakah Amazon MWAA mendukung penerapan UI khusus?
- Buat ember
  - Saya tidak dapat memilih opsi untuk pengaturan Akses Publik Blok S3
- Ciptakan lingkungan
  - Saya mencoba menciptakan lingkungan dan macet di Creating negara bagian
  - Saya mencoba membuat lingkungan tetapi menampilkan status sebagai Create failed
  - Saya mencoba memilih VPC dan menerima kesalahan Network Failure
  - Saya mencoba membuat lingkungan dan menerima kesalahan layanan, partisi, atau sumber daya “harus dilewati”
  - Saya mencoba membuat lingkungan dan itu menampilkan status sebagai Available tetapi ketika saya mencoba mengakses UI Aliran Udara, 502 Bad Gateway kesalahan Empty Reply from Server atau ditampilkan
  - Saya mencoba membuat lingkungan dan nama pengguna saya adalah sekelompok nama karakter acak
- Perbarui lingkungan
  - Saya mencoba mengubah kelas lingkungan tetapi pembaruan gagal
- Lingkungan akses
  - Saya tidak dapat mengakses UI Apache Airflow

Memperbarui `requirements.txt`

Topik berikut menjelaskan kesalahan yang mungkin Anda terima saat memperbarui`requirements.txt`.

Saya menentukan versi baru saya `requirements.txt` dan butuh lebih dari 20 menit untuk memperbarui lingkungan saya

Jika dibutuhkan lebih dari dua puluh menit bagi lingkungan Anda untuk menginstal versi baru `requirements.txt` file, pembaruan lingkungan gagal dan Amazon MWAA kembali ke versi stabil terakhir dari gambar kontainer.

1. Periksa versi paket. Kami menyarankan untuk selalu menentukan versi tertentu (==) atau versi maksimum (<=) untuk dependensi Python di Anda. `requirements.txt`
2. Periksa log Apache Airflow. Jika Anda mengaktifkan log Apache Airflow, verifikasi grup log Anda berhasil dibuat di halaman grup Log di konsol. CloudWatch Jika Anda mendapatkan log kosong, alasan paling umum adalah karena izin yang hilang dalam peran eksekusi Anda untuk CloudWatch atau Amazon S3 tempat log ditulis. Untuk mempelajari lebih lanjut, lihatPeran eksekusi.
3. Periksa opsi konfigurasi Apache Airflow. Jika Anda menggunakan Secrets Manager, verifikasi bahwa pasangan nilai kunci yang Anda tentukan sebagai opsi konfigurasi Apache Airflow telah dikonfigurasi dengan benar. Untuk mempelajari lebih lanjut, lihatthe section called “Mengkonfigurasi Secrets Manager”.
4. Periksa konfigurasi jaringan VPC. Untuk mempelajari lebih lanjut, lihatthe section called “Lingkungan macet”.
5. Periksa izin peran eksekusi. Peran eksekusi adalah peran AWS Identity and Access Management (IAM) dengan kebijakan izin yang memberikan izin Amazon MWAA untuk memanggil sumber daya layanan AWS lain (seperti Amazon S3, Amazon CloudWatch SQS, Amazon ECR) atas nama Anda. Kunci yang dikelola Pelanggan atau kunci AWS milik Anda juga harus diizinkan akses. Untuk mempelajari lebih lanjut, lihatPeran eksekusi.
6. Untuk menjalankan skrip pemecahan masalah yang memeriksa penyiapan dan konfigurasi jaringan Amazon VPC untuk lingkungan Amazon MWAA Anda, lihat skrip Verifikasi Lingkungan di Alat Dukungan aktif. AWS GitHub

Plugin

Topik berikut menjelaskan masalah yang dapat Anda temui saat mengonfigurasi atau memperbarui plugin Apache Airflow.

Apakah Amazon MWAA mendukung penerapan UI khusus?

Dimulai dengan Apache Airflow v2.2.2, Amazon MWAA mendukung pemasangan plugin di server web Apache Airflow, dan menerapkan UI kustom. Jika lingkungan Amazon MWAA Anda menjalankan Apache Airflow v2.0.2 atau yang lebih lama, Anda tidak akan dapat menerapkan UI kustom.

Untuk informasi selengkapnya tentang manajemen versi, dan meningkatkan lingkungan yang ada, lihat. Versi

Buat ember

Topik berikut menjelaskan kesalahan yang mungkin Anda terima saat membuat bucket Amazon S3.

Saya tidak dapat memilih opsi untuk pengaturan Akses Publik Blok S3

Peran eksekusi untuk lingkungan Amazon MWAA Anda memerlukan izin untuk `GetBucketPublicAccessBlock` tindakan di bucket Amazon S3 untuk memverifikasi akses publik yang diblokir bucket. Kami merekomendasikan langkah-langkah berikut:

1. Ikuti langkah-langkah untuk Melampirkan kebijakan JSON ke peran eksekusi Anda.
2. Lampirkan kebijakan JSON berikut:

```
{
  "Effect":"Allow",
  "Action":[
    "s3:GetObject*",
    "s3:GetBucket*",
    "s3:List*"
  ],
  "Resource":[
    "arn:aws:s3:::amzn-s3-demo-bucket",
    "arn:aws:s3:::amzn-s3-demo-bucket/*"
  ]
}
```

Ganti placeholder sampel *amzn-s3-demo-bucket* dengan nama bucket Amazon S3 Anda.

3. Untuk menjalankan skrip pemecahan masalah yang memeriksa penyiapan dan konfigurasi jaringan Amazon VPC untuk lingkungan Amazon MWAA Anda, lihat skrip Verifikasi Lingkungan di Alat Dukungan aktif. AWS GitHub

Ciptakan lingkungan

Topik berikut menjelaskan kesalahan yang mungkin Anda terima saat membuat lingkungan.

Saya mencoba menciptakan lingkungan dan macet di **`Creating`** negara bagian

Kami merekomendasikan langkah-langkah berikut:

1. Periksa jaringan VPC dengan perutean publik. Jika Anda menggunakan VPC Amazon dengan akses internet, verifikasi hal berikut:

    - Bahwa VPC Amazon Anda dikonfigurasi untuk memungkinkan lalu lintas jaringan antara berbagai AWS sumber daya yang digunakan oleh lingkungan Amazon MWAA Anda, sebagaimana didefinisikan dalam. the section called “Tentang jaringan” Misalnya, grup keamanan VPC Anda harus mengizinkan semua lalu lintas dalam aturan referensi mandiri, atau secara opsional menentukan rentang port untuk rentang port HTTPS 443 dan rentang port TCP 5432.
2. Periksa jaringan VPC dengan perutean pribadi. Jika Anda menggunakan VPC Amazon tanpa akses internet, verifikasi hal berikut:

    - Bahwa VPC Amazon Anda dikonfigurasi untuk memungkinkan lalu lintas jaringan antara AWS sumber daya yang berbeda untuk lingkungan Amazon MWAA Anda, sebagaimana didefinisikan dalam. the section called “Tentang jaringan” Misalnya, dua subnet pribadi Anda tidak boleh memiliki tabel rute ke gateway NAT (atau instance NAT), atau gateway internet.
3. Untuk menjalankan skrip pemecahan masalah yang memeriksa penyiapan dan konfigurasi jaringan Amazon VPC untuk lingkungan Amazon MWAA Anda, lihat skrip Verifikasi Lingkungan di Alat Dukungan aktif. AWS GitHub

Saya mencoba membuat lingkungan tetapi menampilkan status sebagai **`Create failed`**

Kami merekomendasikan langkah-langkah berikut:

1. Periksa konfigurasi jaringan VPC. Untuk mempelajari lebih lanjut, lihatthe section called “Lingkungan macet”.
2. Periksa izin pengguna. Amazon MWAA melakukan dry run terhadap kredensi pengguna sebelum membuat lingkungan. Anda Akun AWS mungkin tidak memiliki izin di AWS Identity and Access Management (IAM) untuk membuat beberapa sumber daya untuk suatu lingkungan. Misalnya, jika Anda memilih mode akses Apache Airflow jaringan pribadi, Akun AWS Anda harus telah diberikan akses oleh administrator Anda ke kebijakan kontrol akses MWAAFullConsoleAccessAmazon untuk lingkungan Anda, yang memungkinkan akun Anda membuat titik akhir VPC.
3. Periksa izin peran eksekusi. Peran eksekusi adalah peran AWS Identity and Access Management (IAM) dengan kebijakan izin yang memberikan izin Amazon MWAA untuk memanggil sumber daya layanan AWS lain (seperti Amazon S3, Amazon CloudWatch SQS, Amazon ECR) atas nama Anda. Kunci yang dikelola Pelanggan atau kunci AWS milik Anda juga harus diizinkan akses. Untuk mempelajari lebih lanjut, lihatPeran eksekusi.
4. Periksa log Apache Airflow. Jika Anda mengaktifkan log Apache Airflow, verifikasi grup log Anda berhasil dibuat di halaman grup Log di konsol. CloudWatch Jika Anda mendapatkan log kosong, alasan paling umum adalah karena izin yang hilang dalam peran eksekusi Anda untuk CloudWatch atau Amazon S3 tempat log ditulis. Untuk mempelajari lebih lanjut, lihatPeran eksekusi.
5. Untuk menjalankan skrip pemecahan masalah yang memeriksa penyiapan dan konfigurasi jaringan Amazon VPC untuk lingkungan Amazon MWAA Anda, lihat skrip Verifikasi Lingkungan di Alat Dukungan aktif. AWS GitHub
6. Jika Anda menggunakan VPC Amazon tanpa akses internet, pastikan Anda telah membuat titik akhir gateway Amazon S3, dan memberikan izin minimum yang diperlukan ke Amazon ECR untuk mengakses Amazon S3. Untuk mempelajari selengkapnya tentang membuat titik akhir gateway Amazon S3, lihat yang berikut ini:

    - Membuat jaringan VPC Amazon tanpa akses internet
    - Buat titik akhir gateway Amazon S3 di Panduan Pengguna Amazon Elastic Container Registry

Saya mencoba memilih VPC dan menerima kesalahan **`Network Failure`**

Kami merekomendasikan langkah-langkah berikut:

- Jika Anda mendapatkan `Network Failure` kesalahan saat mencoba memilih VPC Amazon saat membuat lingkungan, matikan proxy dalam browser yang sedang berjalan, lalu coba lagi.

Saya mencoba membuat lingkungan dan menerima kesalahan layanan, partisi, atau sumber daya “harus dilewati”

Kami merekomendasikan langkah-langkah berikut:

- Anda mungkin menerima kesalahan ini karena URI yang Anda tentukan untuk bucket Amazon S3 menyertakan '/' di akhir URI. Kami merekomendasikan untuk menghapus '/' di jalur. Nilai harus dalam format berikut:

```
s3://amzn-s3-demo-bucket
```

Saya mencoba membuat lingkungan dan itu menampilkan status sebagai **`Available`** tetapi ketika saya mencoba mengakses UI Aliran Udara, **`502 Bad Gateway`** kesalahan **`Empty Reply from Server`** atau ditampilkan

Kami merekomendasikan langkah-langkah berikut:

1. Periksa konfigurasi grup keamanan VPC. Untuk mempelajari lebih lanjut, lihatthe section called “Lingkungan macet”.
2. Konfirmasikan bahwa paket Apache Airflow apa pun yang Anda cantumkan sesuai dengan versi Apache Airflow yang Anda jalankan di `requirements.txt` Amazon MWAA. Untuk mempelajari lebih lanjut, lihatMenginstal dependensi Python.
3. Untuk menjalankan skrip pemecahan masalah yang memeriksa penyiapan dan konfigurasi jaringan Amazon VPC untuk lingkungan Amazon MWAA Anda, lihat skrip Verifikasi Lingkungan di Alat Dukungan aktif. AWS GitHub

Saya mencoba membuat lingkungan dan nama pengguna saya adalah sekelompok nama karakter acak

- Apache Airflow memiliki maksimal 64 karakter untuk nama pengguna. Jika peran Anda AWS Identity and Access Management (IAM) melebihi panjang ini, algoritma hash digunakan untuk menguranginya, namun tetap unik.

Perbarui lingkungan

Topik berikut menjelaskan kesalahan yang mungkin Anda terima saat memperbarui lingkungan.

Saya mencoba mengubah kelas lingkungan tetapi pembaruan gagal

Jika Anda memperbarui lingkungan Anda ke kelas lingkungan yang berbeda (seperti mengubah `mw1.medium` ke an`mw1.small`), dan permintaan untuk memperbarui lingkungan Anda gagal, status lingkungan akan masuk ke `UPDATE_FAILED` status dan lingkungan digulung kembali ke, dan ditagih sesuai dengan, versi stabil sebelumnya dari lingkungan.

Kami merekomendasikan langkah-langkah berikut:

1. Uji DAGs, plugin kustom, dan dependensi Python Anda secara lokal menggunakan on. aws-mwaa-docker-images GitHub
2. Untuk menjalankan skrip pemecahan masalah yang memeriksa penyiapan dan konfigurasi jaringan Amazon VPC untuk lingkungan Amazon MWAA Anda, lihat skrip Verifikasi Lingkungan di Alat Dukungan aktif. AWS GitHub

Lingkungan akses

Topik berikut menjelaskan kesalahan yang mungkin Anda terima saat mengakses lingkungan.

Saya tidak dapat mengakses UI Apache Airflow

Kami merekomendasikan langkah-langkah berikut:

1. Periksa izin pengguna. Anda mungkin tidak diberikan akses ke kebijakan izin yang dapat Anda gunakan untuk mengakses UI Apache Airflow. Untuk mempelajari lebih lanjut, lihatthe section called “Mengakses lingkungan Amazon MWAA”.
2. Periksa akses jaringan. Ini mungkin karena Anda memilih mode akses jaringan pribadi. Jika URL UI Apache Airflow Anda dalam `387fbcn-8dh4-9hfj-0dnd-834jhdfb-vpce.c10.us-west-2.airflow.amazonaws.com` format berikut, itu berarti Anda menggunakan perutean pribadi untuk server web Apache Airflow Anda. Anda dapat memperbarui mode akses Apache Airflow ke mode akses jaringan Publik, atau membuat mekanisme untuk mengakses titik akhir VPC untuk server Web Apache Airflow Anda. Untuk mempelajari lebih lanjut, lihatthe section called “Mengelola akses ke titik akhir VPC”.

Pemecahan masalah: CloudWatch Log dan kesalahan CloudTrail

Topik di halaman ini berisi resolusi untuk CloudWatch Log Amazon dan AWS CloudTrail kesalahan yang dapat Anda temui di lingkungan Alur Kerja Terkelola Amazon untuk Apache Airflow.

Daftar Isi

- Beberapa catatan
  - Saya tidak dapat menemukan log tugas saya, atau saya menerima Reading remote log from Cloudwatch log_group kesalahan
  - Tugas gagal tanpa log
  - Saya mendapatkan ResourceAlreadyExistsException kesalahan di CloudTrail
  - Saya mendapatkan Invalid request kesalahan di CloudTrail
  - Saya Cannot locate a 64-bit Oracle Client library: "libclntsh.so: cannot open shared object file: No such file or directory masuk log Apache Airflow
  - Saya mendapatkan psycopg2 'server menutup koneksi secara tidak terduga' di log penjadwal saya
  - Saya Executor reports task instance %s finished (%s) although the task says its %s masuk ke log pemrosesan DAG saya
  - Saya Could not read remote logs from log_group: airflow-*{*environmentName}-Task log_stream:* {*DAG_ID}/*{*TASK_ID}/*{*time}/*{*n}.log. masuk ke log tugas saya

Beberapa catatan

Topik berikut menjelaskan kesalahan yang mungkin Anda terima saat mengakses log Apache Airflow.

Saya tidak dapat menemukan log tugas saya, atau saya menerima **`Reading remote log from Cloudwatch log_group`** kesalahan

Amazon MWAA telah mengonfigurasi Apache Airflow untuk membaca dan menulis log langsung dari dan ke Amazon Logs. CloudWatch Jika seorang pekerja gagal memulai tugas, atau gagal menulis log apa pun, Anda akan merujuk ke kesalahan:

```
*** Reading remote log from Cloudwatch log_group: airflow-environmentName-Task
 log_stream: DAG_ID/TASK_ID/timestamp/n.log.Could not read remote logs from log_group:
 airflow-environmentName-Task log_stream: DAG_ID/TASK_ID/time/n.log.
```

- Kami merekomendasikan langkah-langkah berikut:

  a. Verifikasi bahwa Anda telah mengaktifkan log tugas di `INFO` tingkat untuk lingkungan Anda. Untuk informasi lebih lanjut, lihatMengakses log Aliran Udara di Amazon CloudWatch.

  b. Verifikasi bahwa peran eksekusi lingkungan memiliki kebijakan izin yang benar.

c. Verifikasi bahwa operator atau tugas Anda berfungsi dengan benar, memiliki sumber daya yang cukup untuk mengurai DAG, dan memiliki pustaka Python yang sesuai untuk dimuat. Untuk memverifikasi apakah Anda memiliki dependensi yang benar, coba hilangkan impor hingga Anda menemukan yang menyebabkan masalah. Kami merekomendasikan pengujian dependensi Python Anda menggunakan. aws-mwaa-docker-images

Tugas gagal tanpa log

Jika tugas gagal dalam alur kerja dan Anda tidak dapat menemukan log apa pun untuk tugas yang gagal, periksa apakah Anda menyetel `queue` parameter dalam argumen default, seperti yang tercantum dalam berikut ini.

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from airflow.utils.dates import days_ago

# Setting queue argument to default.
default_args = {
 "start_date": days_ago(1),
 "queue": "default"
}

with DAG(dag_id="any_command_dag", schedule_interval=None, catchup=False,
 default_args=default_args) as dag:
    cli_command = BashOperator(
        task_id="bash_command",
        bash_command="{{ dag_run.conf['command'] }}"
    )
```

Untuk mengatasi masalah, hapus `queue` dari kode Anda, dan panggil DAG lagi.

Saya mendapatkan **`ResourceAlreadyExistsException`** kesalahan di CloudTrail

```
"errorCode": "ResourceAlreadyExistsException",
    "errorMessage": "The specified log stream already exists",
    "requestParameters": {
        "logGroupName": "airflow-MyAirflowEnvironment-DAGProcessing",
        "logStreamName": "scheduler_cross-account-eks.py.log"
    }
```

Persyaratan Python tertentu seperti `apache-airflow-backport-providers-amazon` memutar kembali `watchtower` perpustakaan yang digunakan Amazon MWAA untuk berkomunikasi dengan CloudWatch versi yang lebih lama. Kami merekomendasikan langkah-langkah berikut:

- Tambahkan pustaka berikut ke `requirements.txt`

```
watchtower==1.0.6
```

Saya mendapatkan **`Invalid request`** kesalahan di CloudTrail

```
Invalid request provided: Provided role does not have sufficient permissions for s3
 location airflow-xxx-xxx/dags
```

Jika Anda membuat lingkungan Amazon MWAA dan bucket Amazon S3 menggunakan template CloudFormation yang sama, Anda perlu menambahkan bagian `DependsOn` dalam template Anda. CloudFormation Dua sumber daya (Lingkungan MWAA dan Kebijakan Eksekusi MWAA) memiliki ketergantungan dalam. CloudFormation Kami merekomendasikan langkah-langkah berikut:

- Tambahkan **`DependsOn`** pernyataan berikut ke CloudFormation template Anda.

```
...
  MaxWorkers: 5
  NetworkConfiguration:
    SecurityGroupIds:
      - !GetAtt SecurityGroup.GroupId
    SubnetIds: !Ref subnetIds
  WebserverAccessMode: PUBLIC_ONLY
DependsOn: MwaaExecutionPolicy

 MwaaExecutionPolicy:
 Type: AWS::IAM::ManagedPolicy
 Properties:
   Roles:
    - !Ref MwaaExecutionRole
  PolicyDocument:
    Version: 2012-10-17
    Statement:
      - Effect: Allow
        Action: airflow:PublishMetrics
        Resource:
```

```
...
```

Sebagai contoh, lihatTutorial mulai cepat untuk Alur Kerja Terkelola Amazon untuk Apache Airflow.

Saya **`Cannot locate a 64-bit Oracle Client library: "libclntsh.so: cannot open shared object file: No such file or directory`** masuk log Apache Airflow

- Kami merekomendasikan langkah-langkah berikut:
  - Jika Anda menggunakan Apache Airflow v2, `core.lazy_load_plugins : False` tambahkan sebagai opsi konfigurasi Apache Airflow. Untuk mempelajari lebih lanjut, lihat Menggunakan opsi konfigurasi untuk memuat plugin di 2.

Saya mendapatkan psycopg2 'server menutup koneksi secara tidak terduga' di log penjadwal saya

Jika Anda mendapatkan kesalahan yang mirip dengan berikut ini, penjadwal Apache Airflow Anda mungkin kehabisan sumber daya.

```
2021-06-14T10:20:24.581-05:00    sqlalchemy.exc.OperationalError:
 (psycopg2.OperationalError) server closed the connection unexpectedly
2021-06-14T10:20:24.633-05:00    This probably means the server terminated abnormally
2021-06-14T10:20:24.686-05:00    before or while processing the request.
```

Kami merekomendasikan langkah-langkah berikut:

- Pertimbangkan untuk meningkatkan ke Apache Airflow v2.0.2, yang dapat Anda gunakan untuk menentukan hingga 5 penjadwal.

Saya **`Executor reports task instance %s finished (%s) although the task says its %s`** masuk ke log pemrosesan DAG saya

Jika Anda mendapatkan kesalahan yang mirip dengan berikut ini, tugas Anda yang berjalan lama mungkin telah mencapai batas waktu tugas di Amazon MWAA. Amazon MWAA memiliki batas 12

jam untuk salah satu tugas Aliran Udara, untuk mencegah tugas macet dalam antrian dan memblokir aktivitas seperti penskalaan otomatis.

```
Executor reports task instance %s finished (%s) although the task says its %s. (Info:
 %s) Was the task killed externally
```

Kami merekomendasikan langkah-langkah berikut:

- Pertimbangkan untuk memecah tugas menjadi beberapa tugas yang berjalan lebih pendek. Aliran udara biasanya memiliki model di mana operator asinkron. Ini memanggil aktivitas pada sistem eksternal, dan jajak pendapat Sensor Aliran Udara Apache untuk memeriksa kapan selesai. Jika Sensor gagal, sensor dapat dicoba ulang dengan aman tanpa memengaruhi fungsionalitas Operator.

Saya **`Could not read remote logs from log_group: airflow-*{*environmentName}-Task log_stream:* {*DAG_ID}/*{*TASK_ID}/*{*time}/*{*n}.log.`** masuk ke log tugas saya

Jika Anda mendapatkan kesalahan yang mirip dengan berikut ini, peran eksekusi untuk lingkungan Anda mungkin tidak berisi kebijakan izin untuk membuat aliran log untuk log tugas.

```
Could not read remote logs from log_group: airflow-*{*environmentName}-Task
 log_stream:* {*DAG_ID}/*{*TASK_ID}/*{*time}/*{*n}.log.
```

Kami merekomendasikan langkah-langkah berikut:

- Ubah peran eksekusi untuk lingkungan Anda menggunakan salah satu kebijakan sampel dithe section called “Peran eksekusi”.

Anda mungkin juga telah menentukan paket penyedia dalam `requirements.txt` file Anda yang tidak kompatibel dengan versi Apache Airflow Anda. Misalnya, jika Anda menggunakan Apache Airflow v2.0.2, Anda mungkin telah menentukan paket, seperti apache-airflow-providers-databrickspaket, yang hanya kompatibel dengan Airflow 2.1+.

Kami merekomendasikan langkah-langkah berikut:

1. Jika Anda menggunakan Apache Airflow v2.0.2, ubah file dan tambahkan. `requirements.txt` `apache-airflow[databricks]` Ini menginstal versi yang benar dari paket Databricks yang kompatibel dengan Apache Airflow v2.0.2.
2. Uji DAGs, plugin kustom, dan dependensi Python Anda secara lokal menggunakan on. aws-mwaa-docker-images GitHub

Riwayat Panduan Pengguna Amazon MWAA

Tabel berikut menjelaskan penambahan penting pada dokumentasi layanan Amazon MWAA, dimulai pada November 2020. Untuk menerima pemberitahuan tentang pembaruan dokumentasi ini, berlangganan umpan RSS.

Perubahan	Deskripsi	Tanggal
Mode akses server web baru	Amazon MWAA sekarang mendukung mode akses server web ketiga, Baik akses jaringan publik dan pribadi, untuk lingkungan yang menjalankan Apache Airflow v3.2.1 dan yang lebih baru. Mode ini menciptak an penyeimbang beban jaringan publik untuk akses browser ke Apache Airflow UI dan titik akhir VPC pribadi untuk komunikasi worker-to -webserver, memungkinkan pekerja untuk menjangka u server web di VPC tanpa akses internet. • the section called “Mode akses Apache Airflow” • the section called “Tentang jaringan”	26 Mei 2026
Versi Apache Airflow baru	Amazon MWAA sekarang mendukung Apache Airflow v3.2.1. Pembaruan ini mencakup informasi tentang paket penyedia yang diperbaru	19 Mei 2026

	i, detail tentang penggunaan Apache Airflow v3.2.1 di Amazon MWAA, bagian baru tentang opsi konfigurasi Apache Airflow yang tidak didukung, dan contoh kode pembersihan basis data baru untuk Apache Airflow v3. • Versi • the section called “Version-specific paket penyedia” • the section called “Konfigurasi yang tidak didukung” • the section called “Pembersihan basis data Aurora PostgreSQL”	
Versi Apache Airflow baru	Amazon MWAA sekarang mendukung Apache Airflow v2.11.0. Pembaruan ini mencakup informasi tentang paket penyedia yang diperbarui, dan detail tentang penggunaan Apache Airflow v2.11.0 di Amazon MWAA. • Versi • the section called “Version-specific paket penyedia”	Januari 7, 2026
Menambahkan dukungan Apache Airflow v3	Dokumentasi yang diperbarui untuk menyertakan dukungan untuk Apache Airflow v3 • Contoh kode • Versi	Oktober 1, 2025

Pembaruan IPv6	Menambahkan informasi tentang dukungan IPv6. • the section called “Tentang jaringan”	Agustus 26, 2025
Pembaruan lingkungan	Menambahkan catatan tentang `workerReplacementStrategy` perubahan dari `GRACEFUL` ke `FORCED` jika Anda melakukan pembaruan saat lingkungan Anda dalam `MAINTENANCE` status. • the section called “Perbarui lingkungan”	Agustus 6, 2025
Informasi penghentian versi	Topik yang diperbarui tentang penghentian versi untuk menyertakan pemberitahuan penghentian dan jadwal untuk Apache Airflow v2.4.3, Apache Airflow v2.5.1, dan Apache Airflow v2.6.3. • the section called “Apache Airflow versi usang”	Juni 24, 2025
Menambahkan kelas lingkungan baru: mw1.micro	Amazon MWAA sekarang menyediakan kelas lingkungan baru: mw1.micro. • the section called “Mengkonfigurasi kelas lingkungan” • the section called “Penyetelan kinerja untuk Apache Airflow”	November 19, 2024

Support untuk metode yang lebih sederhana untuk mengakses Apache Airflow REST API	Amazon MWAA sekarang menyediakan pendekatan yang disederhanakan untuk berinteraksi dengan Apache Airflow REST API menggunakan kredensil. AWS • the section called “Menggunakan Apache Airflow REST API” • the section called “Akses API Apache Airflow Rest”	Oktober 23, 2024
Versi Apache Airflow baru	Amazon MWAA sekarang mendukung Apache Airflow v2.10.1. Pembaruan ini mencakup informasi tentang paket penyedia yang diperbarui, dan detail tentang penggunaan Apache Airflow v2.10.1 di Amazon MWAA. • Versi • the section called “Version-specific paket penyedia”	September 26, 2024

Versi Apache Airflow baru	Amazon MWAA sekarang mendukung Apache Airflow v2.9.2. Pembaruan ini mencakup informasi tentang paket penyedia yang diperbarui, dan detail tentang penggunaan Apache Airflow v2.9.2 di Amazon MWAA. • Versi • the section called “Version-specific paket penyedia”	9 Juli 2024
Amazon MWAA mendukung konfigurasi nama domain webserver kustom	Amazon MWAA mendukung konfigurasi nama domain webserver kustom untuk lingkungan pribadi tanpa akses internet. Pembaruan ini mencakup topik baru berikut yang menjelaskan pengaturan domain kustom baru. • the section called “Menyiapkan domain khusus”	Juni 18, 2024

Amazon MWAA mendukung penskalaan otomatis server web dan API REST Apache Airflow	Amazon MWAA sekarang mendukung penskalaan otomatis server web serta kemampuan untuk mengakses dan menggunakan Apache Airflow REST API. • the section called “Mengkonfigurasi penskalaan otomatis server web” • the section called “Menggunakan Apache Airflow REST API”	16 Mei 2024
Deskripsi perilaku penskalaan otomatis yang lebih baik	Memperbarui topik berikut untuk mencerminkan perilaku penskalaan otomatis Amazon MWAA yang baru saat pekerja mengambil tugas baru saat pekerja Fargate menurunkan skala. • the section called “Mengkonfigurasi penskalaan otomatis pekerja”	10 Mei 2024
Support untuk ukuran instans yang lebih besar	Amazon MWAA sekarang mendukung dua opsi ukuran instans yang lebih besar untuk beban kerja yang lebih besar:, dan `mw1.xlarge` `mw1.2xlarge` • the section called “Kemampuan lingkungan”	16 April 2024

Versi Apache Airflow baru	Amazon MWAA sekarang mendukung Apache Airflow v2.8.1. Pembaruan ini mencakup informasi tentang paket penyedia yang diperbarui, dan detail tentang penggunaan Apache Airflow v2.8.1 di Amazon MWAA. • Versi • the section called “Version-specific paket penyedia”	Februari 22, 2024
Dukungan untuk Amazon VPC bersama	Amazon MWAA mendukung pembuatan lingkungan lintas akun untuk organisasi yang menggunakan OpenSearch Layanan Amazon untuk mengelola sumber daya Amazon MWAA menggunakan VPC Amazon bersama pusat di akun pemilik. Sebagai bagian dari peluncuran ini, Amazon MWAA memungkinkan Anda memilih untuk membuat, dan mengelola, titik akhir VPC Amazon Anda sendiri. • the section called “Mengelola titik akhir VPC Amazon Anda sendiri”	15 November 2023

Versi Apache Airflow baru	Amazon MWAA sekarang mendukung Apache Airflow v2.7.2. Pembaruan ini mencakup informasi tentang paket penyedia yang diperbarui, dan detail tentang penggunaan Apache Airflow v2.7.2 di Amazon MWAA. • Versi • the section called “Version-specific paket penyedia”	6 November 2023
Versi Apache Airflow baru	Amazon MWAA sekarang mendukung Apache Airflow v2.6.3. Pembaruan ini mencakup informasi tentang paket penyedia yang diperbarui, dan detail tentang penggunaan Apache Airflow v2.6.3 di Amazon MWAA, • Versi	9 Agustus 2023
Informasi penghentian versi	Topik yang diperbarui tentang penghentian versi untuk menyertakan pemberitahuan penghentian dan jadwal untuk Apache Airflow v2.0.2 dan Apache Airflow v2.2.2. • the section called “Apache Airflow versi usang”	31 Juli 2023

Topik baru dan kasus penggunaan	Amazon MWAA mendukung peningkatan versi minor. Pembaruan ini mencakup topik baru berikut yang menjelaskan cara meningkatkan lingkungan dan memastikan sumber daya alur kerja Anda kompatibel dengan versi Apache Airflow yang Anda upgrade ke: • the section called "Mengubah versi"	Juni 5, 2023
Topik yang diperbarui	Kebijakan IAM yang dikelola pelanggan yang diperbarui yang memberikan pengguna konsol penuh dan akses API ke Amazon MWAA. Pembaruan menjelaskan mengapa Anda harus memberikan izin `iam:PassRole` untuk mengizinkan pengguna meneruskan peran ke Amazon MWAA. Amazon MWAA menggunakan izin ini untuk melakukan tindakan atas nama pengguna. • the section called "Mengakses lingkungan Amazon MWAA"	12 April 2023

Panduan baru	Topik yang diperbarui tentang konfigurasi AWS Secrets Manager sebagai backend untuk Amazon MWAA untuk memberikan panduan tentang penggunaan pola pencarian. Menggunakan pola pencarian mempersempit rahasia yang dicari Apache Airflow dan mengurangi jumlah panggilan API yang dilakukan Amazon MWAA ke Secrets Manager untuk mengambil koneksi dan variabel. Ini mengurangi biaya yang terkait dengan penggunaan Secrets Manager sebagai backend. • Buat backend Secrets Manager sebagai opsi konfigurasi Apache Airflow	12 April 2023
Versi Apache Airflow baru	Amazon MWAA sekarang mendukung Apache Airflow v2.5.1. Pembaruan ini mencakup informasi tentang paket penyedia yang diperbarui, dan detail tentang penggunaan Apache Airflow v2.5.1 di Amazon MWAA, • Versi	11 April 2023

Topik baru dan kasus penggunaan	Menambahkan topik baru tentang menggunakan skrip startup dengan lingkungan Amazon MWAA. Topik ini menjelaskan konfigurasi skrip startup untuk lingkungan yang ada, menggunakannya untuk menginstal runtime Linux, dan mengatur variabel lingkungan. • the section called "Menggunakan skrip startup"	3 April 2023
Bagian yang diperbarui pada akses server web pribadi	Memperbarui topik berikut pada akses server web pribadi. Pembaruan menjelaskan bahwa, di lingkungan dengan akses server web pribadi, Anda harus menggunakan arsip roda Python (`.whl`) untuk mengemas, dan menginstal, dependensi. • Mode akses server web pribadi	Februari 24, 2023

Menambahkan informasi tentang versi Apache Airflow yang tidak digunakan lagi	Memperbarui topik Versi dengan informasi baru tentang cara Amazon MWAA mengelola menghentikan versi Apache Airflow. Menghapus bagian tentang upgrade ke versi Apache Airflow yang lebih baru, dan bagian yang menjelaskan perubahan antara Apache Airflow v1 dan Apache Airflow v2. Untuk informasi selengkapnya tentang migrasi ke versi baru Apache Airflow, lihat Panduan Migrasi Amazon MWAA. • the section called “Apache Airflow versi usang” • the section called “Dukungan dan FAQ versi Apache Airflow”	17 Februari 2023

Perbaikan dalam metrik penampung Amazon MWAA	Memperbarui topik metrik kontainer, dan menghapus satu set metrik yang salah yang tidak ada dalam dimensi. `Cluster` Menambahkan bagian tambahan yang menjelaskan bagaimana Anda dapat mengevaluasi jumlah pekerja tambahan yang digunakan lingkungan pada waktu tertentu dengan membuat grafik `CPUUtilization` atau `MemoryUtilization` metrik untuk `AdditionalWorker` komponen tersebut, dan menyetel jenis statistik. `Sample Count` • the section called “Mengevaluasi jumlah pekerja tambahan dan wadah server web”	20 Januari 2023

Versi Apache Airflow baru	Amazon MWAA sekarang mendukung Apache Airflow v2.4.3. Pembaruan ini mencakup informasi tentang paket penyedia yang diperbaru i, detail tentang penggunaa n Apache Airflow v2.4.3 di Amazon MWAA, dan informasi gabungan tentang fitur mana yang didukung di setiap versi Apache Airflow di Amazon MWAA. • Versi	5 Januari 2023
Topik terbaru tentang peran terkait layanan	Informasi terbaru tentang peran terkait layanan yang digunakan Amazon MWAA untuk membuat dan mengelola AWS sumber daya atas nama Anda, termasuk informasi tentang cara menghapus peran terkait layanan saat Anda tidak lagi membutuhkannya. Ini termasuk kebijakan izin peran terkait layanan yang diperbaru i yang memungkinkan Amazon MWAA mempublik asikan metrik tambahan di namespace. CloudWatch `AWS/MWAA` • the section called “Service-linked peran”	18 November 2022

Topik baru tentang metrik layanan	Menambahkan topik baru yang menjelaskan metrik layanan yang dipancark an oleh Amazon MWAA di namespace. `AWS/MWAA` Ini termasuk penjadwal metrik klaster Amazon ECS, pekerja, dan server web, metrik Amazon SQS untuk antrian yang memungkinkan Amazon MWAA memisahkan penjadwal dan pekerja, serta metrik Amazon RDS untuk database metadata. • the section called “Metrik kontainer, antrian, dan basis data”	18 November 2022
Topik baru	Menambahkan panduan baru tentang memodifikasi file kendala untuk menentuka n versi baru paket penyedia yang akan digunakan dengan lingkungan Amazon MWAA Anda. • the section called “Kendala berkas”	18 November 2022
Entri FAQ yang diperbarui	Informasi terbaru terkait dengan kelayakan HIPAA Amazon MWAA. • the section called “Kepatuha n HIPAA”	15 November 2022

Topik baru	Menambahkan topik baru tentang penggunaan `aws:SourceArn` dan kunci konteks kondisi `aws:SourceAccount` global dalam kebijakan kepercayaan peran eksekusi Amazon MWAA, untuk mencegah deputi kebingungan lintas layanan. • the section called “Cross-service pencegahan wakil bingung”	21 Oktober 2022
Kode sampel baru	Menambahkan instruksi yang diperbarui dan contoh kode DAG yang menulis OS-level metrik khusus. CloudWatch • the section called “Menggunakan DAG untuk menulis metrik khusus”	13 September 2022
Kode sampel baru	Menambahkan instruksi yang diperbarui dan contoh kode AWS Lambda Python baru yang mengambil token CLI Apache Airflow, lalu memanggil DAG di lingkungan Amazon MWAA tertentu. • the section called “Memohon DAGs dengan Lambda”	12 September 2022

Diagram arsitektur baru	Menambahkan diagram arsitektur baru yang menunjukkan lingkungan Amazon MWAA dengan server web publik dan pribadi. • the section called “Mode akses Apache Airflow”	12 September 2022
Kode sampel baru	Menambahkan instruksi yang diperbarui dan contoh kode DAG baru yang mengambil token CLI Apache Airflow, lalu memanggil DAG lain di lingkungan MWAA Amazon yang berbeda. • the section called “Memohon DAGs di lingkungan yang berbeda”	Agustus 16, 2022
Kode sampel baru	Menambahkan instruksi yang diperbarui dan DAG baru yang menanyakan Aurora PostgreSQL lingkungan untuk informasi metadata, menulis hasilnya Ke file CSV dan menyimpan file di Amazon S3. • the section called “Mengekspor metadata lingkungan ke Amazon S3”	12 Agustus 2022

Kode sampel baru	Menambahkan instruksi yang diperbarui dan DAG baru yang menyegarkan AWS CodeArtifact token saat runtime dan menyimpan hasilnya di Amazon S3. • the section called "Menyegarkan AWS CodeArtifact token saat runtime"	3 Agustus 2022
Kode sampel baru	Menambahkan instruksi yang diperbarui dan contoh kode DAG untuk menggunakan `ECSOperator` di Amazon MWAA. • the section called "Menggunakan `ECSOperator`"	26 Juli 2022
Kode sampel baru	Menambahkan instruksi yang diperbarui dan contoh kode DAG untuk menggunakan `SSHOperator` di Amazon MWAA. • the section called "Menggunakan `SSHOperator`"	15 Juli 2022

Kode sampel baru	Menambahkan instruksi baru dan contoh kode DAG untuk menggunakan dbt Postgres dengan Amazon MWAA. • the section called “Menggunakan dbt dengan Amazon MWAA”	Juni 17, 2022
Topik baru dan kasus penggunaan	Menambahkan instruksi baru dan contoh kode DAG untuk menginstal dependensi menggunakan file roda Python untuk lingkungan Amazon MWAA dengan akses publik dan pribadi. • Mengelola dependensi menggunakan roda Python	Mei 13, 2022
Topik baru dan kasus penggunaan	Menambahkan panduan baru untuk memilih metrik Apache Airflow mana yang dikirimkan Amazon MWAA. CloudWatch • Memilih metrik Apache Airflow mana yang dilaporkan	19 April 2022

Panduan baru	Amazon MWAA menawarkan panduan migrasi untuk memigrasi alur kerja Apache Airflow dari penerapan yang dikelola sendiri, serta lingkungan Amazon MWAA yang ada. • Panduan Migrasi Amazon MWAA	7 Maret 2022
Topik baru dan kasus penggunaan	Menambahkan praktik terbaik keamanan baru untuk bekerja dengan Apache Airflow, termasuk solusi untuk mendeteksi perubahan pada hak istimewa pengguna Apache Airflow. • the section called “Praktik terbaik keamanan di Apache Airflow”	18 Februari 2022
Kode sampel baru	Menambahkan contoh kode baru untuk membuat DAG yang sadar zona waktu menggunakan Pendulum, dan mengklarifikasi cara menggunakan plugin khusus untuk mengubah zona waktu di mana log Apache Airflow dibuat. • the section called “Mengubah zona waktu DAG”	Februari 11, 2022

Peluncuran Apache Airflow v2.2.2	Alur Kerja Terkelola Amazon untuk Apache Airflow sekarang mendukung Apache Airflow v2.2.2. Dimulai dengan v2.2, Amazon MWAA akan menginstal paket Python dan plugin khusus langsung di server web Apache Airflow yang memungkinkan Anda fleksibilitas yang lebih besar untuk mengelola lingkungan Anda. Untuk informasi lebih lanjut, lihat yang berikut ini. • Versi Apache Airflow di Amazon Managed Workflows for Apache Airflow. • Apache Airflow v2.2.2 changelog di situs dokumentasi Apache Airflow.	27 Januari 2022
Tutorial baru	Menambahkan tutorial baru yang menunjukkan pembuatan peran Apache Airflow kustom baru, dan menetapkan peran tersebut ke pengguna Apache Airflow yang dipetakan dari IAM untuk membatasi akses pengguna ke subset DAG tertentu. • Tutorial: Membatasi akses pengguna Amazon MWAA ke subset DAGs	8 Desember 2021

Perbaikan	Memperbaiki rekomendasi praktik terbaik untuk menetapkan nilai `scheduler.min_file_process_interval` untuk mengoptimalkan penggunaan CPU. Menambahkan contoh kebijakan IAM yang memberikan akses ke sumber daya Secrets Manager dalam peran eksekusi. Menambahkan topik pemecahan masalah saat menggunakan kunci kondisi Secrets Manager. • Penyetelan kinerja bagaimana penjadwal mem-parsing DAG • Berikan Amazon MWAA izin untuk mengakses kunci rahasia Secrets Manager • Mengonfigurasi kunci kondisi dalam peran eksekusi Amazon MWAA untuk Secrets Manager	22 November 2021

Kode sampel baru	Menambahkan contoh kode baru berikut untuk memodifikasi zona waktu di mana DAG diproses menggunakan plugin khusus, dan topik pemecahan masalah baru untuk menjalankan perintah Apache `dags backfill` Airflow CLI dari dalam operator bash. • the section called "Mengubah zona waktu DAG" • Isi ulang perintah CLI menggunakan operator bash	November 1, 2021
Perbaikan	Memperbaiki masalah dalam contoh kode operator Amazon ECS, dan mengklarifikasi izin tambahan yang diperlukan dalam peran eksekusi Amazon MWAA untuk memungkinkan lingkungan mengakses grup log tugas Amazon ECS di Log. CloudWatch • Izin operator Amazon ECS.	26 Oktober 2021

Kode sampel baru	Menambahkan contoh kode baru yang menanyakan database Aurora PostgreSQL untuk informasi yang relevan dengan DAG berjalan dan menulis hasilnya ke file yang disimpan di Amazon S3. `CSV` • the section called “Mengekspor metadata lingkungan ke Amazon S3”.	1 Oktober 2021
Perbaikan	Informasi yang dikoreksi tentang cara Amazon MWAA secara otomatis menyinkronkan objek baru dan yang diubah dari bucket Amazon S3 target Anda ke penjadwal dan pekerja Anda. • Cara kerja folder DAG.	1 Oktober 2021
Sekarang didukung	Amazon MWAA sekarang mendukung paket penyedia tambahan untuk Apache Airflow 2.0+. Untuk mempelajari lebih lanjut tentang paket yang didukung, lihat berikut ini:	24 September 2021

Perintah dan prosedur baru	Menambahkan panduan tambahan dan contoh AWS CLI perintah untuk membuat titik akhir gateway Amazon S3 saat menggunakan VPC Amazon tanpa akses internet: • Membuat jaringan VPC Amazon tanpa akses internet.	24 September 2021
Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Menambahkan contoh kode baru yang menggunakan operator Amazon Elastic Container Service dithe section called “Mengguna kan `ECSOperator` ”. • Menambahkan topik pemecahan masalah baru untuk masalah dalam mengonfigurasi plugin Apache Airflow di. the section called “Plugin”	September 19, 2021

Wilayah baru yang didukung	Amazon MWAA sekarang tersedia di wilayah berikut: • Asia Pacific (Mumbai) - ap-south-1 • Asia Pacific (Seoul) - ap-northeast-2 • Europe (London) - eu-west-2 • Europe (Paris) - eu-west-3 • Canada (Central) - ca-central-1 • South America (São Paulo) - sa-east-1 Untuk informasi selengkapnya tentang ketersediaan wilayah dan titik akhir layanan, lihat berikut ini: • Titik akhir dan kuota Amazon MWAA di. Referensi Umum AWS	31 Agustus 2021
Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Memperbarui kebijakan sampel agar Amazon MWAA dapat mengambil setelan Amazon S3 () tingkat akun. `s3:GetAccountPublicAccessBlock` Peran eksekusi Amazon MWAA	Agustus 27, 2021

Perbaikan	Menambahkan perubahan berikut: • Memperbaiki CloudForm ation template untuk menggunakan aturan masuk referensi sendiri untuk grup keamanan di. Buat jaringan VPC • Memperbaiki CloudForm ation template untuk menggunakan aturan masuk referensi sendiri untuk grup keamanan di. Tutorial mulai cepat untuk Alur Kerja Terkelola Amazon untuk Apache Airflow	Agustus 27, 2021
Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Menambahkan dekorator DAG ke daftar apa yang didukung untuk Apache Airflow v2.0.2. Versi Apache Airflow di Amazon Managed Workflows for Apache Airflow	Agustus 20, 2021

Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Ditambahkan `celery.sync_parallelism` use case kePenyetelan kinerja untuk Apache Airflow di Amazon MWAA. • Menambahkan titik akhir layanan ke halaman kuota dan mengubah nama menjadi. Alur Kerja Terkelola Amazon untuk titik akhir dan kuota layanan Apache Airflow • Prasyarat jaringan yang diklarifikasi berdasarkan umpan balik pengguna di. Memulai dengan Amazon Managed Workflows for Apache Airflow • Pindah `dags list-runs` dan `dags next-execution` ke perintah CLI Aliran Udara yang tidak didukung di. Referensi perintah CLI Apache Airflow	13 Agustus 2021

Kode sampel baru	Menambahkan perubahan berikut: • Menambahkan contoh bash untuk mengatur, mendapatkan, atau menghapus variabel Apache Airflow v2.0.2 di. Referensi perintah CLI Apache Airflow • Menambahkan dependensi Apache Airflow v2.0.2 dan contoh koneksi Airflow ke. Menggunakan Amazon MWAA dengan Amazon RDS untuk Microsoft SQL Server	13 Agustus 2021
Perbaikan	Menambahkan perubahan berikut: • Memperbaiki contoh kode Python berdasarkan umpan balik pengguna di. Membuat koneksi SSH menggunakan `SSHOperator`	13 Agustus 2021

Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Pindah `variables set` ke perintah CLI Aliran Udara yang didukung di. Referensi perintah CLI Apache Airflow • Menambahkan ringkasan Apa yang diubah di v2.0.2 dari halaman versi Airflow menjadi Menginstal dependensi Python berdasarkan umpan balik pengguna. • Menambahkan ringkasan Apa yang diubah di v2.0.2 dari halaman versi Airflow menjadi Referensi perintah CLI Apache Airflow berdasarkan umpan balik pengguna. • Menambahkan ringkasan Apa yang diubah di v2.0.2 dari halaman versi Airflow menjadi Ikhtisar jenis koneksi berdasarkan umpan balik pengguna. • Menambahkan ringkasan Apa yang diubah di v2.0.2 dari halaman versi Airflow menjadi Menginstal plugin kustom berdasarkan umpan balik pengguna. • Menambahkan ringkasan Apa yang diubah di v2.0.2	6 Agustus 2021

	dari halaman versi Airflow menjadi Menambahkan atau memperbarui DAGs berdasarkan umpan balik pengguna.	
Kode sampel baru	Menambahkan perubahan berikut: • Menambahkan kode sampel Apache Airflow v2.0.2 ke. Menggunakan DAG untuk mengimpor variabel di CLI • Menambahkan kode sampel Apache Airflow v2.0.2 ke. Memanggil DAGs dengan fungsi Lambda	6 Agustus 2021
Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Menambahkan topik pemecahan masalah untuk 'Saya tidak dapat menemukan koneksi saya di UI Aliran Udara' di. Memecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow • Menambahkan daftar Amazon VPC Amazon yang didukung Amazon MWAA ke. Tentang jaringan di Amazon MWAA	29 Juli 2021

Perbaikan	Menambahkan perubahan berikut: • Memperbaiki contoh kode Python berdasarkan umpan balik pengguna untuk mencetak token login web di. Buat token akses server web Apache Airflow • Memperbaiki topik koneksi Snowflake berdasarkan umpan balik pengguna untuk menggunakan satu kutipan untuk parameter gudang di. Memecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow	29 Juli 2021
Menghapus atau memindahkan topik	Menambahkan perubahan berikut: • Merestrukturisasi halaman yang ada untuk menyertakan semua halaman dokumentasi pemantauan dan metrik. Pemantauan dan metrik untuk Alur Kerja Terkelola Amazon untuk Apache Airflow • Pindah Metrik lingkungan Apache Airflow di CloudWatch ke menu navigasi pemantauan dan metrik.	23 Juli 2021

Panduan baru	Menambahkan perubahan berikut: • DibuatPaket penyedia Apache Airflow diinstal di lingkungan Amazon MWAA. • DibuatIkhtisar pemantauan di Amazon MWAA. • DibuatMengakses log audit di AWS CloudTrail. • DibuatMengakses log Aliran Udara di Amazon CloudWatch.	23 Juli 2021
Perbaikan	Menambahkan perubahan berikut: • Memperbaiki contoh kode Python berdasarkan umpan balik pengguna untuk menghasilkan string koneksi Aliran Udara dalam urutan yang benar dan menambahkan parameter port di. Mengkonfigurasi koneksi Apache Airflow menggunakan rahasia AWS Secrets Manager • Menambahkan langkah untuk menginstal paket unzip secara lokal berdasarkan umpan balik pengguna di. Membuat plugin khusus dengan Oracle	23 Juli 2021

Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Menambahkan topik untuk Operator AWS DMS di Amazon MWAA pertanyaan yang sering diajukan. • Menambahkan topik pemecahan masalah untuk kesalahan log jarak jauh ke. Memecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow • Pindah `variables set` ke perintah CLI Aliran Udara yang tidak didukung di. Referensi perintah CLI Apache Airflow	16 Juli 2021

Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Menambahkan langkah-langkah berurutan untuk membuat file requirements.txt berdasarkan umpan balik pengguna di Menginstal dependensi Python. • Menambahkan langkah-langkah berurutan untuk membuat file plugins.zip berdasarkan umpan balik pengguna di Menginstal plugin kustom. • Menambahkan tautan referensi silang di seluruh panduan pengguna ke panduan referensi API di panduan Referensi API Alur Kerja Terkelola Amazon untuk Apache Airflow. • Menambahkan topik mengapa plugin tidak tercantum di menu Admin Airflow 2.0 > Plugin di. Amazon MWAA pertanyaan yang sering diajukan	Juli 9, 2021
Panduan baru	Menambahkan perubahan berikut: • Dibuat Menghapus file di Amazon S3.	Juli 9, 2021

Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Menambahkan daftar nilai yang didukung diMenggunakan kunci yang dikelola pelanggan untuk enkripsi. • Memperbarui dan mengklarifikasi contoh untuk URL repo pribadi berdasarkan umpan balik pengguna di. Mengelola dependensi Python di requirements.txt	2 Juli 2021
Kode sampel baru	Menambahkan perubahan berikut: • Menambahkan kode sampel Apache Airflow v1.10.12 untuk menggunakan kunci pribadi untuk koneksi SSH AWS Secrets Manager di. Membuat koneksi SSH menggunakan `SSHOperator`	2 Juli 2021
Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Ditambahkan StartedTaskInstances dan FinishedTaskInstances metrik keMetrik lingkungan Apache Airflow di CloudWatch.	25 Juni 2021

Kode sampel baru	Menambahkan perubahan berikut: • Menambahkan kode sampel Apache Airflow v2.0.2 di. Menggunakan Amazon MWAA dengan Amazon EKS	25 Juni 2021
Panduan baru	Menambahkan perubahan berikut: • DibuatPenyetelan kinerja untuk Apache Airflow di Amazon MWAA.	25 Juni 2021

Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Ditambahkan `connections add` dan `connections delete` ke perintah Apache Airflow v2.0.2 CLI yang didukung di. Referensi perintah CLI Apache Airflow • Ditambahkan bahwa versi terbaru yang tersedia CloudFormation adalah Apache Airflow v2.0.2 di. Tutorial mulai cepat untuk Alur Kerja Terkelola Amazon untuk Apache Airflow • Menambahkan pertanyaan untuk menyimpan data sementara pada pekerja Apache Airflow ke. Amazon MWAA pertanyaan yang sering diajukan • Menambahkan topik untuk kesalahan 'Executor melaporkan instance tugas %s finished' ke. Memecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow • Menambahkan topik untuk log 'server menutup koneksi secara tak terduga'. Memecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow	18 Juni 2021

- Menambahkan contoh untuk menjalankan perintah CLI pada terowongan SSH ke host bastion ke. Membuat token CLI Apache Airflow
- Menambahkan topik untuk nama pengguna yang dibuat secara acak ke. Memecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow
- Menambahkan topik untuk kesalahan 503 saat menjalankan DAG di Memecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow CLI ke.
- Menambahkan topik untuk plugin khusus di Apache Airflow v2.0.2 yang memerlukan opsi `core.lazy_load_plugins : False` konfigurasi Airflow untuk memuat plugin pada awal setiap proses Airflow untuk mengganti pengaturan default versi ke. Menggunakan opsi konfigurasi Apache Airflow di Amazon MWAA
- Menambahkan langkah opsi konfigurasi Airflow untuk kode sampel plugin Apache Airflow v2.0.2 di. Membuat plugin khusus

	dengan Apache Hive dan Hadoop • Menambahkan langkah opsi konfigurasi Airflow untuk kode sampel plugin Apache Airflow v2.0.2. • Menambahkan langkah opsi konfigurasi Airflow untuk kode sampel plugin Apache Airflow v2.0.2 di. Membuat plugin khusus untuk Apache Airflow PythonVirtualenvOperator • Menambahkan langkah opsi konfigurasi Airflow untuk kode sampel plugin Apache Airflow v2.0.2 di. Membuat plugin khusus dengan Oracle	
Kode sampel baru	Menambahkan perubahan berikut: • Menambahkan kode sampel untuk koneksi Apache Airflow Snowflake di. Menggunakan kunci rahasia AWS Secrets Manager untuk koneksi Apache Airflow Snowflake	18 Juni 2021

Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Menambahkan panduan enkripsi sisi server ke. Buat bucket Amazon S3 untuk Amazon MWAA • Menambahkan backend rahasia untuk Apache Airflow v2.0.2 ke. Mengkonfigurasi koneksi Apache Airflow menggunakan rahasia AWS Secrets Manager • Menambahkan pertanyaan untuk permintaan peningkatan kuota pekerja Apache Airflow ke. Amazon MWAA pertanyaan yang sering diajukan • Pertanyaan tambahan untuk metrik mana yang digunakan untuk menentukan apakah akan menskalakan pekerja Apache Airflow. Amazon MWAA pertanyaan yang sering diajukan • Menambahkan pertanyaan untuk membuat metrik khusus CloudWatch keAmazon MWAA pertanyaan yang sering diajukan. • Menambahkan langkah-langkah untuk mengaktif	2 Juni 2021

	kan alamat IP pribadi untuk titik akhir antarmuka VPC Amazon S3 untuk VPC dengan perutean pribadi. Membuat titik akhir layanan VPC yang diperlukan di VPC Amazon dengan perutean pribadi • Menambahkan opsi untuk mengatur Terowongan SSH menggunakan penerusan port lokal. Tutorial: Mengkonfigurasi akses jaringan pribadi menggunakan Linux Bastion Host	
Kode sampel baru	Menambahkan perubahan berikut: • Menambahkan kode sampel untuk DAG yang menanyakan database metadata Amazon Aurora PostgreSQL dan menerbitkan metrik khusus ke Amazon di. CloudWatch Menggunakan DAG untuk menulis metrik khusus CloudWatch	2 Juni 2021

Panduan baru	Menambahkan perubahan berikut: • Membuat panduan tentang cara menggunakan template koneksi secara bergantian di Apache Airflow UI di. Ikhtisar jenis koneksi	2 Juni 2021
Perbaikan	Menambahkan perubahan berikut: • Menambahkan titik akhir VPC Apache Airflow ke template CloudFormation di Opsi tiga: Membuat jaringan VPC tanpa akses internet ke. Buat jaringan VPC	2 Juni 2021

Peluncuran Apache Airflow v2.0.2	Peluncuran ketersediaan umum Apache Airflow v2.0.2. • DibuatVersi Apache Airflow di Amazon Managed Workflows for Apache Airflow. • DibuatMetrik lingkungan Apache Airflow di CloudWatch. • Menambahkan tautan khusus versi untuk Apache Airflow v2.0.2 ke. Menggunakan opsi konfigurasi Apache Airflow di Amazon MWAA • Menambahkan panduan khusus versi Apache Airflow v2.0.2 ke. Menginstal dependensi Python • Menambahkan panduan khusus versi Apache Airflow v2.0.2 ke. Mengelola dependensi Python di requirements.txt • Menambahkan plugin sampel Apache Airflow v2.0.2 ke. Menginstal plugin kustom • Menambahkan kode sampel Apache Airflow v2.0.2 ke. Pembersihan basis data Aurora PostgreSQL di lingkungan Amazon MWAA	26 Mei 2021

	• Menambahkan kode sampel Apache Airflow v2.0.2 ke. Menggunakan kunci rahasia AWS Secrets Manager untuk koneksi Apache Airflow • Menambahkan kode sampel Apache Airflow v2.0.2 ke. Membuat plugin khusus untuk Apache Airflow PythonVirtualenvOperator • Menambahkan perintah Apache Airflow v2.0.2 ke. Referensi perintah CLI Apache Airflow • Menambahkan skrip Apache Airflow v2.0.2 ke. Membuat token CLI Apache Airflow • Menambahkan catatan bahwa Amazon MWAA menggunakan versi Apache Airflow terbaru secara default ke. Buat lingkungan Amazon MWAA	
Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Menambahkan panduan untuk memecahkan masalah tugas Aliran Udara yang macet atau tidak berjalan. Memecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow	14 Mei 2021

Perbaikan	Menambahkan perubahan berikut: • Kami telah memperbarui kode plugin sampel untuk menggunakan versi Java terbaru diMembuat plugin khusus dengan Apache Hive dan Hadoop. Sebelumnya, ituos.environ["JAVA_HOME"]="/usr/lib/jvm/jre-1.8.0-openjdk-1.8.0.272.b10-1.amzn2.0.1.x86_64" .	12 Mei 2021
Menghapus atau memindahkan topik	Menambahkan perubahan berikut: • Memindahkan topik Memecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow ke halaman baru berdasarkan kategori.	10 Mei 2021
Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Menambahkan ikhtisar bucket Amazon S3 ke. Bekerja dengan DAGs di Amazon MWAA	10 Mei 2021

Menghapus atau memindahkan topik	Menambahkan perubahan berikut: • Pindah Mengakses Apache Airflow ke navigasi tingkat utama, dan menambahkan halaman untukBuat token akses server web Apache Airflow,Membuat token CLI Apache Airflow, dan. Referensi perintah CLI Apache Airflow	7 Mei 2021
Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Menambahkan tautan khusus versi ke panduan referensi Apache Airflow untuk semua perintah CLI Aliran Udara yang didukung dan tidak didukung di. Referensi perintah CLI Apache Airflow • Menambahkan tautan khusus versi ke panduan referensi Apache Airflow untuk semua opsi konfigurasi. Menggunakan opsi konfigurasi Apache Airflow di Amazon MWAA • Menambahkan utilitas Amazon MWAA CLI ke. Mengelola dependensi Python di requirements.txt	7 Mei 2021

Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Menambahkan contoh datar dan bersarang untuk cara menyusun plugins.zip diMenginstal plugin kustom. • Menambahkan utilitas Amazon MWAA CLI keMenambahkan atau memperbarui DAGs,, Menginstal plugin kustom dan halaman. Menginstal dependensi Python • Konten yang direstruk turisasi menjadi ikhtisar, unggah ke Amazon S3, dan instal di bagian Amazon MWAA berdasarkan umpan balik Menginstal plugin kustom pengguna di, dan halaman. Menginstal dependensi Python • Menambahkan contoh kasus penggunaan untuk membuat dan melampirk an titik akhir VPC yang diperlukan ke VPC Amazon yang ada tanpa akses internet. Tentang jaringan di Amazon MWAA	30 April 2021

Kode sampel baru	Menambahkan perubahan berikut: • Menambahkan kode sampel yang menggunakan kunci rahasia di Secrets Manager untuk variabel Apache Airflow di. Menggunakan kunci rahasia AWS Secrets Manager untuk variabel Apache Airflow	30 April 2021
Panduan baru	Menambahkan perubahan berikut: • DibuatMembuat titik akhir layanan VPC yang diperlukan di VPC Amazon dengan perutean pribadi.	30 April 2021
Perbaikan	Menambahkan perubahan berikut: • Kami telah memperbarui `core.default_ui_timezone` ke `webserver.default_ui_timezone` dalamMenggunakan opsi konfigurasi Apache Airflow di Amazon MWAA.	30 April 2021

Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Menambahkan langkah-langkah Windows (PuTTY) untuk terowongan SSH ke. Tutorial: Mengkonfigurasi akses jaringan pribadi menggunakan Linux Bastion Host • Ditambahkan topik untuk`apache-airflow-providers-amazon`, yang hanya kompatibel dengan Apache Airflow 2.0 untuk. Memecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow	23 April 2021
Kode sampel baru	Menambahkan perubahan berikut: • Menambahkan kode sampel yang menggunakan kunci rahasia di Secrets Manager untuk koneksi Apache Airflow. Menggunakan kunci rahasia AWS Secrets Manager untuk koneksi Apache Airflow	23 April 2021

Panduan baru	Menambahkan perubahan berikut: • DibuatTentang jaringan di Amazon MWAA. • DibuatKeamanan di VPC Anda di Amazon MWAA. • DibuatMengelola akses ke titik akhir VPC Amazon khusus layanan di Amazon MWAA.	23 April 2021

Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Menambahkan CloudFormation template baru untuk membuat jaringan VPC Amazon tanpa akses internet. Buat jaringan VPC • Menambahkan tutorial baru untuk membuat AWS Client VPN inTutorial: Mengkonfigurasi akses jaringan pribadi menggunakan AWS Client VPN. • Mengubah nama halaman akses Jaringan ke mode akses Apache Airflow berdasarkan umpan balik pengguna, dan dokumen yang disederhanakan. Mode akses Apache Airflow • Dokumen yang disederhanakan untuk hanya menyertakan informasi dan templat Amazon VPC yang memulai berdasarkan umpan balik pengguna. Buat jaringan VPC • Menambahkan solusi BigQuery operator ke. Memecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow • Menambahkan praktik terbaik file batasan Apache	16 April 2021

	Airflow v1.10.12. Menginstal dependensi Python	
Kode sampel baru	Menambahkan perubahan berikut: • Ditambahkan kode sampel untuk membuat plugin kustom menggunakan Oracle diMembuat plugin khusus dengan Oracle. • Ditambahkan kode sampel untuk membuat plugin kustom yang menghasilkan variabel lingkungan runtime. •	16 April 2021

Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Menambahkan topik untuk persyaratan aturan referensi mandiri pada grup keamanan VPC ke. Amazon MWAA pertanyaan yang sering diajukan • Ditambahkan direktori plugin kustom dan batas ukuran untukMenginstal plugin kustom. • Menambahkan direktori persyaratan dan batas ukuran keMenginstal dependensi Python. • Mengklarifikasi opsi konfigurasi Apache Airflow untuk dan masuk. `foo.user` `foo.pass` Mengelola dependensi Python di requirements.txt • Menambahkan ikhtisar opsi konfigurasi keMenggunakan opsi konfigurasi Apache Airflow di Amazon MWAA.	9 April 2021

Kode sampel baru	Menambahkan perubahan berikut: • Ditambahkan kode sampel untuk membuat plugin kustom menggunakan `PythonVirtualenvOperator` in Membuat plugin khusus untuk Apache Airflow PythonVirtualenvOperator. • Ditambahkan kode sampel untuk membuat plugin kustom dengan Apache Hive dan Hadoop in. Membuat plugin khusus dengan Apache Hive dan Hadoop	9 April 2021
Perbaikan	Menambahkan perubahan berikut: • Kami telah memperbarui format untuk requirements.txt, dan menambahkan contoh yang kompatibel dengan Apache Airflow v1.10.12 in. Menginstal dependensi Python	31 Maret 2021

Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Menambahkan solusi untuk menghapus requirements.txt atau plugins.zip ke. Amazon MWAA pertanyaan yang sering diajukan • Menambahkan solusi bash untuk SSH di lingkungan ke. Amazon MWAA pertanyaan yang sering diajukan • Menambahkan topik untuk CloudTrail ResourceA lreadyExistsException kesalahan keMemecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow.	26 Maret 2021

Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Ditambahkan daftar AWS layanan yang digunakan untukPeran eksekusi Amazon MWAA. • Ditambahkan daftar AWS layanan yang digunakan untukService-linked peran untuk Amazon MWAA. • Menambahkan pertanyaan untuk versi Python 3.7 untuk Amazon MWAA ke. Amazon MWAA pertanyaan yang sering diajukan • Menambahkan pertanyaa n untuk `PythonVir tualenvOperator` untukAmazon MWAA pertanyaan yang sering diajukan. • Menambahkan skrip pemecahan masalah sebagai langkah selanjutn ya untuk semua topik yang terkait Dengan VPC dan konfigurasi lingkungan di. Memecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow • Mengklarifikasi dokumen bahwa benteng linux harus berada di Wilayah yang sama dengan lingkunga	19 Maret 2021

	n di. Tutorial: Mengkonfigurasi akses jaringan pribadi menggunakan Linux Bastion Host	
Panduan baru	Menambahkan perubahan berikut: • Membuat panduan koneksi Apache Airflow untuk at. AWS Secrets Manager Mengkonfigurasi koneksi Apache Airflow menggunakan rahasia AWS Secrets Manager • Membuat tutorial mulai cepat menggunakan CloudFormation template untuk membuat infrastruktur VPC Amazon, bucket Amazon S3, dan lingkungan Amazon MWAA di. Tutorial mulai cepat untuk Alur Kerja Terkelola Amazon untuk Apache Airflow	19 Maret 2021

Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Menambahkan topik pemecahan masalah bucket buat Amazon S3. Memecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow • Menambahkan langkah-langkah untuk membuat dan melampirkan kebijakan JSON kePeran eksekusi Amazon MWAA.	12 Maret 2021
Kode sampel baru	Menambahkan perubahan berikut: • Menambahkan kode sampel untuk menambahkan konfigurasi saat memicu DAG keMengakses Apache Airflow.	12 Maret 2021
Panduan baru	Menambahkan perubahan berikut: • Panduan praktik terbaik yang dibuat diMengelola dependensi Python di requirements.txt.	12 Maret 2021

Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Menambahkan topik Google/GCP/BigQuery pemecahan masalah ke. Memecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow • Menambahkan topik pemecahan masalah Cython ke. Memecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow • Menambahkan topik pemecahan masalah MySQL ke. Memecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow • Menambahkan topik pemecahan masalah kesalahan server web 5xx ke. Memecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow	5 Maret 2021

Sekarang didukung	Menambahkan perubahan berikut: • `backend_kwargs` Sebelumnya, tidak didukung AWS Secrets Manager dan Anda memerlukan solusi untuk mengganti panggilan fungsi Secrets Manager. Sekarang, `backend_kwargs` didukung. Lihat topik AWS Secrets Manager pemecahan masalah di. Memecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow	4 Maret 2021
Perbaikan	Menambahkan perubahan berikut: • Kami telah memperbarui ukuran setiap kelas lingkungan untuk mencerminkan GB yang sebenarnyaMengkonfigurasi kelas lingkungan Amazon MWAA.	4 Maret 2021

Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Menambahkan akses jaringan pribadi menggunakan kebijakan titik akhir VPC ke. Mode akses Apache Airflow • Menambahkan pemeriksaan tambahan untuk membuat topik pemecahan masalah lingkungan ke. Memecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow • Menambahkan langkah-langkah untuk mengakses log `requirements.txt` untukMenginstal dependensi Python.	26 Februari 2021

Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Menambahkan kasus penggunaan Apache Hive ke. Menginstal dependensi Python • Mengklarifikasi dokumen bahwa dependensi yang diperlukan untuk paket Apache Airflow perlu disertakan dalam file di. `requirements.txt` Menginstal dependensi Python • Ditambahkan Memperbarui topik pemecahan masalah requirements.txt ke. Memecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow	25 Februari 2021
Tutorial baru	Menambahkan perubahan berikut: • Menambahkan tutorial jaringan pribadi keTutorial: Mengkonfigurasi akses jaringan pribadi menggunakan Linux Bastion Host.	22 Februari 2021

Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Menambahkan konfigurasi jaringan pribadi dan publik keMode akses Apache Airflow. • Menambahkan kasus penggunaan grup pengembangan dan skenario pengguna kePeran eksekusi Amazon MWAA.	22 Februari 2021
Kode sampel baru	Menambahkan perubahan berikut: • Menambahkan contoh skrip Python untuk token login web dan token CLI ke. Mengakses Apache Airflow • Menambahkan kode sampel untuk memicu DAG di lingkungan lain keContoh kode untuk Alur Kerja Terkelola Amazon untuk Apache Airflow. • Menambahkan kode sampel untuk memicu DAG menggunakan fungsi Lambda ke. Memanggil DAGs dengan fungsi Lambda	22 Februari 2021

Perintah dan prosedur baru	Menambahkan perubahan berikut: • Menambahkan prosedur langkah demi langkah ke semua skrip diMengakses Apache Airflow.	22 Februari 2021
Kode sampel baru	Menambahkan perubahan berikut: • Contoh curl yang diperbaru i untuk token login web diMengakses Apache Airflow. • Menambahkan kode sampel untuk terhubung ke Amazon RDS Microsoft SQL Server ke. Menggunakan Amazon MWAA dengan Amazon RDS untuk Microsoft SQL Server	17 Februari 2021

Perintah dan prosedur baru	Menambahkan perubahan berikut: • Menambahkan AWS CLI perintah ke Bekerja dengan DAGs di Amazon MWAA halaman. • Apache Airflow tidak mendukung DAG serial dalam perintah CLI. Karena CLI berjalan di server web, yang tidak memiliki plugin atau persyaratan untuk alasan keamanan, lingkungan MWAA apa pun dengan plugins.zip atau requirements.txt tidak akan mendukung perintah ini. Memindahkan Apache `list_dags` Airflow `backfill` dan perintah ke perintah yang tidak didukung di. Mengakses Apache Airflow	17 Februari 2021
GitHub peluncuran	Dokumen panduan pengguna sekarang open source aktif. GitHub Pilih Edit halaman ini di GitHub halaman mana pun.	17 Februari 2021

Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Menambahkan pertanyaan untuk Step Functions v. Amazon MWAA use case ke. Memecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow • Menambahkan kebijakan akses CLI ke. Mengakses lingkungan Amazon MWAA • Mengklarifikasi dokumen bahwa opsi konfigurasi Apache Airflow yang didukung dapat ditentukan di. Menggunakan opsi konfigurasi Apache Airflow di Amazon MWAA • Mengklarifikasi dokumen bahwa jika wadah Fargate di satu zona ketersediaan gagal, MWAA beralih ke wadah lain di zona ketersediaan yang berbeda di. Buat jaringan VPC	12 Februari 2021
Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Ditambahkan Mengkonfigurasi kelas lingkungan Amazon MWAA.	5 Februari 2021

Menghapus atau memindahkan topik	Menambahkan perubahan berikut: • Persyaratan yang dihapus untuk nama bucket Amazon S3 untuk memulai `airflow-` di. Memulai dengan Amazon Managed Workflows for Apache Airflow • Pindah Mengakses lingkungan Amazon MWAA dan Peran eksekusi Amazon MWAA keMengelola akses ke lingkungan Amazon MWAA.	4 Februari 2021
Amazon MWAA CloudFormation	Perbarui parameter untuk membuat lingkungan di Amazon MWAA CloudFormation. • Hapus SubnetList. • Hapus TagList. • Tambahkan NetworkConfiguration. • Tambahkan TagMap. • Tambahkan contoh permintaan lingkungan buat.	4 Februari 2021

Topik baru dan kasus penggunaan	Menambahkan perubahan berikut: • Ditambahkan contoh konfigurasi email keMenggunakan opsi konfigurasi Apache Airflow di Amazon MWAA. • Menambahkan topik PostgresHook pemecahan masalah ke. Memecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow • Menambahkan topik AWS Secrets Manager pemecahan masalah ke. Memecahkan Masalah Alur Kerja Terkelola Amazon untuk Apache Airflow • Menambahkan kasus penggunaan kinerja tinggi keMengkonfigurasi penskalaan otomatis pekerja Amazon MWAA.	29 Januari 2021
Peluncuran Amazon MWAA	Peluncuran ketersediaan umum Alur Kerja Terkelola Amazon untuk Apache Airflow. • Dokumentasi panduan pengguna • CloudFormation dokumentasi	24 November 2020

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.