



Seller Guide

AWS Marketplace



AWS Marketplace: Seller Guide

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is AWS Marketplace?	1
Using AWS Marketplace as a seller	1
Contract structure for AWS Marketplace	3
Pricing of products in AWS Marketplace	4
Getting started as a seller	6
Seller requirements for publishing free software products	6
Additional seller requirements for paid products	7
Eligible jurisdictions for paid products	7
AWS Marketplace Management Portal	8
Seller registration process	9
Creating your public profile	10
Providing tax information	11
Providing US bank account information	12
Completing the Know Your Customer process	13
Completing bank account verification process	15
(Optional) Add secondary users for the Know Your Customer procedure	16
Disbursement and buyer billing	17
Already a seller?	19
Complaints handling policy – Amazon Payments Europe	19
Listings fees	21
Public offer listing fees	21
Private offer listing fees	21
Channel partner private offer (CPPO) listing fees	22
Professional services listing fees	22
Seller toolkit	22
AWS Marketplace Commerce Analytics Service	22
AWS Marketplace Field Demonstration Program	40
More resources in AWS Marketplace Management Portal	41
Preparing your product	42
Product delivery	42
Product pricing	46
Pricing models	47
Changing pricing models	51
Changing prices	51

Private offers	52
Product refunds	52
Regions and countries	55
AWS Regions	55
Countries	56
Standardized contracts	56
Standard Contract for AWS Marketplace	57
Reseller Contract for AWS Marketplace	58
Categories and metadata	59
Naming and describing your product	59
Choosing categories and keywords	60
AMI and container product usage instructions	61
.....	59
CloudFormation delivery	63
Monitoring and assessing application functions	63
Rotating programmatic system credentials and cryptographic keys	64
Search engine optimization for products	64
Search engine optimization	64
AWS Marketplace search	65
Preparing your private offer	69
How private offers work	69
Private offer considerations	70
Private offer experience for buyers	71
Reporting for private offers	72
Supported product types	72
Private offers for AMI products	72
Private offers for container products	73
Private offers for professional services products	74
Private offers for SaaS products	74
Private offers for ML products	74
Creating and managing private offers	75
Starting a new private offer	75
Understanding offer statuses	76
Drafting and publishing the private offer	77
Sending a private offer to a buyer	78
Saving your private offer progress	79

Updating the expiration of a private offer	79
Cancelling a private offer	80
Channel partner private offers	80
Additional information	82
Creating a resell opportunity as an ISV	82
Installment plans	85
Creating a payment schedule	85
Reporting for installment plans	86
Private offer upgrades, renewals, and amendments	87
Managing agreements for private offers	87
Supported product types for private offer amendments	87
Submission process for upgrades and renewals	88
Reporting for upgrades, renewals, and amendments	89
Future dated agreements	89
Creating future dated agreements	91
Using an installment plan with future dated agreements	91
Receiving notifications for future dated agreements	91
Using future dated agreements with reselling for Channel Partner private offers	92
AMI-based products	94
AMI-based product delivery methods	94
Understanding AMI-based products	95
Product lifecycle	95
AMI product codes	98
Change requests	99
Product Load Forms	100
Single-AMI products	101
Prerequisites	102
Understand the self-service experience	102
Create your single-AMI product	103
Create a change request	108
Get the status of a change request	111
Update product information	112
Update the allowlist (preview accounts)	113
Update product visibility	113
Add an AWS Region	114
Restrict an AWS Region	114

Update support for future AWS Regions	116
Add an instance	116
Restrict an instance	117
Update version information	118
Add a new version	119
Restrict a version	121
Update pricing	123
Update availability by country	124
Update your EULA	125
Update the refund policy	125
Give AWS Marketplace access to your AMI	126
Remove a product from AWS Marketplace	127
Troubleshooting common errors when submitting change requests	128
AMI-based delivery using CloudFormation	129
Building your product listing	130
Preparing your CloudFormation template	131
Getting the cost estimate for your template infrastructure	133
Architectural diagram	133
Meeting the submission requirements	134
Submitting your product request	136
Adding serverless application components	136
Private images	145
Best practices for building AMIs	145
Securing resell rights	145
Building an AMI	145
Preparing and securing your AMI for AWS Marketplace	146
Scanning your AMI for publishing requirements	147
Verifying your software is running on your AWS Marketplace AMI	148
AMI product pricing	149
AMI pricing models	150
AWS charges and software charges	154
Custom metering pricing for AMI products	154
Contract pricing for AMI products	165
AMI product billing, metering, and licensing integrations	168
Custom metering for AMI products with AWS Marketplace Metering Service	168
Contract pricing for AMI products with AWS License Manager	174

Amazon SNS notifications for AMI products	187
Amazon SNS topic: aws-mp-subscription-notification	188
Subscribing an Amazon SQS queue to the Amazon SNS topic	188
AMI product checklist	189
AMI-based product requirements	191
Security policies	191
Access policies	192
Customer information policies	193
Product usage policies	193
Architecture policies	195
AMI product usage instructions	195
Container-based products	196
Getting help	197
Getting started with container products	197
Prerequisites	198
Creating a container product	198
Product lifecycle	199
Updating a container product (legacy)	200
Updating product visibility	200
Updating the allowlist of AWS account IDs	201
Adding a pricing dimension	201
Updating dimension information	202
Updating pricing terms	202
Updating availability by country	203
Updating end-user license agreement	204
Updating the refund policy of a product	204
Creating the product ID and product code for your container product	205
Creating an initial listing	205
Creating or updating pricing details for container products (legacy)	206
Integrating AWS Marketplace Metering Service for your container product	208
Integrating AWS License Manager for your container product	208
Adding a new version of your product	208
Testing and releasing your product	216
Updating version information	218
Restricting a version of your Amazon EKS add-on	220
Creating or updating product information for your container product	220

Publishing container products (legacy)	221
Container product scans for security issues	222
Container-based product requirements	222
Security requirements	222
Access requirements	223
Customer information requirements	223
Product usage requirements	224
Architecture requirements	225
Container product usage instructions	225
Requirements for Amazon EKS add-on products	226
Container products pricing	242
Container pricing models	243
Contract pricing for container products	247
Container product billing, metering, and licensing integrations	251
Hourly and custom metering with AWS Marketplace Metering Service	251
Contract pricing with AWS License Manager	253
Hourly metering with AWS Marketplace Metering Service	255
Custom metering for container products	266
Contract pricing for Container products with AWS License Manager	279
Amazon SNS notifications for container products	312
Amazon SNS topic: aws-mp-subscription-notification	312
Subscribing an Amazon SQS queue to the Amazon SNS topic	313
Machine learning products	314
Getting started with machine learning products	314
SageMaker model package	314
SageMaker algorithm	315
Deploying an inference model	315
Security and intellectual property	316
Protecting intellectual property	316
No network access	316
Security of customer data	316
Machine learning product pricing	317
Infrastructure pricing	317
Software pricing	317
Prepare your product in SageMaker	320
Packaging your code into images	320

Uploading your images	343
Creating your Amazon SageMaker resource	346
Publishing your product in AWS Marketplace	352
Overview of publishing process	353
Permissions required	353
Creating your product listing	353
Testing your product	361
Signing off for publishing	362
Updating your product	362
Requirements and best practices for creating machine learning products	364
Required assets	365
General best practices for ML products	365
Requirements for usage information	366
Requirements for inputs and outputs	366
Requirements for Jupyter notebook	367
Summary of requirements and recommendations for ML product listings	368
Service restrictions and quotas	372
Network isolation	372
Image size	372
Storage size	372
Instance size	372
Payload size for inference	373
Processing time for inference	373
Service quotas	373
Asynchronous inference	373
Serverless inference	373
Managed spot training	374
Docker images and AWS accounts	374
Publishing model packages from built-in algorithms or AWS Marketplace	374
Supported AWS Regions for publishing	374
Troubleshooting	375
Reporting	377
Daily business report	377
Monthly revenue report	377
Disbursement report	377
Other reports and analysis	377

SaaS-based products	378
Getting started with SaaS products	378
Prerequisites	378
SaaS product lifecycle	379
Creating a SaaS product	380
Create an initial SaaS product page	385
SaaS product settings	386
Integrate your SaaS subscription product	397
Integrate your SaaS contract product	400
Integrate your SaaS contract with pay-as-you-go product	404
Deploy a serverless SaaS integration solution	408
Plan your SaaS product	409
Plan your pricing	410
Plan your billing integration	410
Plan your Amazon SNS integration	410
Plan how customers will access your product	410
SaaS product guidelines	411
Product setup guidelines	412
Customer information requirements	412
Product usage guidelines	412
Architecture guidelines	413
SaaS product pricing	414
Pricing for SaaS subscriptions	415
Pricing for SaaS contracts	417
SaaS free trials	422
Creating a SaaS free trial offer	422
Cancelling a SaaS free trial offer	423
SaaS customer onboarding	424
Configuring your SaaS product to accept new buyers	424
Amazon SNS notifications for SaaS products	427
Amazon SNS topic: aws-mp-entitlement-notification	428
Amazon SNS topic: aws-mp-subscription-notification	428
Subscribing an SQS queue to the SNS topic	430
Accessing the AWS Marketplace Metering and Entitlement Service APIs	430
Metering for usage	431
Checking entitlements	437

SaaS product integration checklist	438
Reporting	442
SaaS code examples	443
ResolveCustomer code example	443
GetEntitlement code example	444
BatchMeterUsage code example	445
BatchMeterUsage with usage allocation tagging code example (Optional)	447
Using AWS PrivateLink with AWS Marketplace	449
Introduction	449
Configuring your product	451
Submitting your product to AWS Marketplace	451
Buyer access to VPC endpoints	452
Appendix: Checklists	453
Professional services products	455
Getting help	456
Getting started with professional services products	456
Prerequisites	456
Creating a professional services product	457
Creating private offers	458
Editing product information	460
Editing product pricing	461
Editing product visibility	462
Removing a professional services product	462
Product details	463
Product descriptions	464
Additional resources	465
Support information	465
Pricing dimensions	466
Product visibility	466
Product requirements	466
Product setup guidelines	467
Customer information requirements	467
Product usage guidelines	467
Architecture guidelines	468
Professional services product pricing	468
Data products	470

Submitting your product	471
Using the Products tab	472
Company and product logo requirements	473
Requirements for submitting paid repackaged software	474
Requirements for products with a hardware component	475
AWS CloudFormation-launched product (free or paid) or usage-based paid AMI product	475
Submitting your product	475
Updating your product	476
Product changes and updates	477
Timing and expectations	477
Submitting AMIs to AWS Marketplace	478
AMI self-service scanning	478
AMI cloning and product code assignment	479
Final checklist	479
Marketing your product	481
180-day GTM Academy	481
Announcing your product's availability	481
AWS Marketplace messaging	482
Reviews on AWS Marketplace	483
Linking to AWS Marketplace	484
Using the AWS Marketplace logo	484
Linking directly to your product on AWS Marketplace	484
Press releases	485
AWS Marketplace trademark usage guidelines	486
Notifications	488
Email notifications	488
Event types	489
Manage notifications	489
Amazon EventBridge notifications	491
AWS Marketplace Catalog API Amazon EventBridge events	491
Amazon Simple Notification Service notifications	499
Seller reports, data feeds, and dashboards	500
Seller delivery data feeds service	500
Storage and structure of data feeds	501
Accessing data feeds	503
Unsubscribing from data feeds	506

Using data feeds	506
Data feed tables overview	507
Data feed query examples	515
Data feeds	531
Seller reports	569
Accessing reports	569
Daily business report	570
Daily customer subscriber report	581
Disbursement report	584
Monthly billed revenue report	593
Sales compensation report	603
US sales and use tax report	606
Supplementary reports	610
Agreement details report	611
Seller dashboards	612
Accessing dashboards	569
Dashboards for finance operations	615
Dashboards for sales operations	644
AWS Marketplace Vendor Insights	662
Understanding AWS Marketplace Vendor Insights	663
Setting up as a seller	664
Create a security profile	664
Upload a certification	665
Upload a self-assessment	667
Enable AWS Audit Manager automated assessments	668
Viewing your profile	674
View your security profile as a seller	674
Managing snapshots	675
Create a snapshot	676
View a snapshot	677
Export a snapshot	677
View latest released snapshot	678
Postpone a snapshot release	678
Change preferences for the snapshot list	678
Controlling access	679
Permissions for AWS Marketplace Vendor Insights sellers	680

CreateDataSource	680
DeleteDataSource	681
GetDataSource	681
UpdateDataSource	681
ListDataSources	681
CreateSecurityProfile	681
ListSecurityProfiles	682
GetSecurityProfile	682
AssociateDataSource	682
DisassociateDataSource	682
UpdateSecurityProfile	683
ActivateSecurityProfile	683
DeactivateSecurityProfile	683
UpdateSecurityProfileSnapshotCreationConfiguration	683
UpdateSecurityProfileSnapshotReleaseConfiguration	684
ListSecurityProfileSnapshots	684
GetSecurityProfileSnapshot	684
TagResource	684
UntagResource	685
ListTagsForResource	685
Additional resources	685
Security	191
IAM for AWS Marketplace	687
Creating users	688
Creating or using groups	690
Signing in as a user	691
Policies and permissions for AWS Marketplace sellers	692
Policies	693
Permissions	693
AWS managed policies	700
AWSMarketplaceAmiIngestion	701
AWSMarketplaceFullAccess	702
AWSMarketplaceGetEntitlements	705
AWSMarketplaceMeteringFullAccess	705
AWSMarketplaceMeteringRegisterUsage	706
AWSMarketplaceSellerFullAccess	706

AWSMarketplaceSellerProductsFullAccess	710
AWSMarketplaceSellerProductsReadOnly	711
AWSVendorInsightsVendorFullAccess	712
AWSVendorInsightsVendorReadOnly	714
Policy updates	715
AWS Marketplace Commerce Analytics Service account permissions	719
Amazon SQS permissions	720
AWS Marketplace metering and entitlement API permissions	721
IAM policy for SaaS products	721
IAM policy for AMI products	722
IAM policy for container products	722
Using service-linked roles	723
Roles for Resale Authorization	723
Logging AWS Marketplace API calls with AWS CloudTrail	728
AWS Marketplace Metering API log file entry examples	728
Document history	735
AWS Glossary	755

What is AWS Marketplace?

AWS Marketplace is a curated digital catalog that customers can use to find, buy, deploy, and manage third-party software, data, and services to build solutions and run their businesses. AWS Marketplace includes thousands of software listings from popular categories such as security, business applications, machine learning, and data products across specific industries, such as healthcare, financial services, and telecommunications. Customers can quickly launch preconfigured software, and choose software solutions in Amazon Machine Images (AMIs), software as a service (SaaS), and other formats. Professional services are also available to help customers configure, deploy, and manage third-party software. For a complete list of delivery methods, see [Product delivery](#).

You can use AWS Marketplace as a buyer (subscriber), seller (provider), or both. Anyone with an AWS account can use AWS Marketplace as a buyer, and can register to become a seller. A seller can be an independent software vendor (ISV), channel partner, managed services provider (MSP), or individual who has something to offer that works with AWS products and services.

Note

Data product providers must meet the AWS Data Exchange eligibility requirements. For more information, see [Providing data products on AWS Data Exchange](#) in the *AWS Data Exchange User Guide*.

Eligible partners can programmatically list AWS Marketplace products outside of AWS Marketplace. For more information about becoming an eligible partner, contact your AWS Marketplace business development partner.

The following video explains more about selling in AWS Marketplace.

[Introduction to AWS Marketplace](#)

Using AWS Marketplace as a seller

The process for selling a software product in AWS Marketplace involves the following seven steps.

Seller process

Step	Action	Description
1	Register	As a seller, you start by registering for the AWS Marketplace Management Portal. We recommend that you implement a new dedicated AWS account that can be easily linked with an existing AWS organization. Verify that the AWS Partner's tax information meets the jurisdictional eligibility criteria. For AWS Partners without an existing U.S. bank account, you can create one at no additional cost by using Hyperwallet .
2	Decide product type	Decide on the type of product that you want to sell. For more information about creating the product types in AWS Marketplace, see the following: <ul style="list-style-type: none"> • AMI-based products • Container-based products • Machine learning products • SaaS-based products • Professional services products • Data products (For more information about data products, see What is AWS Data Exchange? in the <i>AWS Data Exchange User Guide</i>.)
3	Prepare product	Configure your package, set a pricing scheme, determine the relevant categories in which to list your product, and add keywords so your product appears in relevant searches. To simplify the procurement process, you can use standardized license terms for both public product listings and private offers.
4	Submit product	Use the product submission process to make your products available in AWS Marketplace. Products can be simple, for example, a single Amazon Machine Image (AMI) that has one price structure. Or, products can be complicated, with multiple AMIs, AWS CloudFormation templates, and complex pricing options and payment schedules.

Step	Action	Description
5	Market product	Contribute to the success of your product by driving awareness of AWS Marketplace and by driving traffic directly to your product pages in AWS Marketplace.
6	View reports and data feeds	After you're registered as a seller, use the AWS Marketplace Management Portal to access usage reports for your products. AWS Marketplace provides tools for collecting and analyzing information about your product sales.
7	Manage products	Use the AWS Marketplace Management Portal to manage your account and product pages.

As a seller, go to the [AWS Marketplace Management Portal](#) to register. If you're charging for use of your product, you must also provide tax and banking information as part of your registration. When you register, you create a profile for your company or for yourself that is discoverable in AWS Marketplace. You also use the [AWS Marketplace Management Portal](#) to create and manage product pages for your products.

Contract structure for AWS Marketplace

Usage of the software, services, and data products sold in AWS Marketplace is governed by agreements between buyers and sellers. AWS is not a party to these agreements.

As the seller, your agreements include the following:

- Your end user license agreement (EULA) with the buyer, which is located on the product listing page for public software listings in AWS Marketplace. Many sellers use the [Standard Contract for AWS Marketplace \(SCMP\)](#) as their default EULA. You can also use the SCMP as the basis for negotiations in private offers and use the amendment template to modify the SCMP. Private offers can also include custom contract terms negotiated between the parties.

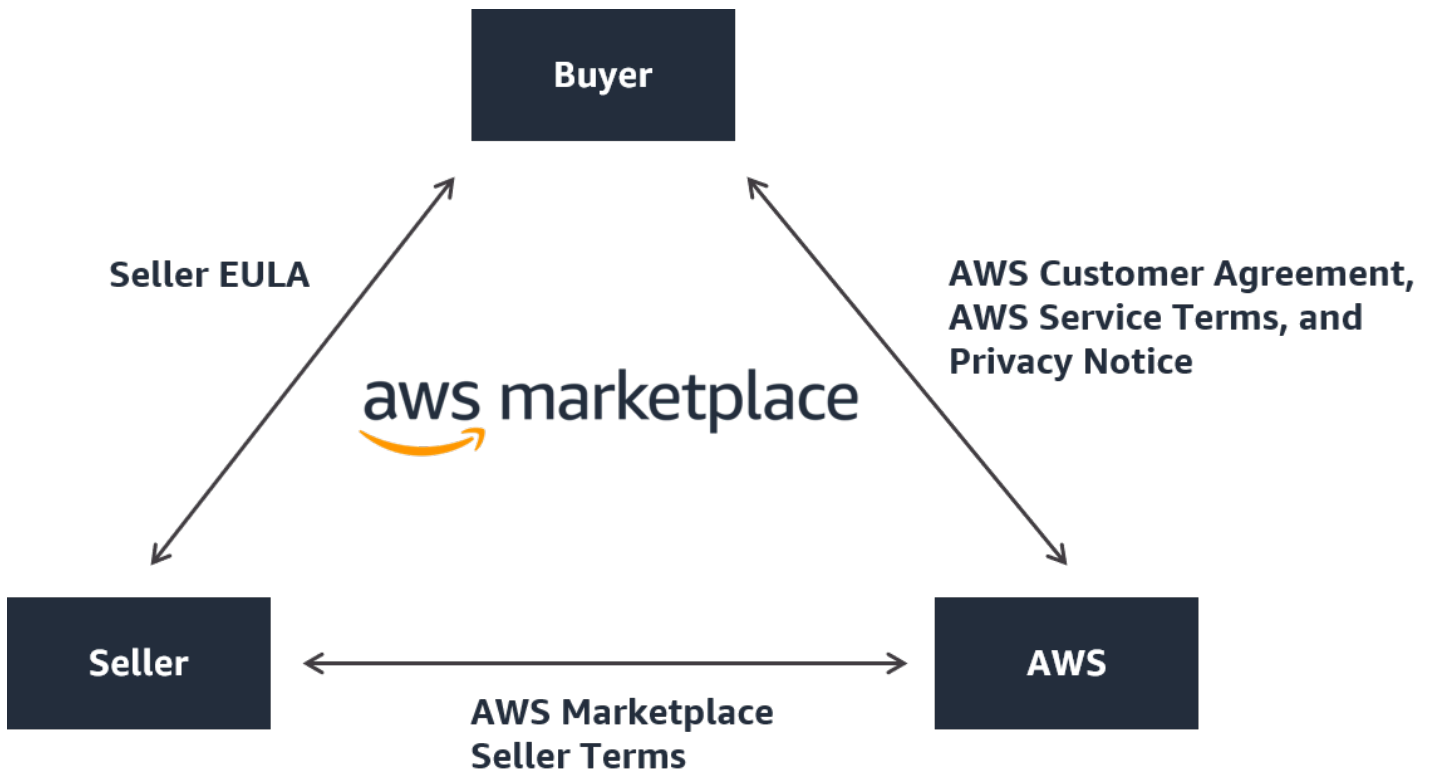
Note

For information on when a EULA update will occur, based on the offer type and pricing model, see [EULA updates](#) in the *AWS Marketplace Buyer Guide*.

- The [AWS Marketplace Seller Terms](#), which govern your activity in AWS Marketplace.

A buyer's use of AWS Marketplace is governed by the [AWS Service Terms](#), the [AWS Customer Agreement](#), and the [Privacy Notice](#).

The following graphic shows the contract structure for AWS Marketplace.



Pricing of products in AWS Marketplace

In AWS Marketplace, products can be free to use or can have associated charges. The charge becomes part of the buyer's AWS bill, and after the buyer pays, AWS pays the seller. Products can take many forms. For example, a product can be offered as an Amazon Machine Image (AMI) that is instantiated using a buyer's AWS account. Products can also be configured to use CloudFormation templates for delivery to the buyer. Products can also be SaaS offerings from an ISV, web access control lists (web ACL), sets of rules, or conditions for AWS WAF. Products can also be professional services from an ISV, channel partners, or MSP.

Flexible pricing options include free trial, hourly, monthly, annual, multi-year, and Bring Your Own License model (BYOL), and being billed from one source. AWS handles billing and payments, and charges appear on customers' AWS bill.

Software products can be purchased at the listed price using the ISV's standard end user license agreement (EULA). In addition, software products can be offered with custom pricing and EULA through private offers. Products can also be purchased under a contract with specified time or usage boundaries. After subscribing to a product, the buyer can use AWS Service Catalog to copy the product and manage how the product is accessed and used in the buyer's organization. For more information about the buyer's experience, see <https://docs.aws.amazon.com/marketplace/latest/buyerguide/service-catalog.html>. For more information about pricing, see [the section called "Product pricing"](#).

Getting started as a seller

If you want to sell your software in AWS Marketplace, review the requirements and then follow the steps to register as a seller. There are different registration requirements based on where you reside and what type of products you're selling. To register as a seller in AWS Marketplace, you can use an existing AWS account or create a new account. All AWS Marketplace interactions are tied to the account that you choose.

Notes

- Registering as an AWS Marketplace seller is a prerequisite to listing data products on AWS Data Exchange and making them available on AWS Marketplace. For more information about these requirements, see [Providing Data Products on AWS Data Exchange](#) in the *AWS Data Exchange User Guide*.
- For information about the permissions that AWS Marketplace sellers need, see [Policies and permissions for AWS Marketplace sellers](#).
- For more information about product listing fees, registered sellers can view the [AWS Marketplace Seller Terms](#) in the AWS Marketplace Management Portal.

Seller requirements for publishing free software products

Regardless of whether you charge for your product when you offer it in AWS Marketplace, you're selling that product. The cost to the customer is \$0.00, but you and the customer agree to a mutual contract for use of the product. If you offer only free products, you don't have to provide banking information to AWS Marketplace. To create and offer free products in AWS Marketplace, you must:

- Sell publicly available, full-feature production-ready software.
- Have a defined customer support process and support organization.
- Provide a means to keep software regularly updated and free of vulnerabilities.
- Follow best practices and guidelines when marketing your product in AWS Marketplace.
- Be an AWS customer in good standing and meet the requirements in the terms and conditions for AWS Marketplace sellers.

Additional seller requirements for paid products

If you charge for your products or offer Bring Your Own License model (BYOL) products, you must also meet the following requirements and provide this additional information:

- You must be a permanent resident or citizen in an [eligible jurisdiction](#), or a business entity organized or incorporated in one of those areas.
- You must provide tax and bank account information. For US-based entities, a W-9 form and a banking account from a US-based bank are required.
- Non-US sellers are required to provide a (i) W-8 form, value-added tax (VAT) or goods and services tax (GST) registration number, and (ii) US bank information. If you don't have a US bank account, you can register for a virtual US bank account from [Hyperwallet](#).
- To provide data products, you must also request on-boarding through the [Create case](#) wizard for AWS Support.
- To sell products to customers whose AWS accounts are based in countries and territories in Europe, the Middle East, and Africa (EMEA) (excluding Turkey and South Africa) through Amazon Web Services EMEA SARL, you must [complete the Know Your Customer process](#). In addition:
 - You receive up to two disbursements (for transactions through AWS Inc. and Amazon Web Services EMEA SARL).
 - You may be taxed on the listing fee for certain transactions, depending on location. For more information about taxes, see the [AWS Marketplace Sellers Tax](#) help page. If value-added tax (VAT) on your listing fee is assessed, AWS Marketplace will provide a tax-compliant invoice.
 - For more information about Amazon Web Services EMEA SARL, see *AWS EMEA Marketplace - Sellers* on the [Amazon Web Services Europe FAQs](#) website.

To sell into the AWS GovCloud (US) Region, sellers must have an [AWS GovCloud \(US\) account](#). For details on ITAR requirements, see the [AWS GovCloud \(US\) User Guide](#).

For questions about AWS Marketplace seller requirements or the registration process, contact the [AWS Marketplace Seller Operations](#) team.

Eligible jurisdictions for paid products

To sell paid software in AWS Marketplace, you must be a permanent resident or citizen in one of the following countries or SARs, or a business entity organized or incorporated therein:

- Australia¹
- Bahrain^{1 2}
- European Union (EU) member state¹
- Hong Kong SAR
- Israel^{1 2}
- Japan^{1 2 3}
- New Zealand¹
- Norway^{1 2}
- Qatar
- Switzerland^{1 2}
- United Arab Emirates (UAE)^{1 2}
- United Kingdom (UK)¹
- United States (US)

¹ Sellers of paid products in these countries must provide VAT registration information in country of establishment.

² If you as a seller are located in the same country as the buyer, you may be responsible for tax invoicing, collections, and remittances. Please consult with your tax advisor.

³ Sellers based in Japan have an obligation to self-account for the Japanese Consumption Tax (JCT) on the listing fee charges. Sellers based in other jurisdictions may have similar obligations. Please consult with your tax advisor.

For more information about VAT, invoicing, and your tax obligations as a seller, see [AWS Marketplace Sellers](#) on [Amazon Web Service Tax Help](#).

If a business entity is not incorporated in one of the countries or SARs listed above, see [Resource for companies outside of AWS Marketplace jurisdictions](#).

AWS Marketplace Management Portal

The [AWS Marketplace Management Portal](#) is the tool that you use to register as an AWS Marketplace seller. Then, you can use the portal to manage the products that you sell in AWS Marketplace. You can complete the following tasks on the portal:

- Register as an AWS Marketplace seller.
- Use the **Products** page to submit new software products and update existing software products.
- Monitor the status of your requests.
- Upload files needed to create and manage your new software products.
- Manage your software products into incremental channel revenue by taking advantage of the go-to-market activities.
- Measure the results of your marketing efforts within hours of launch, including the usage and revenue driven by your campaigns.
- Enable customer service representatives to retrieve customer data in real time.
- Initiate an automatic Amazon Machine Image (AMI) scan to detect vulnerabilities.

Note

Data products are published and managed from the AWS Data Exchange console. AWS Data Exchange providers can use the AWS Marketplace Management Portal to register as a seller, request AWS Data Exchange on-boarding, access seller reports, and submit refund requests.

All registered sellers can access the AWS Marketplace Management Portal using their AWS credentials for the account that they used to create their products. The account that you use is defined as the seller of record when a customer subscribes to your product. If you need help determining the specific account that is the seller of record for your products, contact the [AWS Marketplace Seller Operations](#) team.

AWS Marketplace strongly recommends using AWS Identity and Access Management (IAM) roles to sign in to the AWS Marketplace Management Portal rather than using your root account credentials. For more information, see [AWS Marketplace security](#).

Seller registration process

By registering as a seller for AWS Marketplace, you can sell your products and services to other AWS Marketplace customers.

Registering as a seller requires the following steps:

1. **Create your public profile** – You provide the information that is displayed in AWS Marketplace to buyers that tells them about your company, such as your company name and logo. After you complete this process, you can sell products for free. To sell paid products, you must complete steps two and three.
2. **Provide your tax information** – To appropriately assess, report, and (where applicable) withhold taxes on your paid sales, you must provide your tax and value added tax (VAT) information.
3. **Provide your banking information** – You provide your US bank information so that AWS Marketplace can pay you for your sales.

These steps are described in more detail in the following sections.

After you have completed registering your account as a seller, you can create products to sell to buyers through AWS Marketplace. For more information, see [Preparing your product](#).

You can use AWS Identity and Access Management (IAM) to configure your primary AWS account to allow multiple users with various permissions to access the AWS Marketplace Management Portal. For more information, see [the section called “IAM for AWS Marketplace”](#).

Creating your public profile

The first step to register is to select the AWS account to use as your primary AWS Marketplace account, and provide the information that is displayed to potential buyers in the AWS Marketplace console. This account will be the seller of record for your products in AWS Marketplace and will be used for reporting, disbursement, and communication from AWS Marketplace to you.

Once you use an AWS account to register as a seller and list a product on AWS Marketplace, you can't change the account associated with the product. We recommend that you use a new account to register as an AWS Marketplace seller. However, you can use an existing account if that account was created after September 27, 2017.

To create your public profile

1. From the [AWS Marketplace Management Portal](#) (AMMP), choose **Register now** and then sign in to your chosen seller AWS account.
2. Select **Add public profile** to provide your seller information.

After you have completed the public profile, you can publish and sell free products. To sell paid products, you must provide your tax and banking information.

Providing tax information

You must provide your tax, and value added tax (VAT) where applicable, information so that AWS Marketplace can accurately report and withhold taxes on your product sales.

To provide your tax information

1. Sign in to the [AWS Marketplace Management Portal](#), and choose **Settings**.
2. Select **Go to tax dashboard** in the **Payment Information** section.
3. Complete the U.S. tax interview.
 - To sell professional services on AWS Marketplace, you must complete the **Tax Questionnaire for DAC7**.
4. After you have completed the tax information, return to the **Settings** page, and then select **Complete VAT information**, if it's available. This selection redirects to the **Tax Settings** page on the AWS Billing console.

Note

The VAT information section is only available if you are in an AWS Region that supports VAT.

Accessing tax documents

You can access your tax documents, such as 1099 forms, from the AWS Marketplace Management Portal.

To access your tax documents

1. Sign in to the [AWS Marketplace Management Portal](#), and choose **Settings**.
2. Go to the **Payment Information** section.
3. Select the relevant **tax forms** (1099K or DAC7).
4. If your tax forms are available, you can download them on the **Tax dashboard** page.

Providing US bank account information

A US bank account is required for all sellers who want to sell paid products in AWS Marketplace. AWS Marketplace only disburses to US bank accounts.

Note

For a list of countries where you can offer paid products in AWS Marketplace, see [Eligible jurisdictions for paid products](#).

To provide US bank information

1. Sign in to the [AWS Marketplace Management Portal](#), and choose **Settings**.
2. Select **Complete banking information** in the **Payment Information** section.
3. Provide the required information about your US bank account.

Note

If you have not yet provided your tax information (and value added tax information, where applicable), you will not be able to provide your banking information.

If you don't already have a US bank account, you might be able to obtain one through Hyperwallet. Hyperwallet can provide you with a US account, which you can provide to AWS Marketplace for your AWS Marketplace disbursements.

Hyperwallet is an independent service provider that can enable you to transfer funds to another bank account in a supported currency. For a limited time, you will not be required to pay certain Hyperwallet service fees in connection with AWS Marketplace disbursements.

- By adding your Hyperwallet account details to your AWS Marketplace seller account, you agree and acknowledge that AWS Marketplace will share your name, email address, and account number with Hyperwallet to confirm your status as an AWS Marketplace seller.
- Additional fees may apply to your use of Hyperwallet services (including transfer fees and foreign exchange fees required to transfer funds into your local currency), as well as foreign exchange rates. The Hyperwallet service fee will be waived for a limited time, and only with

respect to AWS Marketplace disbursements of the proceeds from your Paid products into your Hyperwallet account. For more information, see the *Fees* section of the Hyperwallet site or contact Hyperwallet for more information and to review applicable fees. For more information about their services, see the [Hyperwallet support site](#).

To begin registration with Hyperwallet and obtain your US bank account information

1. Sign in to the [AWS Marketplace Management Portal](#), and choose **Settings**, then select **Complete banking information** in the **Payment Information** section.
2. If you don't have a Hyperwallet account, and need one for use in AWS Marketplace, choose **No** in response to **Do you have a US bank account?** and **Are you registered with Hyperwallet?** You will be provided with a personal identification number (PIN) and link to sign up for Hyperwallet.
3. After you have activated your Hyperwallet account, follow the steps described on the Hyperwallet registration portal to complete registration and receive your deposit account information.
4. When you have obtained an account from Hyperwallet, add your Hyperwallet account information to your AWS account by signing in to the [AWS Marketplace Management Portal](#). Then, choose **Settings**, then select **Complete banking information** in the **Payment Information** section.

Completing the Know Your Customer process

Know Your Customer (KYC) is a compliance requirement used by financial institutions and online businesses to verify the identity of their customers. This requirement is due to the revised Payment Services Directive (PSD 2) and European Union anti-money laundering Directives that govern financial institutions such as banks and other payment institutions.

AWS Marketplace transactions through Amazon Web Services EMEA SARL are processed through Amazon Payments Europe, S.C.A. (APE), a licensed electronic money institution in Luxembourg which requires verification of your identity in order to use the payment service.

For you as a seller to transact through Amazon Web Services EMEA SARL, you are required to complete the KYC process. This process involves providing additional information about your company, key points of contact, beneficial ownership, and supporting documentation.

To complete the KYC process

1. On the AWS Marketplace Management Portal, choose **Settings**.
2. In the **Account Summary** section, confirm that the **Country** that is shown is correct.

Note

Choose the **Info** link to see how to change your country.

3. Choose **Go to KYC information** or select the **Know your customer (KYC)** tab and then choose **Start KYC Compliance** and you will be re-directed to the KYC registration portal.

For more information about how information is used and shared by AWS Marketplace, see the [Amazon Payments Europe Privacy Notice](#).

4. Choose **Go to KYC overview**.
5. On the **Know Your Customer (KYC) Overview**, read through the list of required information and documentation and gather the required documentation (if you haven't yet done so). Then choose **Continue to KYC compliance**.
6. Enter the **Basic details** as directed. After you review the Amazon Payments Europe Terms & Conditions, choose **Agree and continue**.

When you continue to the next page or next step in the KYC process, that action indicates that you accept the Amazon Payments Europe Terms & Conditions.

If you have questions, refer to **Frequently Asked Questions (FAQ)** located on the right side of the console.

7. Enter the required **Business information** as directed, and then choose **Next**.

Note

Your information is saved every time you chose **Next** to go to the next step.

8. Enter the required **Point of contact information** as directed, and then choose **Next**.
9. Choose whether the **Beneficial owner** is the same as the point of contact, add beneficial owners (up to four) if necessary, confirm your additions, and then choose **Next**.

10. Choose whether the **Legal representative** is the same as the point of contact or beneficial owner. If the legal representative is a different entity, provide the required information, save your entry, and then choose **Next**.
11. For **Additional documents**, upload your business license, identity document, and letter of authorization (if applicable).
12. On **Review and Submit**, review and verify all of the information that you have entered.

You can select **Edit** to return to any previous section if necessary.

13. Choose **Submit for verification**.

The status of your KYC compliance will be reviewed (typically within 24 hours). You will be notified through an email message after the review is complete. The entire KYC process typically takes approximately 2 weeks.

You can return to the **Settings** tab to view the status of your KYC compliance on the **Account Summary** card. For more information about your KYC status, choose the **Know your customer (KYC)** tab under the **Account Summary** card. It will display **Under review** until the review has been completed.

After your KYC is verified, you must provide a bank statement on the **Payment information** tab before you can receive disbursements through APE.

Completing bank account verification process

To receive disbursements from Amazon Payments Europe (APE), you must provide additional information to verify your disbursement bank account that is listed in the **Payment Information** tab in the AWS Marketplace Management Portal.

Providing additional bank information

To provide additional bank information

1. Sign in to the AWS Marketplace Management Portal, and then choose **Settings**.
2. Select **Update banking information** in the **Payment Information** section.
3. Select the appropriate disbursement account.

The **Verification status** displays **Not Verified**.

4. Choose **Verify**.

5. You will be re-directed to the **Bank Account Verification** registration portal where you can upload and submit your bank statement.

If you use the Hyperwallet virtual bank account solution, see [the section called “Downloading your bank statement from Hyperwallet”](#).

6. In the portal, choose **Upload bank document** and then choose **Submit**.

Downloading your bank statement from Hyperwallet

For sellers who use the [Hyperwallet virtual bank account solution](#), you can download the Hyperwallet bank statement by using the following procedure. Then, you can upload the bank document as directed in [Completing bank account verification process](#).

To download your bank statement from Hyperwallet

1. Sign into your [Hyperwallet account](#).
2. Go to the [Deposit Account Information](#) page.
3. Download your **Bank Account Validation Statement**.

(Optional) Add secondary users for the Know Your Customer procedure

Note

Users are required to enable multi-factor authentication (MFA) to update disbursement information. For more information about MFA, see [Multi-Factor Authentication \(MFA\) for IAM](#).

Secondary users are individuals who can amend KYC information, control the flow of funds or refunds, and change financial information such as bank account details.

Only secondary users that are KYC verified can make the aforementioned updates. These secondary users are subject to the same ongoing screening controls as the root account owner.

To become KYC verified, secondary users must complete the procedure in [Completing the Know Your Customer process](#).

To add secondary users for the Know Your Customer procedure

1. Ask the user to sign in to the AWS Marketplace Management Portal.
2. Navigate to the **Settings** tab.
3. Choose the **Know Your Customer (KYC)** tab and see the section for **Secondary user information**.
4. Choose **Complete secondary user information**.

You are re-directed to the **Secondary User** registration portal.

5. In the **Secondary User** registration portal, complete the required fields, and then choose **Next**.
6. On the **Review and Submit** page, upload a copy of the identity document (**Upload Passport**) and proof of address (**Upload Document**).
7. Choose **Submit for Verification**.

The status of your KYC compliance will be reviewed (typically within 24 hours). You will be notified through an email message after the review is complete. The entire KYC process typically takes approximately 2 weeks.

Disbursement and buyer billing

AWS Marketplace sellers, including independent software vendors (ISVs) and Channel Partners, can set disbursement preferences to receive their outstanding balances. Sellers select daily or monthly disbursement options and can choose which day of the month to receive disbursements.

To set your disbursement preferences

1. Sign in to the AWS Marketplace Management Portal, and choose **Settings**.
2. Select the **Payment information** tab and choose **Update banking information**.
3. In the **Disbursement Preference** section, view your current disbursement option. To change to a daily disbursement schedule, select **Daily** and then choose **Submit**. You'll see a percentage bar that displays the progress of your updated banking information until completion.
4. To change from daily to monthly disbursement, select **Monthly** and choose a number between 1-28 for the day of the month you want your disbursement to process. Choose **Submit**.

Note

Sellers should be onboarded to the fine-grained IAM permissions to access the disbursement preference options. To onboard to the fine-grained IAM permissions, see [the section called “Policies and permissions for AWS Marketplace sellers”](#).

AWS acts as the billing mechanism on your behalf. The two most common payment options available to buyers are *credit card* and *invoicing*.

The following is information about the billing for AWS Marketplace subscriptions:

- Purchases with upfront payments are billed immediately upon subscription.
- Billing schedules for private offers are agreed upon between the buyer and seller.
- Invoice payment terms (including bill due date) are agreed upon between the buyer and AWS. The terms are *not* disclosed to vendors.
- Private offers using the flexible payment scheduler are required to be on *invoicing* as the payment option.
- You can validate the invoicing using the [Monthly billed revenue report](#). This report summarizes invoicing by AWS on your behalf. This report contains a Transaction Reference key to match and provide visibility to the invoice creation date and invoice due date.

The following is information about how you as the seller get your disbursement:

- A valid [payment method](#), a [registered US bank account](#), and a submitted W9 form are required for disbursement.
- Sellers of paid products are required to provide a W-8, value added tax (VAT) or good and services tax (GST) registration number, and a US bank account. [Hyperwallet](#) can provide you with a US bank account, which you can provide to AWS Marketplace for your AWS Marketplace disbursements.
- AWS disburses payments in the following ways:
 - **Daily** – Daily disbursements occur when they become available. Sellers must have a positive balance to receive disbursements.
 - **Monthly** – Sellers choose a day of the month (1–28) to receive disbursements. The disbursement date is the same each month. The [Disbursement report](#) shows your disbursement date.

- AWS disburses payment by using Automated Clearing House (ACH) transfer after the buyer pays an invoice.
- Funds are disbursed only after they are collected from the customer.
- Payments take approximately 1–2 business days to arrive in the seller's bank following the disbursement date. The exact timing is subject to the bank and the time zone.
- The disbursement report is updated in the AWS Marketplace Management Portal 3–5 days after the disbursement.
- Details about disbursed funds and uncollected funds are available in the disbursement report, including any open account receivables.

Already a seller?

Manage your products into incremental channel revenue by taking advantage of the go-to-market activities made available in the [AWS Marketplace Management Portal](#). Activities include the following:

- Measure the results of your marketing efforts within hours, including the usage and revenue driven by your campaigns.
- Enable customer service representatives to retrieve customer data in real time.
- Upload files needed to create and manage your products, and monitor progress as we process them.

Complaints handling policy – Amazon Payments Europe

If you have any issues with the services provided by Amazon Payments Europe (APE), let us know. Your feedback helps us create a better experience for you and all of our buyers and sellers.

Note

Only complaints specific to AWS Marketplace will be addressed through the following procedure. Services provided by Amazon Payments Europe S.C.A. include, among others, processing of payment transactions, verifying the errors that may appear in the fee charges, and disbursements of funds.

Submitting a complaint

If you have an AWS Marketplace account with Amazon Payments Europe S.C.A., your complaint will be handled by Amazon Payments Europe S.C.A.

To submit a complaint

1. Sign in to your [AWS Marketplace](#) Seller account.
2. Go to **Contact Us**.
3. Select **Commercial Marketplace, Seller Account, Registration**.
4. Provide details about your complaint and choose **Submit**.

Amazon Payments Europe Complaint resolution time frames

Amazon Payments Europe S.C.A. (APE) will respond with an update to your complaint within 15 business days following the day on which it received your complaint. In exceptional circumstances beyond the control of APE, the resolution of the complaint may be extended up to 35 business days, following the day on which APE first received that complaint.

Complaint escalation

If you aren't satisfied with our response, you may choose to escalate your complaint by contacting the following:

- Amazon Payments Europe senior management

Submit your complaints by sending an email message to senior management at ape-management@amazon.lu. We will consider your comments carefully and respond within 15 business days following the day on which senior management received your complaint. In exceptional circumstances beyond the control of Amazon Payment Europe, the resolution of the complaint may be extended up to 35 business days, following the day on which senior management first received the complaint.

- Commission de Surveillance du Secteur Financier (CSSF)


The CSSF is the authority responsible for the prudential supervision of companies in the financial sector in Luxembourg. You can contact the CSSF at 110 Route d'Arlon L-2991 Luxembourg or use the **Contact page** at: <https://www.cssf.lu/contacts/>. To obtain further information regarding the CSSF and how to contact them, see [Customer complaints](#) on the CSSF website.

- Online Dispute Resolution

If you opened your account online in the EU, you may also have the option to refer your complaint to the CSSF by using the Online Dispute Resolution platform. This option is available because Amazon Payments Europe S.C.A. provides financial services and the CSSF is the authority responsible for its licence. For more information, see the [Online Dispute Resolution](#) platform on the European Commission website.

Listing fees

AWS Marketplace offers the following listing fees for products.

 **Note**

These listing fees are effective as of January 5, 2024 at midnight UTC.

Public offer listing fees

Listing fees for software and data public offers are determined by the deployment method:

- Software-as-a-service (SaaS) – 3%
- Server (Amazon Machine Image (AMI), container, and machine learning) – 20%
- AWS Data Exchange – 3%

Private offer listing fees

Listing fees for private offers are determined by the total contract value and whether the private offer is renewed from a previous private offer or a previous agreement outside of AWS Marketplace:

- Less than \$1M – 3%
- Between \$1M and less than \$10M – 2%
- Equal to or greater than \$10M – 1.5%
- All renewals – 1.5%

Channel partner private offer (CPPO) listing fees

CPPO products have a .5% uplift on the listing fee, regardless of the offer type or deployment method. For example, if the product is a SaaS private offer with a total contract value of less than \$1M, the listing fee would be 3.5%.

Professional services listing fees

All professional service offerings have a 2.5% listing fee for private offers.

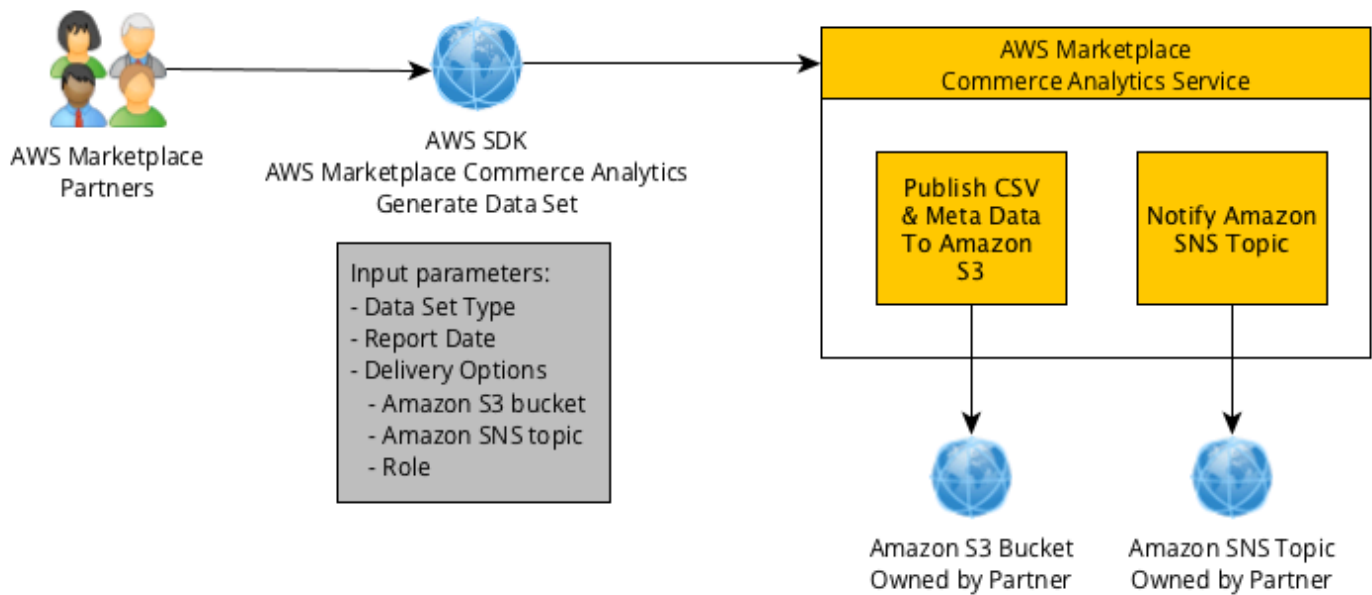
Seller toolkit

The [AWS Marketplace Management Portal](#) is your primary tool for selling products on AWS Marketplace. The following additional tools can give you more insight into your customer base and help you better understand your sales.

- [AWS Marketplace Commerce Analytics Service](#)
- [AWS Marketplace Field Demonstration Program](#)
- [Seller reports, data feeds, and dashboards](#)
- [More resources in AWS Marketplace Management Portal](#)

AWS Marketplace Commerce Analytics Service

The AWS Marketplace Commerce Analytics Service lets you programmatically access product and customer data through AWS Marketplace. After you enroll in the service, you can access your usage, subscription, and billing reports through the AWS SDK.



The data you request using the SDK tools is delivered to your AWS account as datasets. Most of the datasets correspond to the same data as the text-based reports available on the [AWS Marketplace Management Portal](#). You can request datasets for a specific date, and the data is delivered to the provided Amazon S3 bucket. Notification of data delivery is provided by the Amazon Simple Notification Service (Amazon SNS).

Terms and conditions

These AWS Marketplace Commerce Analytics Service Terms and Conditions (these "**CAS Terms**") contain the terms and conditions specific to your use of and access to the AWS Marketplace Commerce Analytics Service ("**CA Service**") and are effective as of the date you click an "I Accept" button or check box presented with these CAS Terms or, if earlier, when you use any CA Service offerings. These CAS Terms are an addendum to the Terms and Conditions for AWS Marketplace Sellers (the "**AWS Marketplace Seller Terms**") between you and Amazon Web Services, Inc. ("**AWS,**" "**we,**" "**us**" or "**our**"), the terms of which are hereby incorporated herein. In the event of a conflict between these CAS Terms and the AWS Marketplace Seller Terms, the terms and conditions of these CAS Terms apply, but only to the extent of such conflict and solely with respect to your use of the CA Service. Capitalized terms used herein but not defined herein shall have the meanings set forth in the AWS Marketplace Seller Terms.

1. **CA Services and CAS Data.** To qualify for access to the CA Service, you must be an AWS Marketplace Seller bound by existing AWS Marketplace Seller Terms. Information and data you receive or have access to in connection with the CA Service ("**CAS Data**") constitutes Subscriber

Information and is subject to the restrictions and obligations set forth in the AWS Marketplace Seller Terms. You may use CAS Data on a confidential basis to improve and target marketing and other promotional activities related to Your AWS Marketplace Content provided that you do not (a) disclose CAS Data to any third party; (b) use any CAS Data in any way inconsistent with applicable privacy policies or law; (c) contact a subscriber to influence them to make an alternative purchase outside of the AWS Marketplace; (d) disparage us, our affiliates, or any of their or our respective products; or (e) target communications of any kind on the basis of the intended recipient being an AWS Marketplace subscriber.

2. **CA Service Limitations and Security.** You will only access (or attempt to access) the CA Service by the means described in the CA Service documentation. You will not misrepresent or mask your identity or your client's identity when using the CA Service. We reserve the right, in our sole discretion, to set and enforce limits on your use of the CA Service, including, without limitation, with respect to the number of connections, calls and servers permitted to access the CA Service during any period of time. You agree to, and will not attempt to circumvent such limitations. We reserve the right to restrict, suspend or terminate your right to access the CA Service if we believe that you may be in breach of these CAS Terms or are misusing the CA Service.
3. **CA Service Credential Confidentiality and Security.** CA Service credentials (such as passwords, keys and client IDs) are intended to be used by you to identify your API client. You are solely responsible for keeping your credentials confidential and will take all reasonable measures to avoid disclosure, dissemination or unauthorized use of such credentials, including, at a minimum, those measures you take to protect your own confidential information of a similar nature. CA Service credentials may not be embedded on open source projects. You are solely responsible for any and all access to the CA Service with your credentials.
4. **Modification.** We may modify these CAS Terms at any time by posting a revised version on the AWS Site or providing you with notice in accordance with the AWS Marketplace Seller Terms. The modified terms will become effective upon posting or, if we notify you by email, as stated in the email message. By continuing use or access the CA Service after the effective date of any modifications to these CAS Terms, you agree to be bound by the modified terms.
5. **Termination.** These CAS Terms and the rights to use CAS Data granted herein will terminate, with or without notice to you upon termination of your AWS Marketplace Seller Terms for any reason. In addition, we may stop providing the CA Services or terminate your access to the CA Services at any time for any or no reason.

Onboarding guide

You must configure your AWS account and AWS services to use the AWS Marketplace Commerce Analytics Service.

To use the AWS Marketplace Commerce Analytics Service

1. [Set up your AWS account with permissions.](#)
2. [Create a destination Amazon S3 bucket.](#)
3. [Configure an Amazon SNS topic for response notifications.](#)
4. [Enroll in the Commerce Analytics Service program.](#)
5. [Verify your configuration.](#)

Set up your AWS account with permissions

AWS Marketplace **strongly** recommends using AWS Identity and Access Management (IAM) roles to sign in to the AWS Marketplace Management Portal rather than using your root account credentials. See [the section called “Policies and permissions for AWS Marketplace sellers”](#) for specific IAM permissions for AWS Marketplace Commerce Analytics Service permissions. By creating individual users for people accessing your account, you can give each user a unique set of security credentials. You can also grant different permissions to each user. If necessary, you can change or revoke an user's permissions any time.

Create a destination Amazon S3 bucket

The Commerce Analytics Service delivers the data you request to an Amazon S3 bucket that you specify. If you already have an Amazon S3 bucket to use, proceed to the next step.

If you don't have an Amazon S3 bucket or you want to create an Amazon S3 bucket specifically for this data, see [How do I Create an S3 Bucket.](#)

Configure an Amazon SNS topic for response notifications

The Commerce Analytics Service delivers response notifications using Amazon SNS. The service publishes messages to this topic to notify you when your datasets are available or if an error occurred. If you already have an Amazon SNS topic for this purpose, proceed to the next step.

If you don't have an Amazon SNS topic configured for this service, configure one now. For instructions, see [Create a Topic.](#)

Record the topic Amazon Resource Name (ARN) for the topic you created, because the ARN is required to call the service.

Enroll in the Commerce Analytics Service program

The Commerce Analytics Service accesses the Amazon S3 bucket and Amazon SNS topic after you configure the service with the ARN for the topic and name of the bucket.

To enable access

1. Log in to the [AWS Marketplace Management Portal](#) with the AWS account you use to manage your AWS Marketplace products.
2. Ensure you have the [necessary IAM permissions](#) to enroll in the AWS Marketplace Commerce Analytics Service.
3. Navigate to the [Commerce Analytics Service enrollment page](#).
4. Enter the Amazon S3 bucket name and Amazon SNS topic ARN, and choose **Enroll**.
5. On the permissions page, choose **Allow**.
6. On the AWS Marketplace Management Portal, record the **Role Name ARN** in the success message. You need the ARN to call the service.

Note

Onboarding to the Commerce Analytics Service creates an IAM role in your AWS account. The IAM role allows AWS Marketplace to write to the Amazon S3 bucket and publish notifications to the Amazon SNS topic. AWS Marketplace uses the account 452565589796 to perform these associated actions with this IAM role.

Verify your configuration

The last step is to verify that your configuration works as expected.

To test your configuration

1. Download, install, and configure the [AWS Command Line Interface](#) (AWS CLI).
2. Using the AWS CLI, run this command.

```
aws marketplacecommerceanalytics generate-data-set \  
--data-set-type "customer_subscriber_hourly_monthly_subscriptions" \  
--data-set-publication-date "{TODAY'S-DATE}" \  
--role-name-arn "{YOUR-ROLE-NAME-ARN}" \  
--destination-s3-bucket-name "{YOUR-S3-BUCKET}" \  
--destination-s3-prefix "test-prefix" \  
--sns-topic-arn "{YOUR-SNS-TOPIC-ARN}"
```

- For `--data-set-publication-date`, replace `{TODAY'S DATE}` with the current date using ISO-8601 format, `YYYY-MM-DDT00:00:00Z`, where `YYYY` is the four-digit year, `MM` is the two-digit month, and `DD` is the two-digit day.
- For `--role-name-arn`, replace `{YOUR-ROLE-NAME-ARN}` with the ARN of the role you received from the enrollment process in [Enroll in the Commerce Analytics Service program](#).
- For `--destination-s3-bucket-name`, replace `{YOUR-S3-BUCKET}` with the Amazon S3 bucket you created in [Create a destination Amazon S3 bucket](#).
- For `--sns-topic-arn`, replace `{YOUR-SNS-TOPIC-ARN}` with the Amazon SNS topic you created in [Configure an Amazon SNS topic for response notifications](#).

If you receive a response including the `dataSetRequestId` response from the service, you've completed the on-boarding process. A successful response looks like this:

```
{  
  "dataSetRequestId": "646dd4ed-6806-11e5-a6d8-fd5dbcaa74ab"  
}
```

Technical implementation guide

The AWS Marketplace Commerce Analytics Service is provided through the [AWS SDK](#). This guide shows you how to interact with the service using the [AWS CLI](#) and the [AWS SDK for Java](#).

IAM policies for Commerce Analytics Service

To allow your users to use the Commerce Analytics Service, the following permissions are required.

Use the following IAM permissions policy to enroll in the AWS Marketplace Commerce Analytics Service.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:ListRoles",
        "iam:CreateRole",
        "iam:CreatePolicy",
        "iam:AttachRolePolicy",
        "aws-marketplace-management:viewReports"
      ],
      "Resource": "*"
    }
  ]
}
```

Use the following IAM permissions policy to allow an user to make requests to the AWS Marketplace Commerce Analytics Service.

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

For more information, see [Creating Policies in the IAM console](#) in the *IAM User Guide*.

Making Requests with the AWS CLI

To get started, download the [AWS CLI](#). The following AWS CLI example makes a request for the **Hourly/Monthly Subscriptions** dataset for October 1, 2017. This dataset is published to the **demo-bucket** Amazon S3 bucket using the prefix **demo-prefix**, and the notification message is delivered to the **demo-topic** Amazon SNS topic.

```
aws marketplacecommerceanalytics generate-data-set \
```

```
--data-set-type "customer_subscriber_hourly_monthly_subscriptions" \  
--data-set-publication-date "2017-10-01T00:00:00Z" \  
--role-name-arn "arn:aws:iam::123412341234:role/MarketplaceCommerceAnalyticsRole" \  
--destination-s3-bucket-name "demo-bucket" \  
--destination-s3-prefix "demo-prefix" \  
--sns-topic-arn "arn:aws:sns:us-west-2:123412341234:demo-topic"
```

This request returns an identifier that is unique for each request. You can use this identifier to correlate requests with notifications published to your Amazon SNS topic. The following example is an example of this identifier.

```
{  
  "dataSetRequestId": "646dd4ed-6806-11e5-a6d8-fd5dbcaa74ab"  
}
```

Making requests with the AWS SDK for Java

To start, download the [AWS Java SDK](#). The following AWS SDK for Java example makes a request for the **Hourly/Monthly Subscriptions** dataset for October 1, 2015. This dataset is published to the **demo-bucket** Amazon S3 bucket using the prefix **demo-prefix**, and the notification message is delivered to the **demo-topic** Amazon SNS topic.

```
/*  
 * Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License.  
 * A copy of the License is located at  
 *  
 * http://aws.amazon.com/apache2.0  
 *  
 * or in the "license" file accompanying this file. This file is distributed  
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either  
 * express or implied. See the License for the specific language governing  
 * permissions and limitations under the License.  
 */  
import java.text.DateFormat;  
import java.text.ParseException;
```

```
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.TimeZone;
import com.amazonaws.AmazonClientException;
import com.amazonaws.AmazonServiceException;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
import
    com.amazonaws.services.marketplacecommerceanalytics.AWSMarketplaceCommerceAnalyticsClient;
import
    com.amazonaws.services.marketplacecommerceanalytics.model.GenerateDataSetRequest;
import com.amazonaws.services.marketplacecommerceanalytics.model.GenerateDataSetResult;
/**
 * This sample demonstrates how to make basic requests to the AWS Marketplace Commerce
 * Analytics service using the AWS SDK for Java.
 * <p>
 * <b>Prerequisites:</b> Follow the on-boarding guide: {URL OR SOMETHING}
 * <p>
 * Fill in your AWS access credentials in the provided credentials file
 * template, and be sure to move the file to the default location
 * (~/.aws/credentials) where the sample code will load the credentials from.
 * <p>
 * <b>WARNING:</b> To avoid accidental leakage of your credentials, DO NOT keep
 * the credentials file in your source directory.
 * <p>
 * http://aws.amazon.com/security-credentials
 */
public class MarketplaceCommerceAnalyticsSample {
    public static void main(String[] args) throws ParseException {
        /**
         * The ProfileCredentialsProvider will return your [default]
         * credential profile by reading from the credentials file located at
         * (~/.aws/credentials).
         */
        AWSCredentials credentials = null;
        try {
            credentials = new ProfileCredentialsProvider().getCredentials();
        } catch (Exception e) {
            throw new AmazonClientException("Cannot load the credentials from the credential
                profiles "
                + "file. Make sure that your credentials file is at the correct "
                + "location (~/.aws/credentials), and is in valid
```

```
format.", e);
}
AWSMarketplaceCommerceAnalyticsClient client = new
    AWSMarketplaceCommerceAnalyticsClient(credentials);
Region usEast1 = Region.getRegion(Regions.US_EAST_1);
client.setRegion(usEast1);
System.out.println("=====");
System.out.println("Getting Started with AWS Marketplace Commerce Analytics Service");
System.out.println("=====
\n");
// Create a data set request with the desired parameters
GenerateDataSetRequest request = new GenerateDataSetRequest();
request.setDataSetType("customer_subscriber_hourly_monthly_subscriptions");
request.setDataSetPublicationDate(convertIso8601StringToDateUtc("2014-06-09T00:00:00Z"));
request.setRoleNameArn("arn:aws:iam::864545609859:role/
MarketplaceCommerceAnalyticsRole");
request.setDestinationS3BucketName("awsmp-goldmine-seller");
request.setDestinationS3Prefix("java-sdk-test");
request.setSnsTopicArn("arn:aws:sns:us-west-2:864545609859:awsmp-goldmine-seller-
topic");
System.out.println(
String.format("Creating a request for data set %s for publication date %s.",
request.getDataSetType(), request.getDataSetPublicationDate()));
try {
// Make the request to the service
GenerateDataSetResult result = client.generateDataSet(request);
// The Data Set Request ID is a unique identifier that you can use to correlate the
// request with responses on your Amazon SNS topic
System.out.println("Request successful, unique ID: " + result.getDataSetRequestId());
} catch (AmazonServiceException ase) {
System.out.println("Caught an AmazonServiceException, which means your request made it
"
+ "to the AWS Marketplace Commerce Analytics service, but was rejected with an "
+ "error response for some reason.");
System.out.println("Error Message: " + ase.getMessage());
System.out.println("HTTP Status Code: " + ase.getStatusCode());
System.out.println("AWS Error Code: " + ase.getErrorCode());
System.out.println("Error Type: " + ase.getErrorType());
System.out.println("Request ID: " + ase.getRequestId());
} catch (AmazonClientException ace) {
System.out.println("Caught an AmazonClientException, which means the client encountered
"
+ "a serious internal problem while trying to communicate with the AWS Marketplace"
+ "Commerce Analytics service, such as not being able to access the "
```

```

+ "network.");
System.out.println("Error Message: " + ace.getMessage());
}
}
private static Date convertIso8601StringToDateUtc(String dateIso8601) throws
    ParseException {
    TimeZone utcTimeZone = TimeZone.getTimeZone("UTC");
    DateFormat utcDateFormat = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ssX");
    utcDateFormat.setTimeZone(utcTimeZone);
    return utcDateFormat.parse(dateIso8601);
}
}

```

You should expect results similar to this example.

```

=====
Getting Started with AWS Marketplace Commerce Analytics Service
=====
Creating a request for data set customer_subscriber_hourly_monthly_subscriptions for
publication
date Sun Jun 08 17:00:00 PDT 2014.
Request successful, unique ID: c59aff81-6875-11e5-a6d8-fd5dbcaa74ab

```

Technical documentation

The service exposes one method, `GenerateDataSet`, which enables you to request datasets to be published to your Amazon S3 bucket. The following table lists the parameters for `GenerateDataSet`.

Dataset parameters

Field	Description
Data Set Type	This dataset will be returned as the result of the request.
Data Set Publication Date	The date a dataset was published.

Field	Description
	<p>For daily datasets, provide a date with day-level granularity for the desired day.</p> <p>For monthly datasets, provide a date with month-level granularity for the desired month. The day value is ignored.</p>
Role Name ARN	The ARN of the role with an attached permissions policy that provides the service with access to your resources.
Destination Amazon S3 Bucket Name	The name (the friendly name, not the ARN) of the destination Amazon S3 bucket. Your datasets are published to this location.
Destination Amazon S3 Prefix	<p>(Optional) The Amazon S3 prefix for the published dataset, similar to a directory path in standard file systems.</p> <p>For example, if given the bucket name <code>mybucket</code> and the prefix <code>myprefix/mydatasets</code>, the output file is published to <code>s3://DOC-EXAMPLE-BUCKET/myprefix/mydatasets/outputfile</code>.</p> <p>If the prefix directory structure doesn't exist, it's created.</p> <p>If no prefix is provided, the dataset is published to the Amazon S3 bucket root.</p>
SNS Topic ARN	The ARN for the Amazon SNS topic that is notified when the dataset has been published or if an error occurs.

Responses

The AWS Marketplace Commerce Analytics Service returns two responses. The first is synchronous, which is returned immediately, and the second is asynchronous, which is returned using the Amazon SNS. The synchronous response is similar to this example.

Data set parameters

Field	Description
Data Set Request ID	A unique identifier representing a specific request to the service. This identifier can be used to correlate a request with notifications on the Amazon SNS topic.

The asynchronous response is posted as a JSON-formatted document to your Amazon SNS topic and is similar to this example.

Dataset parameters

Field	Description
Data Set S3 Location	The bucket name and key for the delivered dataset.
Data Set Meta Data S3 Location	The bucket name and key for the delivered dataset metadata file.
Data Set Request ID	A unique identifier representing a specific request to the service. This identifier can be used to correlate a request with notifications on the Amazon SNS topic.
Success	"True" if the operation succeeded; "false" if not.
Message	(Optional) If an error occurred (for example, "Success" is "false"), this message contains information about the failure.

Example JSON-formatted asynchronous response

```
{
  "dataSetS3Location":{
    "bucketName":"demo-bucket",
    "key":"demo-prefix/
customer_subscriber_hourly_monthly_subscriptions_2014-06-09.csv"
  },
  "dataSetMetaDataS3Location":{
    "bucketName":"demo-bucket",
    "key":"demo-prefix/
customer_subscriber_hourly_monthly_subscriptions_2014-06-09.meta.json"
  },
  "dataSetRequestId":"f65b7244-6862-11e5-80e2-c5127e17c023",
  "success":true
}
```

Outputs

After a successful request, the requested dataset is delivered to your Amazon S3 bucket as a .csv file. A JSON-formatted metadata file is published to the same location as the dataset file. The metadata file provides useful information about the dataset and original request parameters. The metadata file has the same name as the dataset file, but ends with the extension .meta.json. The following table lists the metadata fields in the .csv file.

Metadata fields

Field	Description
Data Set Request ID	A unique identifier representing a specific request to the service. This identifier can be used to correlate a request with notifications on the Amazon SNS topic.
Data Set Coverage Range	Defines the start date/time and end date/time for the data coverage range. These dates are in ISO 8601 format.

Field	Description
Data Set Request Parameters	The original request parameters to the <code>GenerateDataSet</code> method.
Data Set S3 Location	The bucket name and key for the delivered dataset.
Data Set Meta Data S3 Location	The bucket name and key for the delivered dataset metadata file.

Following is an example of JSON-formatted metadata contents.

```
{
  "dataSetRequestId": "43d7137b-8a94-4042-a09d-c41e87f371c1",
  "dataSetCoverageRange": {
    "startDateTime": "2014-06-08T00:00:00.000Z",
    "endDateTime": "2014-06-08T23:59:59.000Z"
  },
  "dataSetRequestParameters": {
    "sellerAccountId": "123412341234",
    "dataSetType": "customer_subscriber_hourly_monthly_subscriptions",
    "dataSetPublicationDate": "2014-06-09T00:00:00.000Z",
    "roleNameArn": "arn:aws:iam::123412341234:role/MarketplaceCommerceAnalyticsRole",
    "destinationS3BucketName": "demo-bucket",
    "destinationS3Prefix": "demo_prefix/customer_subscriber_hourly_monthly_subscriptions",
    "snsTopicArn": "arn:aws:sns:us-west-2:123412341234:demo-topic"
  },
  "dataSetS3Location": {
    "bucketName": "demo-bucket",
    "key": "demo_prefix/customer_subscriber_hourly_monthly_subscriptions_2014-06-09.csv"
  },
  "dataSetMetaDataSetS3Location": {
    "bucketName": "demo-bucket",
    "key": "demo_prefix/customer_subscriber_hourly_monthly_subscriptions_2014-06-09.meta.json"
  }
}
```

For a complete list of available datasets, including availability dates, refer to the [AWS SDK documentation](#).

Troubleshooting

This section describes solutions to issues you may encounter with using the AWS Marketplace Commerce Analytics Service.

I can't access the service because of an allow list issue.

If you're not yet registered as a seller on the AWS Marketplace, visit [AWS Marketplace Management Portal](#) to register. If you have already registered as a seller on AWS Marketplace, contact the [AWS Marketplace Seller Operations](#) team.

I can't request datasets for a date in the past, even though the SDK documentation says it should be available for this date.

Even though datasets are listed as being available for certain dates in the past, we have data only since the time that you joined AWS Marketplace. If you believe that this is in error, contact the [AWS Marketplace Seller Operations](#) team.

When I call the service, I receive the error message "Could not connect to the endpoint URL: https://marketplacecommerceanalytics.eu-central-1.amazonaws.com/"

The AWS Marketplace Commerce Analytics Service is available only in the US East (N. Virginia) Region. You must make all calls to the Commerce Analytics Service to the us-east-1 endpoint.

If you're using the AWS CLI, add the `--region` flag to each call and specify the AWS Region as us-east-1, as shown in the following example.

```
aws marketplacecommerceanalytics generate-data-set \  
--data-set-type "customer_subscriber_hourly_monthly_subscriptions" \  
--data-set-publication-date "2016-04-21T00:00:00Z" \  
--role-name-arn "arn:aws:iam::138136086619:role/MarketplaceCommerceAnalyticsRole" \  
--destination-s3-bucket-name "marketplace-analytics-service" \  
--destination-s3-prefix "test-prefix" \  
--sns-topic-arn "arn:aws:sns:eu-  
central-1:138136086619:Marketplace_Analytics_Service_Notice" \  
--region us-east-1
```

I want to use a different Amazon S3 bucket or Amazon SNS topic than the ones I selected when I went through the on-boarding process.

When enrolling in the AWS Marketplace Commerce Analytics Service, you specified an Amazon S3 bucket and Amazon SNS topic. The onboarding process configures your IAM permissions to allow the service access to only these specific resources. To use different resources, you need to modify your IAM policy:

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. Choose **Roles** on the left side of the IAM console.
3. Choose **MarketplaceCommerceAnalyticsRole**.
4. Expand the **Inline Roles** section, if not already expanded.
5. Locate the policy with a name that starts with *oneClick_MarketplaceCommerceAnalyticsRole* and choose **Edit Policy**.
6. In the policy document, locate the section that specifies actions related to the service that you want to modify. For example, to change your Amazon S3 bucket, locate the section that includes the actions that start with **s3:** and change their respective **Resource** selection to specify your new Amazon S3 bucket.

For additional information about IAM policies, see the following guide: https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html

I get an `AccessDeniedException` error when I call the `GenerateDataSet` action

This can happen if your user doesn't have the permissions necessary to call `GenerateDataSet`. The following procedure outlines the steps needed to create an IAM policy with those permissions using the IAM console and add the permissions to your users, groups, or roles.

To use the JSON policy editor to create a policy


1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane on the left, choose **Policies**.

If this is your first time choosing **Policies**, the **Welcome to Managed Policies** page appears. Choose **Get Started**.

3. At the top of the page, choose **Create policy**.
4. In the **Policy editor** section, choose the **JSON** option.
5. Enter the following JSON policy document:

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

6. Choose **Next**.

 **Note**

You can switch between the **Visual** and **JSON** editor options anytime. However, if you make changes or choose **Next** in the **Visual** editor, IAM might restructure your policy to optimize it for the visual editor. For more information, see [Policy restructuring](#) in the *IAM User Guide*.

7. On the **Review and create** page, enter a **Policy name** and a **Description** (optional) for the policy that you are creating. Review **Permissions defined in this policy** to see the permissions that are granted by your policy.
8. Choose **Create policy** to save your new policy.

To provide access, add permissions to your users, groups, or roles:

- Users and groups in AWS IAM Identity Center:

Create a permission set. Follow the instructions in [Create a permission set](#) in the *AWS IAM Identity Center User Guide*.

- Users managed in IAM through an identity provider:

Create a role for identity federation. Follow the instructions in [Creating a role for a third-party identity provider \(federation\)](#) in the *IAM User Guide*.

- IAM users:
 - Create a role that your user can assume. Follow the instructions in [Creating a role for an IAM user](#) in the *IAM User Guide*.
 - (Not recommended) Attach a policy directly to a user or add a user to a user group. Follow the instructions in [Adding permissions to a user \(console\)](#) in the *IAM User Guide*.

My problem isn't listed here.

Contact the [AWS Marketplace Seller Operations](#) team.

AWS Marketplace Field Demonstration Program

The AWS Marketplace Field Demonstration Program (FDP) allows the AWS field team (internally approved AWS employees) to use some products and solutions through AWS Marketplace at no charge.

Examples of approved AWS employees may include solutions architects and sales and marketing professionals. The FDP allows these employees to demonstrate product capabilities for education and potential inclusion in customer workloads.

The following product types are supported:

- [Amazon Machine Images \(AMIs\)](#)
- [Containers](#)
- [Machine learning algorithms and model packages \(SageMaker\)](#)
- [Data sets \(AWS Data Exchange\)](#)

Note

For AWS Data Exchange products, the FDP only applies to products with a public offer of \$0 (free).

For AWS Data Exchange products that have subscription verification enabled, providers need to approve the subscription request. For more information about subscription verification, see [Subscription verification for subscribers](#) in the *AWS Data Exchange User Guide*.

You're automatically enrolled in the FDP program when you sign up as an AWS Marketplace seller. To opt out, submit a support request to the [AWS Marketplace Seller Operations](#) team.

To view information about product usage under this program, see the [AWS field demonstration usage](#) section of the [Monthly billed revenue report](#).

More resources in AWS Marketplace Management Portal

There are more resources available to you in the AWS Marketplace Management Portal. If you open the [AWS Marketplace Management Portal](#) and sign in, you can see links to additional resources on the home page, in the **Marketplace Resources** section.

For example, to get support for marketing your product in the 90 days leading up to launch and the 90 days after launch, you can see the [180-day GTM Academy](#) that is linked from the AWS Marketplace Management Portal home page under **Marketplace Resources**.

Preparing your product

Preparing to publish a product on AWS Marketplace includes configuring your package, setting a pricing scheme, determining the relevant categories in which to list your product, and adding keywords so your product appears in relevant searches.

Topics

- [Product delivery](#)
- [Product pricing](#)
- [Regions and countries for your AWS Marketplace product](#)
- [Standardized contracts in AWS Marketplace](#)
- [Categories and metadata](#)
- [AMI and container product usage instructions](#)
- [Search engine optimization for products](#)

Product delivery

Each product delivery method has several options for packaging, pricing, and delivery. Some methods aren't available to you as a seller on AWS Marketplace until you register for the program supporting it.

You can create products with a standard list price and end user license agreement (EULA). You can also create private offers for individual customers with custom pricing and EULAs. If you need to make additional changes to the terms of the contract, you can work with the AWS Marketplace team to create a custom private offer. To simplify the procurement process, you can use [standardized license terms](#) for both public product listings and private offers.

Note

AWS offers certain sellers the option to provide guided demonstrations on AWS Marketplace. If you're an AWS Partner Network (APN) partner who's eligible for [APN Customer Engagements \(ACE\)](#) and you'd like to provide this option to buyers, contact your AWS representative to determine further eligibility.

The following table lists the methods that you can use to deliver software products and how AWS Marketplace buyers find each type of deliverable in the AWS Marketplace console.

Product delivery methods

Product delivery method	Delivery Method filter on the console	Description
Single AMI	Amazon Machine Image (AMI)	<p>You deliver a single custom Amazon Machine Image (AMI) for your product. The AMI provides the information required to launch an Amazon Elastic Compute Cloud (Amazon EC2) instance.</p> <p>Buyers can use the single AMI to create Amazon EC2 instances with your product already installed and ready to use.</p> <p>For more information, see AMI-based products.</p>
AMI delivered using AWS CloudFormation templates	CloudFormation Template	<p>You can list AMI-based products that are delivered to AWS Marketplace buyers by using CloudFormation templates.</p> <p>Buyers can purchase a single solution that entitles them to all of the AMIs in that product.</p> <p>For more information about delivering AMIs as an CloudFormation template,</p>

Product delivery method	Delivery Method filter on the console	Description
		<p>see AMI-based delivery using AWS CloudFormation.</p> <p>For more information about CloudFormation templates , see AWS CloudFormation concepts in the <i>AWS CloudFormation User Guide</i>.</p>
Private image build	Private Image Build	<p>You offer products in a way that lets buyers install your product on a base gold image that meets their internal standards for operating system configuration.</p> <p>For more information, see Private images.</p>
Container-based product or application	Container	<p>You deliver products packaged in container images. Container products consist of options, which are a set of container images and deployment templates that work together.</p> <p>For more information, see Container-based products.</p>

Product delivery method	Delivery Method filter on the console	Description
Data products	AWS Data Exchange	<p>You use AWS Data Exchange to create data products.</p> <p>For information about publishing and managing data products and offers through AWS Data Exchange, see Providing data products on AWS Data Exchange in the <i>AWS Data Exchange User Guide</i>.</p>
Machine learning algorithms and model packages	SageMaker Model	<p>You use Amazon SageMaker to create the algorithm or model package, and then publish it on AWS Marketplace.</p> <p>For more information about delivering machine learning algorithms and model packages, see Machine learning products.</p> <p>For information about SageMaker, see What is SageMaker? in the <i>Amazon SageMaker Developer Guide</i>.</p>

Product delivery method	Delivery Method filter on the console	Description
Software as a service (SaaS)	SaaS	You can offer SaaS products with subscription- based, contract-based, or contract with consumption pricing models. For more information, see SaaS-based products .
Professional services	Professional Services	You can offer professional services that support or work with other AWS Marketplace products.

Product pricing

This topic provides general pricing information about software products in AWS Marketplace. All pricing is based on US dollars (USD).

For paid products, AWS Marketplace collects software charges from the customer.

There is no service fee for free or open-source software that is made available to customers without charge.

For information about refunds, see [Product refunds in AWS Marketplace](#).

Topics

- [Pricing models](#)
- [Changing pricing models](#)
- [Changing prices](#)
- [Private offers](#)
- [Product refunds in AWS Marketplace](#)

Pricing models

The following topics provide general information about the pricing models available in AWS Marketplace.

Topics

- [Annual pricing](#)
- [Usage pricing](#)
- [Contract pricing](#)
- [Bring Your Own License pricing](#)

For information about the pricing models for specific product delivery methods, see:

- [AMI product pricing](#)
- [Container products pricing](#)
- [Machine learning product pricing](#)
- [SaaS product pricing](#)
- [Professional services product pricing](#)

Annual pricing

An annual pricing model enables you to offer products to customers who can purchase a 12-month subscription. As an example, the subscription pricing can provide up to 40 percent savings compared to running the same product hourly for extended periods. The customer is invoiced for the full amount of the contract at the time of subscription. For more information about how annual subscriptions are presented to customers, see [AMI subscriptions](#) or [Pricing models for paid container products](#).

Considerations when working with an annual subscription include the following:

- Annual pricing is defined per instance type. It can be the same for all Amazon Elastic Compute Cloud (Amazon EC2) instance types or different for each instance type.
- All Annual instance types must also have an Hourly instance type defined. AWS Marketplace doesn't offer Annual-only pricing or Hourly without Annual on the same product. For any product offering Annual pricing, Hourly pricing also needs to be specified.

- A \$0 Annual price is allowed on a specific instance type, if the Hourly price is also \$0 and there are other non-\$0 Annual instance types defined.
- At the end of the annual subscription period, the customer will start being charged at the hourly price.
- If a customer buys X Annual subscriptions but is running Y software on Y instances, then the customer is charged at Hourly software price for (Y-X) instances which are not covered by Annual subscriptions. As such, an Hourly rate must be included for all Annual pricing instance types.
- Using seller private offers, you can offer a multi-year (up to 3 years) or custom duration AMI with upfront payment, or a flexible payment schedule. For more information about multi-year and custom duration contracts, see [Preparing your private offer](#) and [the section called "Installment plans"](#).

If you offer an Annual product in AWS Marketplace, you agree to the specific refund policies for Annual products, located in the **File Uploader** documents section in the [AWS Marketplace Management Portal](#).

Price change

You can change annual prices (the \$ value, for example \$1,000/year to \$1,200/year) every 90 days. However, you must give 90 days' notice to existing customers of annual pricing. The new price will apply to new subscriptions but will have no impact on existing subscriptions.

Note

During the 90 day notice period, you can't update the supported instance type.

Price changes will be effective for auto-renewals only if the price was changed at least 90 days before the auto-renewal date. The customer will receive an email message prior to auto-renewal that includes the new price.

End user license agreement

An AWS customer's usage of software for 12 months under an annual subscription is covered by the EULA that you provide on your product's details page on AWS Marketplace.

Usage pricing

A usage pricing model, also known as *pay as you go* pricing, enables you to offer products to customers who only pay for what they use.

As a seller, you can choose one of the following usage categories:

- **Users**
- **Hosts**
- **Bandwidth**
- **Data**
- **Tiers**
- **Units** (for custom categories)

You can also define up to 24 dimensions for the product. Charges are measured and reported when the API is called by the software. We recommend that sellers configure the API to be called once per hour as a best practice, depending on their use case. All usage is calculated monthly and billed monthly using the same mechanism as existing AWS Marketplace software.

Using the AWS Marketplace Metering Service, you can handle several new pricing scenarios.

Example Charge by Host

If your software monitors hosts, you can charge for each host monitored and set different pricing based on the host size.

Example Charge by User

If your software allows multiple users across an organization, you can charge by user. Each hour, the customer is charged for the total number of provisioned users.

Note

In the Product Load Form (PLF), relevant columns are preceded with "FCP" (Flexible Consumption Pricing). For example: **FCP Category (Custom Pricing Category)**.

For AWS Marketplace Metering Service products, note the following:

- If your software is already on AWS Marketplace, you will need to create a product to enable an alternate usage dimension. You can't convert a standard product to use the AWS Marketplace Metering Service. After the new product is published, you can remove the old product or keep both on the website.
- The AWS Marketplace Metering Service requires that your software reports usage every hour, recording the customer usage for the hour. If there is a failure in the transmission or receipt of metering service records, AWS will be unable to bill for such usage. You are responsible for ensuring the successful receipt of metering records.
- Products that use the AWS Marketplace Metering Service don't support 1-Click. Buyers are required to launch your software with an AWS Identity and Access Management (IAM) role with specific permissions and have an internet gateway.
- Free Trial and Annual Pricing aren't compatible with the AWS Marketplace Metering Service.
- Changing dimension (user, hosts, bandwidth, and data) or dimension name isn't supported. You will need to create a new product.

Contract pricing

Using the contract pricing model, you can offer upfront pricing to customers that enables them to buy a license for 1 month, 12 months, 24 months, or 36 months.

Contract pricing is available for the following products:

- Single AMI-based products and AMI with AWS CloudFormation template-based products. For more information, see [Contract pricing for AMI products](#)
- Container-based products. For more information, see [Contract pricing for container products](#).
- Software as a service (SaaS)-based products. For more information, see [Pricing for SaaS contracts](#).

Note

Contract pricing for AMI and container-based products is only for new products. If you have an existing AMI or container-based product and want to use contract pricing, create a new listing and then apply the contract pricing model by using the Product Load Form (PLF) to add different dimensions, integrate the AMI or container-based product with AWS License Manager, and then publish the AMI or container-based product.

When a customer purchases a product with contract pricing, a license is created by AWS Marketplace in the customer AWS account that your software can check using the License Manager API. Customers will need an IAM role to launch an instance of the AMI or container-based product.

Bring Your Own License pricing

There is no service fee for Bring Your Own License (BYOL) products on AWS Marketplace.

To deliver on our customer promise of selection, we require that all BYOL products also have a paid option. This is so that customers who don't have existing licenses have the option to purchase and use the products.

For BYOL products, we realize that the online purchase of software is a departure from how some companies do business. Therefore, for the first 90 days after launch, we will relax the requirement that this software is accompanied by a version available for purchase on AWS Marketplace. During this time, the AWS Marketplace account management teams will work with you to address challenges. The team can help you to determine if and how the software can be made available for purchase on AWS Marketplace.

Changing pricing models

Changes to pricing models must be reviewed and approved by AWS Marketplace to ensure a positive customer experience and reduced risk to all parties. Discuss the pricing model changes you want to make by contacting the [AWS Marketplace Seller Operations](#) team.

All requests for pricing model changes can take 30–90 days to process and review.

Changing prices

You can update prices and metadata through the AWS Marketplace Management Portal.

To change prices

1. Sign in to the [AWS Marketplace Management Portal](#).
2. In the **Products** tab, a list of current products that you created is available. You edit your product listing or request changes here.

Note

For new subscribers, the price change is effective immediately. For existing subscribers, the price change is effective on the first day of the month following a 90-day period that begins on the date that the price change notification is sent. For example, say you send a price change notification on March 16. June 16 is about 90 days after March 16. Because the price change happens on the first day of the month that follows the 90-day period, the effective date of the change is July 1.

Private offers

In the AWS Marketplace Seller Private Offer program, AWS Marketplace sellers can negotiate custom pricing and EULAs with individual AWS Marketplace customers (buyers). For more information, see [Preparing your private offer](#).

Product refunds in AWS Marketplace

All paid products in AWS Marketplace, regardless of pricing model, must have a stated refund policy for software charges. The refund policy must include the terms of the refund as well as a method of contacting the seller to request a refund. As a seller, the details of the refund policy are up to you. However, we encourage you to offer customers some manner of refund for usage of the product. You must comply with your posted refund policies. This topic provides information about the types of AWS Marketplace product refund requests, the related policy and approvals process, and how you can submit a refund request for a customer.

Refund request types for AWS Marketplace products

Customers can request different types of refunds for AWS Marketplace products. For AWS Marketplace products sold by AWS, refer to the refund policy page and then submit the contact support form using the AWS Support Center Console. If a customer requests a software refund directly from AWS, we instruct them to contact the seller using your posted support contact information for the product in question. Refunds of any AWS infrastructure charges are up to the discretion of AWS and are handled independently of software refunds.

If the product is sold by a third-party, buyers will be instructed to view refund policies on the product detail page. Software charges for AWS Marketplace subscriptions are paid to the seller of the product, and refunds must be requested from the seller directly. Each AWS Marketplace seller is required to include a refund policy on their AWS Marketplace page.

AWS Marketplace product refund policy and approvals

The following list outlines the AWS Marketplace refund policy and whether your approval is needed:

- **Free trials**

If you list your software as a free trial product, AWS can issue refunds on your behalf for software charges accruing within seven days of a conversion from a free trial to a paid subscription. Refunds issued in connection with free trial conversions require no action on your part. By enabling a free trial on a product, you are agreeing to this policy.

- **Private offers**

All refunds for private offers must be authorized by you before AWS can process them.

- **Software metering refunds**

If you meter the usage of your software by using the AWS Marketplace Metering Service, AWS can issue refunds on your behalf for software charges resulting from software metering errors. If these errors are common across multiple customers, AWS reserves the right to determine an appropriate refund for each customer and apply it directly to each customer. Refunds issued in connection with the AWS Marketplace Metering Service must be confirmed with the seller one time, but does not require the seller to confirm each individual refund. By using the AWS Marketplace Metering Service with a product, you are agreeing to this policy.

- **Subscription cancellation within 48 hours of purchase**

If a buyer cancels their subscription within 48 hours of a non-private offer purchase, AWS will issue a full refund (cancel with 100 percent refund). Refunds issued in connection with cancellation within 48 hours of purchase require no action on your part. After 48 hours, such buyer request is at your discretion. By listing your product on AWS Marketplace, you are agreeing to this policy.

- **Subscription upgrade**

If a buyer replaces an existing non-private offer subscription with a more expensive subscription or a subscription of equal value, AWS can issue refunds on your behalf for the lower-tier subscription. This is a two-step process for the buyer: Buy a new subscription and then request cancellation of the old subscription with a refund.

- **Subscription downgrade**

All downgrade subscription refund requests must be authorized by you before AWS can process them.

All AWS authorized refunds are processed automatically and require no action on your part.

AWS Marketplace product refund process

You can initiate refunds for your product software usage by submitting a [Refund Request Form](#). Once received by the AWS Marketplace Buyer Support Team, a related support case will be created in the [AWS Support Center Console](#), with the refund status noted in the subject line. Refund-related support is facilitated directly through these cases. For more information, see [Accessing AWS Support](#).

The following procedure outlines how to request a refund for an external customer or an internal testing account.

To initiate a software refund for a customer

1. Gather the following information from the customer:
 - The customer's email address that is associated with their AWS account.
 - The customer's AWS account number of the account used to subscribe to your product. Remind your customer that if they are the payer of an organization, they need to provide you with the AWS account ID for the linked account subscribed to your product.
 - The billing periods for which the customer would like a refund.
2. Sign in to your AWS account and then navigate to the [Refund Request Form](#).
3. Enter the customer's information in the form.
4. Enter the Product ID for the product that your customer is requesting a refund for. You can find the Product ID in your [daily customer subscriber report](#).
5. For annual products where a customer is requesting a refund, upgrade, or downgrade, you must perform the following tasks:
 - a. Verify the customer has purchased an annual subscription using your daily customer subscriber report (there might be a 24-hour delay).
 - b. Provide a **Subscription Cancellation Date** in the comments field.

- c. Provide a description of the change that you're authorizing (refund, upgrade, or downgrade) in the comments field.
6. Submit the form. We'll be notified and will begin to process the refund and issue it to the customer.
7. An outbound case will be created in the [AWS Support Center Console](#) with status information on the refund request. The subject line will contain one of the following:
 - **Completed** – The refund was processed and no further action is required.
 - **Pending** – The refund will be processed once the current billing cycle ends.
 - **Action Required** – The request could not be processed, and we need additional information from you. You can respond directly to the support case; however, you will also need to submit a new refund request form.
8. Once a refund is successfully processed, it will reflect on the customer's account within 24–48 hours. However, it can take up to five business days for the funds to appear in the customer's financial account.

Regions and countries for your AWS Marketplace product

When you create a product in AWS Marketplace, you choose the AWS Regions where it is available. You also choose the countries where buyers can purchase your product from. These two properties are similar, but they are not the same. For example, a buyer might be located in, and purchasing from, the United States but is installing your product in the Europe (Frankfurt) Region. In order for this buyer to purchase your product, you must include both the United States in your list of countries, and the Europe (Frankfurt) Region in your list of Regions.

AWS Regions

When creating or editing server or machine learning product information, you can limit your product to specific AWS Regions where your users can install and use the product.

For server products, including Amazon Machine Image (AMI)-, container-, and AWS CloudFormation-based products, you can select specific Regions where the product is available. You can also choose to automatically make your product available in new US Regions, non-US Regions, or all Regions as they become available.

For machine learning products, you can either select specific Regions, or all Regions including future Regions as they become available.

For more information about AWS Regions, see [AWS service endpoints](#) in the AWS General Reference.

Countries

By default, your product is available to buyers in all countries where AWS Marketplace is available. For new and existing server and software as a service (SaaS) products, you can control product availability in specific countries for tax, compliance, support, or marketing purposes.

There are exceptions to this functionality:

- **Previous purchases** – After updating your product with a new list of countries, buyers that have already subscribed to your product will still have access while their subscription is active.
- **Private offers** – When you limit your product to buyers in specific countries, it does not limit private offers. When you create a private offer to a specific buyer, it is available to that buyer, even if they are in a country that you did not include in your specified countries.

Note

Customer eligibility is determined at an AWS linked account level. For more information, see [How does AWS determine the Location of your account?](#)

Customers that share their entitlement can only activate the entitlement in a region you have allowed. For more information about managing entitlements, see [Sharing subscriptions in an organization](#) in the *AWS Marketplace Buyer Guide*.

Standardized contracts in AWS Marketplace

As you [prepare your product](#), you need to determine which end user license agreement (EULA) will govern the use of your product. You can either apply your own EULA or use the Standard Contract for AWS Marketplace (SCMP). The SCMP is a contract template that AWS Marketplace offers to help streamline procurement workflows and speed transactions. Also available is the Reseller Contract for AWS Marketplace (RCMP), which is a standardized reseller contract template that ISVs can use.

This section outlines two standardized contracts you can use in AWS Marketplace.

Topics

- [Standard Contract for AWS Marketplace](#)

- [Reseller Contract for AWS Marketplace](#)

Standard Contract for AWS Marketplace

AWS Marketplace developed the [Standard Contract for AWS Marketplace \(SCMP\)](#) in collaboration with the buyer and seller communities. The SCMP governs usage and defines the obligations of buyers and sellers for digital solutions. Examples of digital solutions include server software, software as a service (SaaS), and artificial intelligence and machine learning (AI/ML) algorithms).

The SCMP proactively defines common ground across key contractual clauses like use, warranty, indemnification, and governing law. Sellers can offer SCMP terms as the EULA for self-service transactions, where buyers can search for, buy, and quickly deploy solutions. For [private offers](#), buyers can request the SCMP template from the seller, and the terms can be amended to address custom transaction requirements as agreed upon by the parties.

You can also use the following optional addendums with the SCMP for self-service or private offers:

- [Enhanced Security Addendum](#) – Supports transactions with elevated data security requirements.
- [HIPAA Business Associate Addendum](#) – Supports transactions with Health Insurance Portability and Accountability Act of 1996 (HIPAA) compliance requirements.

Disclaimer

The EULA is between you and the buyer. Using the SCMP as your EULA is at your discretion. By applying the SCMP to your product listing, you are opting in to the SCMP program. Under this program, AWS may update the SCMP template periodically and may update product listings carrying the terms with the current version. You may withdraw from the SCMP program at any time by replacing the SCMP template with your own EULA.

Getting started with the SCMP

You can update a EULA to the SCMP and offer it to buyers of new and existing single Amazon Machine Instance (AMI) and software as a service (SaaS) products. The procedure that you use to request the update depends on whether a product is or is not listed through a self-service listing (SSL).

If you have questions, send an email message to the AWS Marketplace Standard Contracts team at aws-mp-standardcontract@amazon.com.

To update a EULA to the SCMP for AMI and SaaS products listed through SSL

1. Review the terms of the [Standard Contract for AWS Marketplace](#).
2. Sign in to the [AWS Marketplace Management Portal](#).
3. For products, choose the **product type** and select the **product listing** for which you want to update the contract.
4. Choose **Request Change** and then choose **Update regions and pricing**.
5. Choose **Standard Contract for AWS Marketplace** for the EULA if it's not already selected.
6. In **Notes & Notifications**, type **Please update this product to SCMP V2**.
7. Review the changes, and choose **Submit for review**.

To update a EULA to the SCMP for AMI and SaaS products not listed through SSL

1. Open the [Contact us](#) page on the AWS Marketplace Management Portal.
2. Sign in to your AWS Marketplace seller account.
3. Complete the form as follows:
 - For the subject of your question, choose **Commercial Marketplace**.
 - For the category, choose **Product Listing**.
 - For the subcategory, choose **Standard Contract Request**.
 - In the description, type **Please update these products to SCMP V2** and provide the product titles and IDs that you want to update with the SCMP.

Reseller Contract for AWS Marketplace

The Reseller Contract for AWS Marketplace (RCMP) is a standardized reseller contract template that ISVs can use when authorizing channel partners to resell ISV products to AWS Marketplace buyers. The contract helps reduce redundancy in legal contract reviews and accelerates time to market when ISVs and channel partners enter into a reseller relationship and/or use the reseller terms for their [channel partner private offer \(CPPO\)](#). When creating a CPPO, ISVs can upload the contract to the opportunity (Resale Authorization), and then channel partners can view and accept the contract. AWS Marketplace buyers can't view the RCMP.

Disclaimer

The RCMP is an optional contract for ISVs. If ISVs decide to resell their product through a channel partner, they can either attach the RCMP or their own customized contract terms—existing or pre-negotiated—when creating an opportunity.

Getting started with the RCMP

This section describes how to review terms and use the RCMP.

To use the RCMP while creating an AWS Marketplace opportunity

1. Review the terms of the [Reseller Contract for AWS Marketplace](#).
2. [Create a resell opportunity for a channel partner](#) while referring to the [RCMP guide](#).

Categories and metadata

Here are best practices and information for supplying product metadata. AWS Marketplace revises product metadata solely for quality assurance and error correction.

Naming and describing your product

The information that you provide about your product is visible to buyers. Ensure that potential buyers have enough information to make informed decisions about buying your product.

Creating the product name

Keep the following guidelines in mind as you create the product name:

- Use title case (capitalize the first letter of each important word)
- Ensure that a buyer can identify the product by the name alone
- Use the name of the brand or manufacturer
- Avoid descriptive data or hyperbole

Example product name: Smart Solution Load Balancer - Premium Edition.

Writing the product description

The product description lists the product's features, benefits, and usage. It can also provide other relevant, specific product information. The description can be up to 350 characters long.

Keep the following guidelines in mind as you write the product description:

- Avoid unnecessary capitalization
- Avoid unnecessary punctuation marks
- Don't include redirect information
- Check spelling and grammar
- Include only critical, useful information

Example product solution: Smart Solution automatically distributes incoming application traffic across multiple Amazon EC2 instances. It enables you to achieve even greater fault tolerance in your applications, providing the amount of load-balancing capacity you need to respond to incoming application traffic. Smart Solution detects unhealthy instances in a pool and automatically reroutes traffic to healthy instances until the unhealthy instances are restored. You can enable Smart Solution in a single AWS Availability Zone or across multiple Availability Zones to ensure more consistent application performance.

Writing the product highlights

The product information page displays up to three product highlight bullet points. Use these bullet points to briefly describe the product's primary selling points.

Example product highlight: Projecting costs: With Smart Solution, you pay only for what you use. You're charged for each hour or partial hour that Smart Solution is running.

Choosing categories and keywords

When you list your product, you can choose up to three software categories and corresponding subcategories for your product. This helps buyers discover your product as they browse or search for products on AWS Marketplace. Choose only categories that are relevant to your product; in most cases, only one category applies. The product load form and the **Products** tab both contain a complete list of categories.

Categories aren't the same as keywords. The categories and subcategories available are predefined for AWS Marketplace, and you decide which ones apply to your product by selecting them from

a list during the product request process. Keywords aren't predefined, but are created during the process. You don't need to add the category as a keyword.

Creating search keywords

During the product request process, you can enter up to three keywords (single words or phrases) to help buyers discover your product through site searches. The keywords field can contain a maximum of 250 characters.

The following tips can help you to create a relevant set of search keywords:

- Use relevant terms.
- Don't use the names of products published by other sellers or use other sellers' names.
- Choose keywords from your buyer's vocabulary—that is, words and phrases that buyers are likely to use when thinking about your type of product.
- Create keywords based on specific features in your product.
- Don't use the product title as a keyword. The product title is already indexed in searches.

Note: Keywords aren't the same as software categories. Keywords are more specific terms that are related to your product.

AMI and container product usage instructions

When creating usage instructions for your product, you must include the following information:

- Location of all sensitive information saved by customers
- Explain all data encryption configuration
- Step-by-step instructions for rotating programmatic system credentials and cryptographic keys. The [the section called “AMI-based product requirements”](#) explain the basic requirements for listings that use credentials and cryptographic keys.
- Provide detailed instructions on how the user interacts with your application to decrypt necessary data if your application makes use of any encryption techniques
- Step-by-step instructions for how to assess and monitor the health and proper function of the application. For example:
 - Navigate to your [Amazon EC2 console](#) and verify that you're in the correct region.

- Choose **Instance** and select your launched instance.
- Select the server to display your metadata page and choose the **Status checks** tab at the bottom of the page to review if your status checks passed or failed.

Writing the release notes

Each time you update a product, you must provide a description of the changes in the release notes. The release notes should contain specific information to help the user decide whether to install the update. Use clear labels for the update, such as "Critical" for a security update or "Important" or "Optional" for other types of updates.

Writing the usage instructions

Provide usage instructions that help ensure that the buyer can successfully configure and run the software. The usage instructions you provide are shown during the configuration process.

To write effective usage instructions, follow these guidelines:

- Write them with a new or moderately technical audience.
- Don't assume that the user has prior experience with or extensive knowledge of the product, computer operating systems, engineering, or IT operations.
- Take the buyer from launching to using the product, including any configuration or special steps to get the application running.

Example usage instructions:

1. Launch the product via 1-Click.
2. Use a web browser to access the application at `https://<EC2_Instance_Public_DNS>/index.html`.
3. Sign in using the following credentials:
 - Username: user
 - Password: the instance_id of the instance

Writing the upgrade instructions

Provide details on how buyer can upgrade from an earlier version of the product. Include information on how to preserve data and settings when creating another instance. If there is no upgrade path, edit this field to specifically mention that.

Example upgrade instructions:

1. Do ****, and then ****.
2. Check that all plugins used by your project are compatible with version *.* , by doing ***. If they aren't compatible, do ***.
3. Make a backup of your data, by doing ***.

CloudFormation delivery

When using CloudFormation delivery, you must also include the following:

- A purpose for each AWS Identity and Access Management (IAM) role and IAM policy created by the AWS CloudFormation template
- A purpose and location of each key created by the AWS CloudFormation template
- Network configuration details in deployments involving more than a single element
- A detailed guide on how your applications are launched and how they're configured to communicate if the deployment includes multiple AWS resources
- A pricing breakdown that includes the cost of running AWS resources added above the standard limits. Provide prescriptive guidance on managing AWS service limits.
- All data encryption configuration. For example: Amazon S3 server-side encryption, Amazon Elastic Block Store (Amazon EBS) encryption, Linux Unified Key Setup (LUKS), etc.)

Monitoring and assessing application functions

To monitor and assess application functions

1. Navigate to your [Amazon EC2 console](#) and verify that you're in the correct region.
2. Choose **Instances** and select your launched instance.
3. Select the server to display your metadata page and choose the **Status checks** tab at the bottom of the page to review if your status checks passed or failed.

Note

If any of the data stores are proprietary, provide step-by-step instructions for configuration, backup, and recovery.

Rotating programmatic system credentials and cryptographic keys

The [the section called “AMI-based product requirements”](#) explain the basic requirements for listings that use credentials and cryptographic keys.

Include the following for rotating programmatic system credentials and cryptographic keys:

- Prescriptive guidance on managing AWS service quotas. For more information see the [AWS General Reference Guide](#).
- A pricing breakdown including the cost of running AWS resources added above the standard quota. This can be included in your product usage instructions or linked to [documentation](#) containing detailed information about managing and requesting increased service quotas.

Search engine optimization for products

Search is a critical tool in the buyer’s journey that enables customers to find the best product to meet their unique needs. For AWS Marketplace customers, searches happen in two primary locations: search engines (for example, Google or Bing) and the search function in AWS Marketplace. This page provides information on how to optimize your listing for both locations.

Search engine optimization

Optimizing your product detail page to rank higher for relevant keywords is critical to driving more unique visitors to your product detail page through search engines.

There are three primary page elements that are foundational and impactful for improving organic search to any webpage, including product detail pages: keywords, title tags, and H1 heading tags.

Keywords

Keywords are a central element to search engine optimization, as they distill topics into focused queries that drive search engine results. The process of identifying the most relevant keywords

for your pages involves keyword research. Search engine optimization tools can provide valuable information, such as keyword search volumes (how many times per month a keyword is searched on Google), current rankings, search trends, keyword competitiveness, and related keywords. From this research, you can identify primary and secondary keywords.

Your primary search engine optimization keyword should be a unique single word or phrase that represents the main topic of your page. This primary keyword should be naturally woven into the copy of your product title, short description, and highlight section. Secondary keywords should be highly relevant terms that are found within the remaining page content.

Title tags

The title tag, which appears in the search engine results pages and as the title of a page in a browser window or tab, informs both readers and search engine web crawlers about the page's content. For AWS Marketplace product detail pages, the product title serves as the title tag, so it's important to optimize your product titles with search engine optimization keywords to improve ranking potential. To increase the likelihood of achieving a high rank in the search engine results pages, incorporate your brand name, product name, and relevant keywords in your title tag.

H1 heading tags

H1 heading tags have three roles:

- They help visitors scan the page contents for the information that they need.
- They increase accessibility for visitors with visual impairments that use screen readers to understand the page's content.
- They provide keywords found in page headings, which receive additional search engine optimization relevance weight if supported by the page content that follows.

AWS Marketplace search

The AWS Marketplace website ranks the results of search queries using search optimization techniques similar to those used across the industry. By understanding how AWS Marketplace ranks and returns search results, you can create product details optimized for the AWS Marketplace search engine. We recommend taking this guidance into consideration when you create your product detail pages.

Keywords

During the product creation process, you can submit up to three keywords (single words or phrases) to help customers discover your product through site searches. The keywords text box can contain up to 250 characters.

Use the following tips to create search keywords:

- Use terms that are relevant so that customers can easily find your products.
- Choose keywords from your customers' vocabulary—that is, words and phrases that they're likely to use when thinking about your type of product.
- Create keywords based on specific features in your product.
- Don't include the product title in the terms that you submit. The product title is already indexed in the search.

Note

Keywords aren't the same as software categories. Keywords are more specific terms that are related to your product.

You can edit keywords after you create a product by editing the metadata for the product. For products that you created using the **Products** tab in AWS Marketplace Management Portal, you also use the **Products** tab to make changes. For more information, see [Product changes and updates](#).

The AWS Marketplace Seller Operations team helps redirect queries with similar-sounding words or words with similar meanings. For example, when customers search for *automobile* when you expect them to search for *car*.

Software categories

When you list your product, you can choose up to three software categories and corresponding subcategories for your product. This helps customers discover your product as they browse or search the products on AWS Marketplace. Choose only categories that are relevant to your product. In most cases, only one category applies. Both the product load form and the **Products** pages contain a complete list of categories.

Note

Categories aren't the same as keywords. The available categories and subcategories are predefined for AWS Marketplace. You decide which of them apply to your product by choosing them from a list. Keywords aren't predefined, but they are created during the process.

Highlights section

The product details page displays up to three product highlights as bullet points. Customers can search for products by highlights, so include highlights when you create a product. A highlight should describe the product's primary selling points in brief and informative language.

Example Highlights

- Projecting costs: With AnyCompany's product, you pay only for what you use. You're charged for each hour or partial hour that it's running.

Short description

The product description lists the product's features, benefits, and usage instructions, along with other relevant and specific product information. Keep the following guidelines in mind as you create the product description:

- Avoid unnecessary capitalization and punctuation marks
- Don't include redirect information
- Check spelling and grammar
- Include only critical and useful information

Example Short description

AnyCompany's product automatically distributes incoming application traffic across multiple Amazon EC2 instances. It enables you to improve fault tolerance in your applications by seamlessly providing the load balancing capacity that you need to respond to incoming application traffic. AnyCompany's product detects unhealthy instances in a pool and automatically reroutes traffic to healthy instances until the unhealthy instances have been restored. Customers can enable it

in a single AWS Availability Zone or across multiple Availability Zones to enable more consistent application performance.

Preparing your private offer

Private offers are negotiated terms used to purchase a product from AWS Marketplace. This can involve a custom pricing plan, end user license agreement (EULA), or custom solutions. Sellers and buyers negotiate before committing to a private offer that's different from the public offer.

You can create and extend multiple private offers to a single buyer. Buyers that you extend the private offers to have the option to choose between the private offers and the public offer. Buyers can only be subscribed to one offer at any given time. They can't be subscribed to both a private offer and the public offer at the same time.

Note

AWS offers buyers with unique or enterprise use cases on AWS Marketplace to request a private offer for your product directly from the product detail page. If you're an AWS Partner Network (APN) partner who's eligible for [APN Customer Engagements \(ACE\)](#) and you'd like to provide this option to buyers, contact your AWS representative to determine further eligibility.

Topics


- [How private offers work](#)
- [Supported product types for private offers](#)
- [Creating and managing private offers](#)
- [AWS Marketplace Channel Partner private offers](#)
- [Installment plans](#)
- [Private offer upgrades, renewals, and amendments](#)
- [Future dated agreements and private offers](#)

How private offers work

You can create and manage your private offers from the **Offers** page in the [AWS Marketplace Management Portal](#). You specify the product for the offer to generate a unique ID and URL. You'll create a pricing plan for the private offer, add legal terms and sales documents, and extend the

offer to specific buyer AWS accounts. The offer is only visible to the accounts for which you created the offer.

After you create a private offer and notify potential buyers, they can view and accept the offer. To view the offer, the buyer must be signed into the AWS account that received the offer.

 **Note**

Buyers can't view the offer unless you extend it to either their linked account or their management account. You can't provide service limits in the offer, so the buyer can use as much of your product at the negotiated prices as they want, unless the product has a limit.

For information on creating a private offer, see [Creating and managing private offers](#).

Private offers are tracked in seller reports. For more information, see [Reporting for private offers](#) and the [Seller reports guide](#).

Private offer considerations

When working with private offers, consider the following:

- You can't create private offers for second party, Amazon Machine Image (AMI) monthly, or multi-AMI-based delivery using AWS CloudFormation products, or for limiting customer usage.
- For private offers with an installment plan, it's possible to break upfront payments into multiple payments over time. For more information, see [Installment plans](#).
- If the buyer account for your private offer is managed through a private marketplace, you must include both the buyer's account and the account that includes their private marketplace administrator in the offer.
- Private offers don't support the Bring Your Own License (BYOL) model.
- Use the **Custom EULA** option when creating a private offer with unique negotiated contract terms in your private offer. You can attach up to five documents.
- For software as a service (SaaS) contracts and SaaS contracts with consumption products, you can offer upgrades and renewals on agreements that were made when buyers accepted private offers. For example, you can do this to grant new entitlements, offer pricing discounts, adjust payment schedules, or change the end user license agreement (EULA) to use standardized license terms. For more information, see [Private offer upgrades, renewals, and amendments](#).

Private offer experience for buyers

When the buyer navigates to your product's subscription page, a banner indicates that a private offer is available. After the buyer accepts the offer, they're invoiced for the purchase using the same portal tools used for all AWS Marketplace transactions. Accepted offers become agreements. Buyers can find agreement details in the **Manage Subscriptions** section of the AWS Management Console, and sellers can find details in the **Agreements** tab of AWS Marketplace Management Portal.

AWS Marketplace buyers can access third-party financing for private offers. For more information, see [Customer financing is now available in AWS Marketplace](#).

Note

An offer can only be accepted before the expiration date. If the offer expires, it's moved to the **Accepted and expired offers** tab.

To view and accept a private offer	The buyer can
From the AWS Marketplace console	<p>Navigate to Private offers in the AWS Marketplace console and select the offer ID from the Available offers tab.</p> <p>For more information about the buyer experience for private offers, see Private offers in the <i>AWS Marketplace Buyer Guide</i>.</p>
Using a seller-provided link	<p>Follow the link sent by the seller to directly access the private offer.</p> <p>For more information, see Sending a private offer to a buyer.</p>
From your product page	<p>Navigate to the product page for the product, and choose the link in the banner to view the private offer.</p> <p>For more information about the buyer experience for private offers, see Private offers in the <i>AWS Marketplace Buyer Guide</i>.</p>

Reporting for private offers

Private offers appear on the existing seller reports and in the reports relevant to the offer. The [Monthly billed revenue report](#) is generated every month and has offer visibility and offer ID information. When an invoice is generated for a buyer, it appears in the report covering the appropriate billing period. For more information, see [Seller dashboards](#).

The **Offer ID** field contains the unique offer ID generated for the private offer. It's blank unless the report entry is for a private offer. The **Offer Visibility** field indicates whether the report entry is a public or private offer. For all private offers, the entry is marked private.

Supported product types for private offers

Amazon Machine Images (AMIs), container, professional services, and software as a service (SaaS) products are supported for private offers.

Private offers for AMI products

You can provide private offers pricing for AMI products.

The offer can be any custom duration for the following:

- AMI hourly or AMI hourly with annual private offers: up to 3 years (1,095 days). Only AMI hourly with annual private offers or AMI contracts support flexible payment scheduling.
- AMI contract private offers: up to 5 years (60 months)

For AMI contracts, private offers don't monitor usage.

Buyers can manually upgrade to new contract levels at any time. However, it is up to the independent software vendor (ISV) to define contract tiers, enforce service limitations, and advise buyers to manually upgrade their contracts with more units. Only non-tiered pricing-based contracts support upgrades at this time. The contract duration of the private offer can match the public product listing, or can be a custom duration in months (up to 60).

License entitlements begin on the date the buyer accepts the private offer.

For AMI private offers with flexible payment schedules, you can set the number of annual instance types agreed to in the contract, for the duration of the contract.

Note

Private offers are not available for monthly billing contracts.

Private offers for container products

You can provide private offers pricing for container-based product contracts.

The offer can be any custom duration for the following:

- Container hourly or container hourly with long term private offers – Up to 3 years (1,095 days). Only container hourly with long term private offers or container contracts support flexible payment scheduling.
- Container contract private offers – Up to 5 years (60 months)

For Container contracts, private offers don't monitor usage. Upgrading for container contracts is only possible if you're using non-tiered pricing.

Buyers can manually upgrade to new contract levels at any time. However, the independent software vendor (ISV) defines the contract tiers, enforces service limitations, and advises buyers to manually upgrade their contracts with more units. Only non-tiered pricing-based contracts support upgrades at this time. The contract duration of the private offer can match the public product listing, or it can be a custom duration in months (up to 60 months).

License entitlements begin on the date the buyer accepts the private offer. For container private offers with flexible payment schedules, you can set the number of units agreed to in the contract, for the duration of the contract. You can also define a custom hourly price for those same units if the buyer uses more.

Note

Private offers are not available for monthly billing contracts.

Private offers for professional services products

All professional services product offerings are done through private offers. For more information, see [Creating private offers](#).

Private offers for SaaS products

Software as a service (SaaS) private offer products can't change the pricing level for a given pricing tier based on timing. For example, an offer can't charge \$0.80/hour for three months and then change pricing to \$0.60/hour thereafter for the same pricing tier. For SaaS contracts, private offers don't monitor usage.

Buyers can manually upgrade to new contract levels at any time. However, the independent software vendor (ISV) defines contract tiers, enforces service limitations, and advises buyers to manually upgrade to higher contract tiers when needed. The contract duration of the private offer can match the public product listing, or it can be a custom duration in months (up to 60 months).

Private offers for ML products

Machine Learning (ML) private offer products give specific buyers a different price than your publicly displayed price. The set of terms and agreement between you and the buyer in private offers can differ from the one in the public offer or other private offers.

Private offers work in one of several ways:

- **Hourly** – Private offers can be an hourly rate that is different from the publicly displayed hourly rate. This hourly rate is perpetual because private offers for machine learning products don't expire. If a price change is needed in the future, the buyer must switch to the new private offer. Existing running instances or endpoints of the product are automatically billed the hourly rate set in the new accepted offer. Ensure you set it to the hourly rate for your product after any contract component within the private offer expires. Setting this hourly rate to \$0 allows the buyer to use the product without your software fee indefinitely.
- **Per inference** – Private offers can have an inference rate that is different from the publicly displayed inference rate, if you've configured [inference pricing](#) for when your product is deployed as an endpoint.
- **Contract** – Private offers can be a contract with a fixed upfront fee for a specified number of days. The buyer is allowed to use an unlimited number of instances for the entire duration of the contract. At the end of the contract, any instances that continue to run are billed at the hourly rate that you set in the private offer. For example, you can create a contract with a fixed upfront

fee for 365 days of unlimited use. You also set an hourly rate for the private offer. When the buyer accepts this private offer, they pay that upfront fee. When the contract ends, any instances still running are billed at that hourly rate. If you're offering a free private trial, ensure you set the correct hourly rate after the free trial period ends to avoid a free perpetual license.

You can create and extend multiple private offers to a single buyer. Buyers that you extend the private offers to have the option to choose between the private offers and the public offer. Buyers can only be subscribed to one offer at any given time. They can't be subscribed to both a private offer and the public offer at the same time.

To create a private offer for a specific buyer for SageMaker products, contact the [AWS Marketplace Seller Operations](#) team.

You must provide the following information when requesting to create a private offer: **ProductId**, **Targeted buyer AWS accounts**, **Date and time** (indicating when an offer must be accepted by), **Custom EULA** file (optional), **Refund policy**, **License duration** (optional), **License price** (optional), **Flexible payment schedules** (optional), and **Prices for each instance type**. After the offer is created, the buyer must accept it within the date and time specified.

Creating and managing private offers

The following sections describe how to create and manage private offers.

Topics

- [Starting a new private offer](#)
- [Understanding offer statuses](#)
- [Drafting and publishing the private offer](#)
- [Sending a private offer to a buyer](#)
- [Saving your private offer progress](#)
- [Updating the expiration of a private offer](#)
- [Cancelling a private offer](#)

Starting a new private offer

Use the following process to create an offer and generate an offer ID using the CreateOffer API change request. It creates a blank offer in a draft state.

To start creating a new private offer

1. Sign into the [AWS Marketplace Management Portal](#), and choose **Offers**.
2. On the **Offers** page, choose **Create offer**.
3. On the **Create offer** page, choose the product type and the product you want to create your private offer from. Processing will take up to 30 seconds. Don't close or refresh the page during this process.

Note

You won't be able to change the product type and product after the offer is created. For more specific information on private offers per product type, see [Supported product types](#).

If you're an AWS Marketplace Channel Partner, choose whether you're creating an offer for your own product or an AWS Marketplace Channel Partner private offer (CPPO) from a resale authorization. If it's a CPPO, choose the independent software vendor (ISV), product, and authorization.

4. Choose **Continue to offer details**. A step-by-step experience will open so you can continue creating your private offer.

Understanding offer statuses

Offers have one of three statuses depending on the lifecycle:

- **Draft** – The offer is incomplete and still being prepared by you. All required details must be completed and submitted to publish the offer and extend it to your buyer.
- **Active** – The offer is published and extended to the buyer. The offer hasn't expired, so buyers can subscribe to the offer.
- **Expired** – The offer is published and extended to the buyer. The offer has expired, so buyers can't subscribe to the offer. The expiration date can be updated to give your buyers more time to accept the offer. To update offer expiration, see [Updating the expiration of a private offer](#).

Note

After the offer is accepted, it will show up as an agreement in the **Agreements** tab. The status of the offer won't change.

Drafting and publishing the private offer

Use the following process to draft and publish your private offer.

To draft and publish your private offer

1. On the **Provide offer information** page, provide the offer name, offer details, renewal type, and offer expiration date. If this is a renewal offer, you must choose either **Existing Customer on AWS Marketplace** for renewals intended to renew an existing agreement created on AWS Marketplace, or **Existing Customer Moving to AWS Marketplace** for renewals intended to migrate your existing customer to AWS Marketplace.

Note

The offer expiration date is the date that the offer becomes null and void. After 23:59:59 UTC on this date, the buyer won't be able to view and accept this private offer.

2. Choose **Next**.
3. On the **Configure offer pricing and duration** page, choose the pricing model, contract or usage duration, pricing, and payment schedule. For pricing models that have an installment plan, see [Installment plans](#).
4. On the **Add buyers** page, provide an AWS account ID for each AWS Marketplace buyer you are extending the private offer to. Choose the **Add another buyer** button to add another AWS account ID. You can add up to 24 buyers to each private offer.
5. Choose **Next**.
6. On the **Configure legal terms and offer documents** page, choose one of the following options:
 - Public offer end user license agreement (EULA) – Use the EULA from your public offer.

- Standard contract for AWS Marketplace (SCMP) – Use the standard contract provided by AWS Marketplace.
 - Custom legal terms – Upload up to five files related to your private offer, including legal terms, a statement of work, a bill of materials, a pricing sheet, or other addendums. These files will be merged into one document when the offer is created.
7. On the **Review and create** page, review the details of your private offer. After you review and confirm, choose **Create offer** to publish the offer and extend it to the buyers you chose. Offer publishing includes a request to the AWS Marketplace Catalog API, so it can take up to an hour to validate and process the offer. This request can be viewed on the **Requests** page.

 **Note**

The offer will be published and extended only if the request succeeds. If the request fails, it won't be extended to the customer. A failure means there was either a system error or an error you'll need to correct before resubmitting.

The following guides provide more information about creating private offers for specific products.

- [AWS Marketplace – Create SaaS PAYG Private Offers](#)
- [AWS Marketplace – Create AMI Hourly/PAYG Private Offers](#)
- [AWS Marketplace – Create AMI Hourly with Annual Private Offers](#)
- [AWS Marketplace – Create SaaS Contract Private Offers](#)

The following video explains more about creating a SaaS contract private offer.

Sending a private offer to a buyer

After the private offer has been published, buyers can view it by navigating to the **Available private offers** tab on the **Private offers** page in the AWS Marketplace Management Portal. On the **Available private offers** tab, the buyer can see offers extended by AWS Marketplace Channel Partners in the **Seller of record** column. The independent software vendor (ISV) will display in the **Publisher** column. A buyer can navigate to a private offer by choosing the appropriate **Offer ID** in their offers list.

Buyers can view offer IDs that have been accepted or that have expired on the **Accepted or expired offers** tab.

After the private offer has been published, you can send your buyer a URL to the fulfillment page for the offer.

To send private offer to your buyer

1. Sign into the [AWS Marketplace Management Portal](#), and choose **Offers**.
2. Select the **radio button** next to the offer.
3. Choose **Actions** and then **Copy Offer URL**.
4. Send the URL to your buyer.

Saving your private offer progress

Use the follow pricess to save your progress and resume later.

To save and resume your work

1. At any completed step, choose **Save and exit**. In the dialog box, confirm that you're saving the content in a draft state and review any validation errors. If there are any validation errors or missing details, you can choose **Fix it** to go to the step and resolve the issue. When you're ready, choose **Save and exit** to save your changes.

After you save and exit, the request is under review while it's processing. It could take a few minutes or hours to finish processing. You can't continue the steps or make changes until the request has succeeded. After the request has succeeded, you have completed the save. If the request fails, there was either a system error or an error you'll need to correct before resubmitting.

2. To resume working on your offer, open the **Offers** page, choose your offer, and then choose **Resume offer creation**.
3. When you're finished, you can choose either **Save and exit** to save your progress or **Create offer** to publish and extend the private offer to your selected buyers.

Updating the expiration of a private offer

Use the following process to update the expiration date of a private offer.

To update the expiration date of a private offer

1. Sign into the [AWS Marketplace Management Portal](#), and choose **Offers**.
2. On the **Offers** page, choose the **offer** you want to update.
3. Choose **Edit**.
4. Provide a new **offer expiration date**.
5. Choose **Submit**.

After the update is complete, the offer will change to an **Active** status and your buyer can accept the offer.

Canceling a private offer

Use the following process to cancel the private offer.

1. Sign into the [AWS Marketplace Management Portal](#), and choose **Offers**.
2. On the **Offers** page, choose the **offer** you want to update.

Note

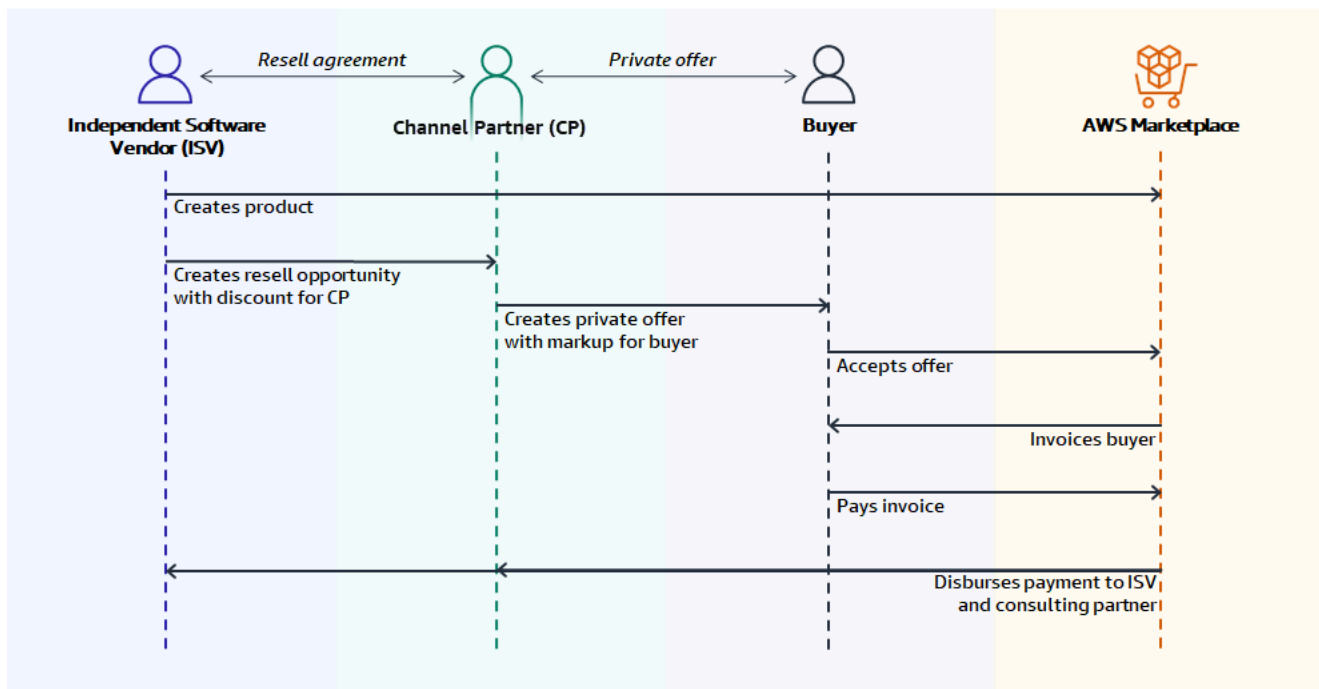
Canceling the offer will modify the offer expiration date, so the offer will display as expired for buyers who were extended this offer.

3. Choose **Action** and then choose **Cancel offer**.

AWS Marketplace Channel Partner private offers

AWS Marketplace Channel Partner private offers allow channel partners to resell independent software vendors' (ISVs) products on AWS Marketplace. The AWS Marketplace Channel Partner and ISV establish an agreement to resell one or more of the ISV's products, and then they extend a private offer to the buyer for that product.

The following diagram shows this relationship between an ISV, a channel partner, and a buyer.



Note

For more information about creating a resell opportunity for a channel partner, as an ISV, see [Creating a resell opportunity for an AWS Marketplace Channel Partner as an ISV](#).

Each AWS Marketplace Channel Partner private offer is visible only to a single buyer, with customized pricing and unique commercial terms to meet that buyer's needs. When creating a private offer, you start from a wholesale cost set by the ISV. Then you mark up that price to create the buyer's offer price. The wholesale cost is determined in one of two ways:

- **Recurring discount** – An ISV authorizes the AWS Marketplace Channel Partner to resell their product or products at an agreed-to discount from their list price with a recurring opportunity. This discount allows the AWS Marketplace Channel Partner to continue to resell the product without further price negotiation with the ISV. This discount can be set up to last until a specified date, or indefinitely, until ended by either the ISV or the channel partner.
- **Non-recurring discount** – The opportunity that the ISV gives the AWS Marketplace Channel Partner is a one-time discount intended to be used only with a specific buyer.

In both cases, after the buyer pays for the private offer, AWS Marketplace uses the standard process to distribute the funds to the AWS Marketplace Channel Partner and the ISV based on the agreed-to pricing.

Tip

As an ISV or a channel partner, you can view opportunities that you have granted or received from the **Partners** menu of the [AWS Marketplace Management Portal](#).

For detailed instructions about creating private offers, see [AWS Marketplace Channel Partner creates](#).

For information about third-party financing for private offers, see [Customer financing is now available in AWS Marketplace](#).

Additional information

For additional information and questions, we encourage ISVs and channel partners to connect with the AWS Marketplace channel team. If you don't know who to contact specifically, send an email message to aws-mp-channel@amazon.com, and someone on the team will respond to you within one business day.

Creating a resell opportunity for an AWS Marketplace Channel Partner as an ISV

As an ISV, you can authorize AWS Marketplace Channel Partner to resell your products by creating a resale *opportunity* for that partner. You can specify a discount percentage or custom price per product dimension to create a wholesale price for the AWS Marketplace Channel Partner. The partner can mark up the wholesale price when creating their AWS Marketplace Channel Partner private offer for a buyer. Supported product types include:

- AMI-based products
- Container-based products
- SaaS-based products
- Professional services products

The following procedure outlines how ISVs can create an opportunity for an AWS Marketplace Channel Partner. To use this feature, you must have permissions to use the **Partners** tab in the AWS Marketplace Management Portal. For more information, see [Policies for AWS Marketplace sellers](#).

To create a reseller opportunity for an AWS Marketplace Channel Partner as an ISV

1. Sign in to the [AWS Marketplace Management Portal](#) with your AWS Marketplace Seller account.

Tip

Be sure that you are signed out from another AWS account before signing in with your AWS Marketplace Seller account.

2. Choose the **Partners** tab, and then choose **Create opportunity**.
3. On the **Opportunity details** page, enter the **Opportunity name** and **Opportunity description**.

Note

The information you enter in **Opportunity name** and **Opportunity description** will be visible to channel partners in their seller reports.

4. For **Resellers**, choose the AWS Marketplace Channel Partner (reseller) that you want to authorize from the dropdown list. You can select resellers by name or account ID.
5. Select one of your **Products** to be part of this opportunity.
6. Choose the **Discount** that you want to apply.

Discount types can be issued in multiple ways:

- **Percentage Discount** – Applies one discount rate (a percentage) to all selected products.
- **Individual Pricing** – Applies specific discounts to specific products.
- **Flexible payment schedule** – Sets a flexible payment schedule for an AWS Marketplace Channel Partner opportunity.

Note

Only individual pricing and flexible payment schedule are supported for professional services sellers.

7. Select the **Duration** of the opportunity.

Note

The start date for resellers must be earlier than the date that the manufacturer has listed in the resale authorization.

Opportunity time length can be issued in multiple ways:

- **Single Use** – Applies to one opportunity and is no longer applicable after the AWS Marketplace Channel Partner creates the private offer.
 - **Specific Time Duration** – Lasts for a specific time duration that is no longer applicable after a date selected by the ISV.
 - **No Set Time Duration** – Lasts until ended by one of the involved parties.
8. (Optional) For SaaS contract products, add or remove custom **Product dimensions** and modify the **Additional usage fees** to customize your opportunity.
 9. (Optional) Set one or more **Buyer account IDs** to specify that the opportunity is only for those buyers.
 10. (Optional) Select the **End User License Agreement (EULA)** version or upload the EULA to be included in the opportunity.

Note

Only custom EULAs are supported for professional services sellers.

11. (Optional) Select the **Reseller Contract for AWS Marketplace (RCMP)** or upload a custom contract to be included in the opportunity.
12. Select **Review opportunity**, and make sure that the information is correct.
13. Select **Create opportunities** to finalize the opportunity and authorize the channel partners.

The **Opportunities created** table is updated to display relevant opportunity details including **Opportunity name, Product name, Reseller name, Discount, Created date, and Status.**

After opportunities are created, you can't extend their dates. However, you can revoke an opportunity and re-create it at any time. When you revoke an opportunity, new offers can't make use of that discount. Any existing offers are unaffected and retain their opportunity discount.

You can also clone an opportunity by selecting the opportunity and then choosing **Clone**. This will prepopulate everything and then you can edit fields

Installment plans

Installment plans (also known as *flexible payment schedules*) enable you to extend private offers with a custom payment schedule. The schedule can be spread over the accepted contract duration, and the customer makes payments in regular installments. After they're subscribed, your customers can see all the payments on the schedule and on their AWS invoice, helping them track their spending. Installment plans are available for private offers on certain product and pricing types. For more information, see [Product types eligible for private offers](#).

Creating a payment schedule

The process for creating a custom payment schedule using installment plan is part of the process for creating a private offer. To enable the installment plan option for your private offer, choose **Contract pricing with installment plan** in the **Buyer payment options** section when creating the private offer. After you choose a buyer payment option, choose the contract duration for this offer and specify the offer details. For more information, see [Private offers](#).

Note

For private offers with an installment plan, for multi-year and custom duration Amazon Machine Image (AMI) products, set the number of instances for each instance type included in the offer and the hourly pricing for any additional launched instances. After the customer launches the specified number of instances, any additional instances launched are charged at the hourly rate specified in the private offer.

Under **Buyer installment plan**, provide the **Contract total**, **Number of payments**, and **First invoice date**, and then choose **Generate installment plan**. You can add up to 60 payments. You also have the option to make adjustments to each payment line item. Each time you adjust a payment line item, the **Total amount due from buyer** is updated.

Note

The invoice date for the first installment is the first time that the customer is invoiced for your private offer. You receive the payment for each invoice after AWS Marketplace receives the payment from the customer.

The flexible payment scheduler feature validates that the invoice dates fall within the contract duration. If any invoice dates are after the duration of the contract, you receive an error message.

After you verify all invoice amounts and dates, confirm that the **Total amount due from buyer** matches the total price that you want your customer to pay over the course of the private offer. To finish creating the private offer, complete the remaining steps.

Note

Only one invoice date can occur before the offer acceptance date that you're extending to your customer.

Your customer is invoiced based on the schedule that you defined, and invoices start after they accept the offer. If the first invoice date is scheduled before the offer is accepted, it's processed immediately after the offer is accepted.

Note

You can't modify the payment schedule on a private offer that has been extended to and subscribed by a buyer. To make changes, you must create a new offer.

Reporting for installment plans

Reporting for private offers with flexible payment schedules is in the [Section 4: Contracts with flexible payment schedule](#), of the monthly billed revenue report.

Private offer upgrades, renewals, and amendments

Managing agreements for private offers

For software as a service (SaaS) contract and SaaS contract with consumption products, you can offer upgrades, renewals, and amendments by using a private offer on any active agreements. For example, you can do this to grant new entitlements, offer pricing discounts, adjust payment schedules, or change the end user license agreement (EULA) to use [standardized license terms](#). You can also change the number of units and payment schedule, and add a custom end date.

The difference between an *offer* and an *agreement* is whether the buyer accepted its terms:

- An **offer** is a set of terms for a buyer's use of a product. Offers can be public or private.
- An **agreement** is an offer that a buyer accepted. Agreements include purchased and free products that a seller made available using a public or private offer.

This feature is available to all AWS Marketplace sellers, including independent software vendors (ISVs) and channel partners. You can't amend an agreement to specify a seller of record that's different from the seller of record from the original agreement.

To use this feature, you must have permissions to use the **Agreements** tab in the AWS Marketplace Management Portal. For information, see [Permissions for AWS Marketplace sellers](#).

Supported product types for private offer amendments

You can view the following product types on the **Agreements** page:

- SaaS contracts
- SaaS contracts with consumption
- SaaS usage-based products
- AMI-based products
- Container-based products
- Server contract
- Professional services products

However, only the following product types support private offer amendments:

- SaaS contracts
- SaaS contracts with consumption

Submission process for upgrades and renewals

You can create private offer upgrades, renewals, and amendments from the AWS Marketplace Management Portal by using the following procedure.

To create private offer upgrades, renewals, and amendments

1. Sign in to the [AWS Marketplace Management Portal](#) and choose **Agreements**.
2. On the **Agreements** page, choose a check box next to an agreement, and then choose **View Details**.
3. On the **View agreement** page, choose **Create agreement-based offer**.
4. On the **Amend agreement details** page, sellers must indicate whether a private offer is for a renewal. Select **Yes** or **No** when asked if the private offer is for a renewal.

Note

You can also make changes to service dates, product dimensions, the payment schedule, the renewal status, and the offer expiration date on the **Amend agreement details** page.

5. When you're finished, choose **Create offer** and then **Submit**.

Tip

Entering descriptive custom offer names can help you distinguish between your active offers on the **Offers** page. Custom offer names are also visible to buyers. AWS recommends that you specify a custom offer name that includes any additional identifying details, such as your own IDs and purchase order numbers. Using high-level descriptions like **upgrade** or **renewal** and custom company names are also recommended. Don't use any personally identifiable data (for example, first or last names, phone numbers, or addresses). You can enter up to 150 characters for this field.

Edit the information for any dates, dimensions, payment schedules, and EULAs that you want to change, and then choose **Next**. On the **Review and create** page, review the information. When you're ready, choose **Create agreement-based offer**.

The new private offer appears on the **Manage Private Offer** page in approximately 45 minutes. To view the offer, sign in to the AWS Marketplace Management Portal and choose **Offers** to open the **Manage Private Offer** page.

From there, the buyer has the option to accept it or to continue to operate under the original agreement:

- If the buyer accepts the private offer upgrade or renewal, the new agreement takes effect immediately and the agreement is listed on the **Agreements** page in the AWS Marketplace Management Portal. Any remaining scheduled payments from previous agreements are cancelled.

Buyers accept agreement-based private offers the same way they accept private offers. For more information about the buyer experience for private offers, see [Private offers](#) in the *AWS Marketplace Buyer Guide*.

- If the buyer doesn't accept the private offer upgrade or renewal before it expires, the original agreement remains in effect with no changes.

Reporting for upgrades, renewals, and amendments

Upgrade and renewal private offers appear on the existing seller reports and in the reports relevant to the offer. The [Daily customer subscriber report](#) report and [Daily business report](#) report are generated daily. The [Monthly billed revenue report](#) report is generated monthly.

In the Daily customer subscriber report, the **Subscription intent** field indicates whether the report entry is a new private offer. The **Previous offer ID** field indicates the ID of the offer that preceded the new offer, if one exists. For all private offers, the entry is marked **private**.

Future dated agreements and private offers

In a future dated agreement (FDA) in AWS Marketplace, the buyer receives the product license or entitlement on a *predetermined future date*. In a typical AWS Marketplace transaction, the

buyer receives the product license or entitlement *immediately* after the offer is accepted or the agreement is created.

As a seller using FDA, you can close transactions with buyers when they choose instead of when the buyer wants to begin product usage. FDA helps sellers perform the following actions independently for transactions on AWS Marketplace :

- Book (buyer accepts the offer) the deal based on sales needs.
- Charge the buyer based on your finance or accounting needs.
- Provide buyer access to the product, such as activating a license or entitlement, based on buyer needs.

FDA can be used to setup renewals for existing transactions with the buyer.

FDA is supported for software as a service (SaaS) products for contract and contracts with consumption (CCP) pricing, with and without flexible payments.

When you use future dated agreements, keep the following dates in mind :

Agreement sign date

The date when the buyer accepts the offer and when the agreement is created.

Agreement start date

The date when the buyer's license or entitlement to the product is activated and the buyer can begin using the product.

Agreement end date

The date when the agreement ends. The agreement and the buyer's license or entitlement expire on this date.

Topics

- [Creating future dated agreements](#)
- [Using an installment plan with future dated agreements](#)
- [Receiving notifications for future dated agreements](#)
- [Using future dated agreements with reselling for Channel Partner private offers](#)

Creating future dated agreements

The seller of record sets the agreement start date when generating a private offer with a future start date. Buyers can't change the start date, but they can review the start date before accepting the private offer in AWS Marketplace.

To create a private offer with a future start date

1. When creating a private offer, choose **Start at a future date** under **Contract duration**.
2. In the **Service dates** section, enter the **Service start date** and **Service end date**. The service start date you choose here will be the agreement start date of your future dated agreement when the buyer accepts the offer.

Note

To use an FDA for renewals, align the service start date with the end date of the agreement that you want to renew.

Sellers can choose a service start date up to 3 years in the future.

Using an installment plan with future dated agreements

Using an installment plan with an FDA, you can set up payments for purchases to occur at any time between the agreement sign date and the agreement end date. This includes payments before and after the agreement start date.

The seller of record chooses private offer payment dates and amounts. For more details about setting up an installment plan, see [the section called "Creating a payment schedule"](#).

Receiving notifications for future dated agreements

You receive [email notifications](#) to your designated root account for the following actions taken on your future dated agreements:

- Offer acceptance/agreement creation (agreement sign date)
- Upon license or entitlement activation (agreement start date)
- Reminders for agreements expiring 30, 60, or 90 days in advance
- Agreement expiration (agreement end date)

- Upon an agreement amendment or replacement

Note

All existing Amazon Simple Notification Service (Amazon SNS) notifications for SaaS also work for FDA. For FDAs, both Amazon SNS topics are initiated on the agreement start date (and not agreement sign date). For more information, see [the section called "Amazon SNS notifications for SaaS products"](#).

Using future dated agreements with reselling for Channel Partner private offers

Manufacturers and resellers can use future dated agreements for AWS Marketplace Channel Partner private offers.

As the manufacturer:

- Similar to standard AWS Marketplace Channel Partner Private Offers (CPPOs), manufacturers must authorize AWS Marketplace Channel Partners to create CPPOs with a future start date by extending a resale authorization to them.

To learn how to create a resale authorization, follow the steps on the [the section called "Creating a resell opportunity as an ISV"](#) page.

- When creating a resale authorization, manufacturers can optionally choose to specify a maximum allowed service start date. This will be the maximum service start date the AWS Marketplace Channel Partner can specify when creating the corresponding AWS Marketplace Channel Partner private offer.

Note

If the manufacturer doesn't specify a maximum date, the AWS Marketplace Channel Partner can specify any future service date up to 3 years in the future.

As the reseller:

- For resellers and Channel Partners, the steps for creating a future dated Channel Partner private offer and an ordinary future dated private offer are the same, with one key difference. The agreement start date resellers can specify must be earlier than what is specified as the maximum allowed service start date in the resale authorization by the manufacturer.
- To learn how to create a Channel Partner private offer, see [the section called "Channel partner private offers"](#).

AMI-based products

One way of delivering your products to buyers is with [Amazon Machine Images \(AMIs\)](#). An AMI provides the information required to launch an Amazon Elastic Compute Cloud (Amazon EC2) instance. You create a custom AMI for your product, and buyers can use it to create EC2 instances with your product already installed and ready to use.

When buyers use the AMI that you provide, they're billed for instances that they create, following the pricing and metering options that you create for your product. Buyers can use your product AMI in the same way that they use other AMIs in AWS, including making new custom versions of the AMI. EC2 instances created from the AMI are still billed as your product, based on the AMI product code.

See the following resources:

- For more information about pricing AWS Marketplace products, see [Product pricing](#).
- For more information about creating custom metering for your product, see [Custom metering for AMI products with AWS Marketplace Metering Service](#).

AMI-based product delivery methods

Important

AWS Marketplace will discontinue the Private Image Build delivery method in April 2024. The delivery method is only available to existing subscribers until it's discontinued. For more information, see [Private image build](#) in the *AWS Marketplace Buyer Guide*.

You can deliver your AMI-based product in one of three ways:

- **Single AMI** – Buyers select and use the AMI as a template for an EC2 instance. Buyers can find these products using the **Amazon Machine Image** delivery method filter.

For more information, see [Single-AMI products](#).

- **AWS CloudFormation templates** – You create templates that allow buyers to install a system of multiple instances with different roles as a single unit. Buyers can find these products using the **CloudFormation** delivery method filter.

For more information, see [AMI-based delivery using AWS CloudFormation](#).

- **Private image build** – This approach allows buyers to install your product on a base gold image that meets their internal needs for operating system configuration. They create a new AMI, with your product code for tracking and billing. Buyers can find these products using the **Private Amazon Machine Image** delivery method filter.

For more information, see [Private images](#).

See the following resources:

- For more information about how your AMIs are tracked as buyers use them, see [AMI product codes](#).
- For more information about the details of AMI-based products, and their lifecycle, see [Understanding AMI-based products](#).

Understanding AMI-based products

This section outlines key concepts in working with AMI-based products.

Topics

- [Product lifecycle](#)
- [AMI product codes](#)
- [Change requests](#)
- [Product Load Forms](#)

Product lifecycle

AMI-based products include a set of one or more versions of the software, and metadata about the product as a whole. When you create the product, you configure its properties in AWS Marketplace including your product's name, description, and pricing. You also determine the appropriate categories for your product and add keywords so your product appears in relevant searches.

You also create the first version of the software. Depending on how you are delivering your software, this might be a single AMI, a set of one or more AMIs with AWS CloudFormation

templates, or software packages for your buyer to use in creating their own AMIs. For more information, see [AMI-based product delivery methods](#).

For paid products, buyers are billed for the number of installed instances. To meter on a different dimension that your software tracks (for example, number of users of the product), integrate your product with the AWS Marketplace Metering Service. For more information, see [Custom metering for AMI products with AWS Marketplace Metering Service](#).

When you create your product and the first version of your software, it's initially published in a limited scope so that only your account can access it. When you're ready, you can publish it to the AWS Marketplace catalog to allow buyers to subscribe and purchase your product.


On the [Server product](#) page, you can view the list of your products. Depending on what stage it is at, the product will have one of the following statuses:

- **Staging** – An incomplete product for which you are still adding information. At the first **Save and exit** from the create self-service experience, the successful change request creates an unpublished product with information from the full steps you submitted. From this state, you can continue adding information to the product or change already submitted details through change requests.
- **Limited** – A product is complete after it is submitted to the system and passes all validation in the system. Then the product is released to a **Limited** state. At this point, the product has a detail page that is only accessible to your account and whoever you have allowlisted. You can test your product through the detail page. If necessary, for more information and help, contact the [AWS Marketplace Seller Operations team](#).
- **Public** – When you're ready to publish the product so that buyers can view and subscribe to the product, you use the **Update visibility** change request. This initiates a workflow for the AWS Marketplace Seller Operations team to review and audit your product against our [policies](#). Once the product is approved and the change request is processed, the product is moved from a status of **Limited** to **Public**.
- **Restricted** – If you want to stop new users from subscribing to your product, you can restrict the product by using the **Update visibility** change request. A **Restricted** status means that existing users can continue to use the product. However, the product will no longer be visible to the public or be available to new users.

The lifecycle of an AMI-based product for AWS Marketplace does not end after you publish the first version. You should keep your product up to date with new versions of your software and with security patches for the base operating system.

As an example of a complete AMI-based product lifecycle, imagine a seller wants to sell their AMI-based product on AWS Marketplace. Following is how the seller creates and maintains the product over time:

1. **Create a product** – The seller creates the product, and publishes version 1.0.0 to AWS Marketplace. Buyers can create instances of version 1.0.0 and use it.
2. **Add a new version** – Later, the seller adds a new feature to the product, and adds a new version, 1.1.0, that includes the feature. Buyers can still use the original version, 1.0.0, or they can choose the new version, 1.1.0.

 **Note**

Unlike new products, new versions are published to full public availability. You can only test them in AWS Marketplace without customers seeing them if the product as a whole is in limited release.

3. **Update product information** – With version 1.1.0 available, the seller lets buyers know about the new feature by updating the product information with new highlight text describing the feature.
4. **Add a minor version** – When the seller fixes a bug in version 1.1.0, they release it by adding a new version 1.1.1. Buyers now have the choice of using version 1.0.0, 1.1.0, or 1.1.1.
5. **Restrict a version** – The seller decides that the bug is serious enough that they don't want buyers to be able to use version 1.1.0, so they restrict that version. No new customers can then buy 1.1.0 (they can only choose 1.0.0 or 1.1.1), although existing buyers still have access to it.
6. **Update version information** – To help those existing buyers, the seller updates the version information for 1.1.0 with a suggestion to upgrade to version 1.1.1.
7. **Monitor usage** – As buyers purchase and use the product, the seller monitors sales, usage, and other metrics using the AWS Marketplace [Seller reports, data feeds, and dashboards](#).
8. **Remove the product** – When the product is no longer needed, the seller removes it from AWS Marketplace.

In this example, the seller created three different versions of the AMI in the product, but only two were available to new buyers (prior to removing the product).

To make modifications to versions or the product information, you create [Change requests](#) in the AWS Marketplace Management Portal.

For detailed instructions on the steps to create and manage your AMI-based product, see [Single-AMI products](#).

AMI product codes

A unique product code is assigned to your product when you create it in AWS Marketplace. That product code is associated with the AMIs for your product and is used to track usage of your product. Product codes are propagated automatically as buyers work with the software. For example, a customer subscribes and launches an AMI, configures it, and produces a new AMI. The new AMI still contains the original product code, so correct usage tracking and permissions remains in place.

Note

The product *code* is different than the product *ID* for your product. Each product in AWS Marketplace is assigned a unique product ID. The product ID is used to identify your product in the AWS Marketplace catalog, in customer billing, and in seller reports. The product code is attached to instances created from your AMI as instance metadata. When an AMI with that product code is used to create an instance, the customer will get a bill that shows the associated product ID. After you create your product, find the product code and the product ID in the AWS Marketplace Management Portal page for your product.

As a seller, your software can get the product code for the running Amazon Elastic Compute Cloud (Amazon EC2) instance at runtime from the instance metadata. You can use the product code for extra security, such as validating the product code at product start. You can't make API calls to an AMI's product code until the product has been published into a limited state for testing. For more information about verifying the product code, see [Verifying your software is running on your AWS Marketplace AMI](#).

Change requests

To make changes to a product or version in AWS Marketplace, you submit a **change request** through the AWS Marketplace Management Portal. Change requests are added to a queue and can take from minutes to days to resolve, depending on the type of request. You can see the status of requests in the AWS Marketplace Management Portal.

Note

In addition to the AWS Marketplace Management Portal, you can also create change requests by using the [AWS Marketplace Catalog API](#).

The types of changes you can request for AMI-based products include:

- Update product information displayed to buyers.
- Update version information displayed to buyers.
- Add a new version of your product.
- Restrict a version so that new buyers can no longer access that version.
- Update the AWS Regions that a product is available in.
- Update the pricing and instance types for a product.
- Remove a product from AWS Marketplace.

For more information, see [Create a change request](#).

Note

Some change requests require you to use product load forms to create the request. For more information, see [Product Load Forms](#).

Update change request

Change requests that start with an update will load the current details. Then you can make updates, which overwrite the existing details.

Add or restrict change request

Add and restrict request pairs are specifically for steps and updates that are provisioned after each request succeeds. A request succeeds after you choose **Save and exit** and **Submit** actions in the self-service experience.

For example, if the AMI asset is provisioned to the instances and Regions once added, then they can only be restricted rather than completely removed. This means existing subscribers and users can continue to use the product until their subscription or contract runs out. However, no new subscribers can be added to a product that is in a **Restricted** status.

Product Load Forms

Typically, when you create or edit your product, you work within the AWS Marketplace Management Portal user interface to make the changes that you want. However, a few operations direct you to use a *Product Load Form* (PLF).

A PLF is a spreadsheet that contains all the information about a product. There are several ways that you can get the PLF:

- You can download the PLF for an existing product from the product's details page in the AWS Marketplace Management Portal.
- You are prompted to download the PLF when you select a menu item for an action that requires it. For example, if you choose to create a new monthly billed server product, you will be prompted to download the appropriate PLF.

If the action is an edit to an existing product, the PLF is pre-populated with the information for that product, so you only need to change the details that you are updating.

- If you need a new, blank PLF, there are links to PLFs, based on the type of product you want to create, on the AWS Marketplace Management Portal [File upload](#) page.

After you have completed your PLF, upload it to the AWS Marketplace Management Portal [File upload](#) page. The PLF itself has more detailed instructions in the **Instructions** tab.

Single-AMI products

This section discusses how you can work with product listings in AWS Marketplace for single-AMI products. Customers can use Amazon Machine Images (AMIs) to create Amazon EC2 instances with your product already installed and configured.

The AMI self-service experience guides you as you create your product listing and make change requests. By using the self-service experience, you can update your product listing directly with less time needed for processing by the AWS Marketplace Seller Operations team. Many steps of the self-service experience align with the catalog system in AWS Marketplace, which facilitates direct validation instead of waiting for processing and validation from the AWS Marketplace Seller Operations team.

Note

For a few tasks, you can choose from two procedures: A procedure that is appropriate only for the self-service experience and an older procedure that isn't relevant to the self-service experience. You can use either procedure during the current transition period. The older procedure displays a banner with information about when it will no longer be available.

Topics

- [Prerequisites](#)
- [Understand the self-service experience](#)
- [Create your single-AMI product](#)
- [Create a change request](#)
- [Get the status of a change request](#)
- [Update product information](#)
- [Update the allowlist \(preview accounts\)](#)
- [Update product visibility](#)
- [Add an AWS Region](#)
- [Restrict an AWS Region](#)
- [Update support for future AWS Regions](#)
- [Add an instance](#)

- [Restrict an instance](#)
- [Update version information](#)
- [Add a new version](#)
- [Restrict a version](#)
- [Update pricing](#)
- [Update availability by country](#)
- [Update your EULA](#)
- [Update the refund policy](#)
- [Give AWS Marketplace access to your AMI](#)
- [Remove a product from AWS Marketplace](#)
- [Troubleshooting common errors when submitting change requests](#)

Prerequisites

Before you create an AMI product listing, you must complete the following prerequisites:

1. Have access to the AWS Marketplace Management Portal. This is the tool that you use to register as a seller and manage the products that you sell on AWS Marketplace. To learn more about getting access to the AWS Marketplace Management Portal, see [Policies and permissions for AWS Marketplace sellers](#).
2. Register as a seller and, if you want to charge for your products, submit your tax and banking information. To learn more about becoming a seller, see [Getting started as a seller](#).
3. Have a product that you want to sell. For AMI-based products, this typically means you have created or modified your server software, and have created an AMI for your customers to use. To learn more about preparing an AMI for use in AWS Marketplace, see [Best practices for building AMIs](#).

Understand the self-service experience

The self-service experience guides you through creating your product on AWS Marketplace. As you proceed through the steps, you specify product information and AMI deployment settings, such as AWS Region, instance types, and AMI details. You also configure transaction details including the pricing, country availability, EULA, and refund policy. As an option, you can specify an allowlist of AWS account IDs to access and test the product while it is in the **Limited** status.

Before you get started, review the following key aspects of the self-service experience:

- You can only go to the next step after you complete the required fields in the current step. This requirement is because there is page-level validation at the end of each step. You can't save or submit an incomplete step.
- If you need to end your session before completing all the steps in the process, you can choose **Save and exit** to submit the steps that you completed to the staging area.
- A step that is incomplete and doesn't pass validation isn't submitted to the system. A partially completed step isn't valid and can't be saved.
- When you choose **Save and exit**, the **Save and exit** dialog box shows the steps that passed validation checks. You can review and choose to save up to the last completed and validated steps. If there is a validation error or missing details, you can choose **Fix it** to go back to that step.
- After you **Save and exit**, the request is under review while it's processing. It could take a few minutes or hours to finish processing. You can't continue the steps or make changes until the request has succeeded. For the first **Save and exit**, the request is creating the product in parallel with the steps that you have completed.
 - After the request has **Succeeded**, you have completed the save. To resume changes on the **Product overview** page, choose **Resume product creation** or use **Request changes** to update the details you have previously submitted in the last session. When you resume, notice that the steps you have completed are marked with a green **Succeeded** label. To update a previously submitted step, use **Request changes**. The previous **Save and exit** request must be completed before you can continue this step.
- When you've completed all the steps, you can choose **Next** to see a review. Choose **Submit** which requests that the system perform a final validation. After you receive a **Succeeded** response, the product moves to a **Limited** status. You can see on the detail page that the product is now available to anyone on the allowlist. If the request fails, the product remains in the **Staging** status and requires you to make corrections before resubmitting.

Create your single-AMI product

This section provides the procedures that you can use to create a listing for a single-AMI product in AWS Marketplace, including the option to use the self-service experience.

Topics

- [Create a single-AMI product by using self-service](#)

- [Create a single-AMI product](#)
- [Additional resources](#)

Create a single-AMI product by using self-service

You can use the following process that guides you through creating your single-AMI product in AWS Marketplace.

Note

You can only go to the next step when you complete the required fields in the current step. You can't save or submit an incomplete step. If you need to end your session before completing all the steps in the process, use the **Save and exit** function to submit the steps that you completed to the staging area. For more information, see [Understand the self-service experience](#).

To create a single-AMI product using self-service

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and then sign in to your seller account.
2. From the **Products** menu, choose **Server**. Or, you can go directly to the [Server Products](#) page.
3. From the **Server products** tab, select **Create server product**, select **Amazon Machine Image (AMI)**, and then select one of the licensing types for single-AMI products:
 - **Bring your own license (BYOL)** – A product that the user gets a license from you outside of AWS Marketplace. It can be either a paid or free license.
 - **Free** – A product that is free for your subscribers to use. (They will still pay charges for any associated Amazon Elastic Compute Cloud (Amazon EC2) instance, or other AWS resources.)
 - **Paid hourly or hourly-annual** – A product that the buyer pays for either on an hourly basis or hourly with an annual contract. AWS does the metering based on the product code on the AMI.
 - **Paid monthly** – A product that the buyer is billed for monthly by AWS.
 - **Paid usage** – Software is directly charged for the value you provide, along with one of four usage categories: users, data, bandwidth, or hosts. You can define up to 24 dimensions for the product. All charges are still incurred by the customer.

- **AMI with contract pricing** – A Single-AMI product or Single-AMI with an AWS CloudFormation stack that the buyer pays an upfront fee for.
4. The self-service experience guides you through the steps to create an AWS Marketplace listing. You must enter product information (metadata), product deployment details (AWS Region, instances, and AMI details), and public offer details (price, EULA, availability by country, EULA, refund). As an option, you can add accounts to the allowlist to test the product. Complete each step to move to the next step in the process.


 **Note**

If you need to end your session before completing all steps in the process, you can use the **Save and exit** function to submit the steps you completed to the staging area. This creates a request for the information you provided to be validated. While the request is processing, you can't edit the product. After the request succeeds, you can continue creating your product by choosing **Resume product creation**.

A failed request means no update was made to the product because of a validation error. This will be visible on the request log for your product. You can select the request to view the error, use **Copy to new** under **Actions** to correct the error, and resubmit the request. When you resume the steps, you can continue from the step after the step that you saved in the last session. To update previous steps, go to the product overview page and submit a [Change request](#) to update steps that you submitted previously.

5. After entering required information for all of the change request steps, choose **Submit**. This submittal creates a request to the AWS Marketplace catalog system to validate the information and release the product to a **Limited** state, if the validation passes. While the request is processing, you can't continue to edit the product. After the request succeeds, the product is moved to a **Limited** state.
 - When your product is initially published, it's only accessible to your AWS account (the one you used to create the product) and the AWS Marketplace Seller Operations team's test account. If you view the product from the **Server products** page, you can choose **View on AWS Marketplace** to view the product details as they will appear in AWS Marketplace for buyers. This detail listing isn't visible to other AWS Marketplace users.
 - This capability allows you to test your product (and even publish multiple versions for testing) before releasing it publicly.

6. Test your product in the **Limited** state and make sure that it follows AWS Marketplace [AMI-based product requirements](#) and the [product checklist](#). Then, to request that your product be published to **Public**, choose **Update visibility**. The AWS Marketplace Seller Operations team must review your product before approving it to go **Public**.

 **Note**

Product verification and publication is a manual process, which is handled by the AWS Marketplace Seller Operations team. It can take 7–10 business days to publish your initial product version, if there are no errors. For more details about timing, see [Timing and expectations](#).

For more information about preparing and submitting both your single-AMI product and your product information, see [Additional resources](#).

Create a single-AMI product


 **Important**

On July 14, 2023, AWS Marketplace will discontinue the following procedure. After July 14, 2023, use the [the section called “Create a single-AMI product by using self-service”](#) procedure.

To create a single-AMI product (Legacy)

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and then sign in to your seller account.
2. From the **Products** menu, choose **Server**. Or, you can go directly to the [Server products](#) page.
3. From the **Server products** tab, select **Create server product**, select **Amazon Machine Image (AMI) (Legacy experience)**, and then select one of the licensing types for single-AMI products:
 - **Bring your own license (BYOL)** – A product that the user gets a license from you outside of AWS Marketplace. It can be either a paid or free license.
 - **Free** – A product that is free for your subscribers to use. (They will still pay charges for any associated Amazon Elastic Compute Cloud (Amazon EC2) instance or other AWS resources.)

- **Paid hourly or hourly-annual** – A product that the buyer pays for either on an hourly basis or hourly with an annual contract. AWS does the metering based on the product code on the AMI.
- **Paid monthly** – A product that the buyer is billed for monthly by AWS. If you select **Paid monthly**, you are asked to download a Product Load Form (PLF).
- **Paid usage** – Software is directly charged for the value you provide, along with one of four usage categories: users, data, bandwidth, or hosts. You can define up to 24 dimensions for the product. All charges are still incurred by the customer.
- **AMI with contract pricing** – A Single-AMI product or Single-AMI with an AWS CloudFormation stack that the buyer pays an upfront fee for.

 **Note**

There are additional types of non self-service based pricing models for AMI-based products. These pricing types apply when your product integrates with the AWS Marketplace Metering Service to provide custom metering based on your customers' usage. To create a product that has usage-based pricing, you must download, complete, and upload a Product Load Form (PLF). For more information, see [Custom metering pricing for AMI products](#)

For more information about PLFs, see [Product Load Forms](#).

For more information about the different types of licensing, see [AMI pricing models](#).

4. Based on your selection, fill out the information for the new product, and choose **Submit**.
5. Verify that the request appears on the **Requests** tab with the **Under review** status. You can return to this page to see the status of your request as it is processed.

 **Note**

Product verification and publication is a manual process, handled by the AWS Marketplace Seller Operations team. It can take 7–10 business days to publish your initial product version, if there are no errors. For more details about timing, see [Timing and expectations](#).

When your product is initially published, it's only accessible to your AWS account (the one you used to create the product). If you view the product from the **Server products** page, you can select **View**

on AWS Marketplace to view the product details as it will appear in AWS Marketplace for buyers. This detail listing isn't visible to other AWS Marketplace users.

This capability allows you to test your product (and even publish multiple versions for testing) before releasing it publicly. If you need to make the product available to additional test accounts, or to publish your product publicly, contact the [AWS Marketplace Seller Operations](#) team.

For more information about preparing and submitting both your single-AMI product and your product information, see [Additional resources](#).

Additional resources

For more information about preparing your product information and submitting it for publication, see the following resources:

- [Preparing your product](#)
- [Submitting your product for publication](#)

For more information about preparing your single-AMI product for submission to AWS Marketplace, see the following resources:

- [Best practices for building AMIs](#)
- [AMI product checklist](#)
- [AMI-based product requirements](#)

Create a change request

This section provides the procedures that you can use to create a change request for a single-AMI product in AWS Marketplace, including the option to use the self-service experience. You create a change request for the following situations:

- You saved your in-progress steps, but didn't complete the entire process, while using the self-service experience to create a single-AMI product listing. To complete the remaining steps, you create a change request.
- You want to make modifications to the product information for your product that is in either a **Limited** or **Public** state. To update the information, you create a change request. For more information about the types of changes that you can request for AMI-based products, see [Change requests](#).

Note

In addition to the AWS Marketplace Management Portal, you can also create change requests by using the [AWS Marketplace Catalog API](#).

Topics

- [Create a change request](#)
- [Create a change request by using self-service](#)
- [Additional resources](#)

Create a change request**⚠ Important**

On June 15, 2023, AWS Marketplace will discontinue the following procedure. After June 15, 2023, use the [the section called “Create a change request by using self-service”](#) procedure.

To make modifications to versions or the product information, you create a *change request* in the AWS Marketplace Management Portal.

To create a change request

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and sign in to your seller account, then go to the [Server products](#) page.
2. On the **Server products** tab, select the product that you want to modify.
3. Choose an option from the **Request changes** dropdown list.

For most change requests, you simply fill out the form in the user interface and submit it. However, for certain changes, you must download, complete, and then upload a Product Load Form (PLF). This is a spreadsheet that contains a form for you to fill out with the required information. When you choose one of these change requests, you are prompted to download the correct PLF for the request you are attempting to create. The PLF is pre-populated with information from your existing product details. You can upload your completed PLF to the AWS Marketplace Management Portal [File upload](#) page.

Note

We strongly recommend that you download and use the most recent PLF. The form is regularly updated with new information, including instance types and AWS Regions as they become available. You can find the latest PLF for a product from the **Server products** page, by selecting the product and then choosing **Download Product Load Form**.

For more information about the status of a change request, see [the section called “Get the status of a change request”](#). For insight into potential issues with change requests, see [Troubleshooting common errors when submitting change requests](#).

Create a change request by using self-service

To make modifications to versions or the product information, you create a *change request* in the AWS Marketplace Management Portal. Change requests are the building blocks of a self-service listing that you use to make changes to your product. Each time you select **Save and exit** from the steps or select **Submit** for any update, you are making a change request. You can find your requests on the AWS Marketplace Management Portal [Request](#) tab.

To create a change request using self-service

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and sign in to your seller account, then go to the [Server products](#) page.
2. On the **Server products** tab, select the product that you want to modify.
3. Choose an option from the **Request changes** dropdown.
4. After you make a change request, there is a wait time for the system to process your request, reflected **Under Review**. When the request completes, it will result in either **Succeeded** or **Failed**.
 - After the request is submitted, it begins processing through these statuses: **Under review**, **Preparing changes**, and **Applying changes**.
 - **Succeeded** means that the requested change has been processed and changes reflect in the system.
 - **Failed** means that something went wrong with the request, so the changes were not processed. If the status is **Failed**, you can select the request to find error codes that provide

recommendations on how to correct the error. At this point, you can troubleshoot the errors and create a new request for the change. To make the process faster, you can choose **Copy to new request** to copy the details of the failed request. Then, you can make the adjustment and resubmit the request.

Additional resources

For more details about change requests for specific types of updates, see the following resources:

- [Update product information](#)
- [Update version information](#)
- [Add a new version](#)
- [Restrict a version](#)

Get the status of a change request

Important

On June 15, 2023, AWS Marketplace will discontinue the following procedure. This procedure is no longer needed for the self-service experience.

After you submit a change request, you can see the status of your request from the **Requests** tab of the [Server products](#) page of the AWS Marketplace Management Portal. The status could be any of the following:

- **Under review** means that your request is being reviewed. Some requests require manual review by the AWS Marketplace team but most are reviewed automatically in the system.
- **Succeeded** means that your request is complete. Your product or version has been updated as you requested.
- **Action required** means that you need to update your request to fix an issue or answer a question about the request. Select the request to see the details, including any issues.
- **Failed** means that something went wrong with the request, and you should create a new request for the change, with the same data.

Update product information

After you have created your product, you might want to change some of the information associated with it in AWS Marketplace. For example, if a new version modifies the description or highlights of the product, you can edit the product information with the new data.

To update product information

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and then sign in to your seller account.
2. Go to the [Server products](#) page, and on the **Server products** tab, select the product that you want to modify.
3. From the **Request changes** dropdown, choose **Update product information**.
4. Update any of the following fields that you need to change:
 - **Product title**
 - **SKU**
 - **Short description**
 - **Long description**
 - **Product logo image URL**
 - **Highlights**
 - **Product categories**
 - **Keywords**
 - **Product video URL**
 - **Resources**
 - **Support information**

Note

For details about the logo format, see [Company and product logo requirements](#).

5. Select **Submit**.
6. Verify that the request appears on the **Requests** tab with the **Under review** status. You might need to refresh the page to see the request on the list.

You can check the status of your request at any time from the **Requests** tab of the [Server Products](#) page. For more information, see [Get the status of a change request](#).

Update the allowlist (preview accounts)

To change the list of AWS account IDs that can see your product in a **Limited** state, use **Update allowlist**.

To update the allowlist

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and then sign in to your seller account.
2. Go to the [Server products](#) page, and on the **Current server product** tab, select the product that you want to modify.
3. From the **Request changes** dropdown, choose **Update allowlist**. The current list is provided with the list of accounts that are currently allowlisted.
4. Add the AWS account IDs that are preferred for visibility and separate the IDs with commas.
5. Choose **Submit change request** to submit your request for review.
6. Verify that the **Requests** tab shows the **Request status** as **Under review**. When the request completes, the status becomes **Succeeded**.

Update product visibility

To change which buyers can view your product in AWS Marketplace, you can use **Update visibility**.

To update visibility

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and then sign in to your seller account.
2. Go to the [Server products](#) page, on the **Current server product** tab, select the product that you want to modify.
3. From the **Request changes** dropdown, choose **Update visibility**.

Note

You can request that the product be moved from a **Limited** status to a **Public** status by using this change request. However, the change request must go through an AWS Marketplace Seller Operations team approval process to be moved to **Public**.

4. Choose **Submit change request** to submit your request for review.
5. Verify that the **Requests** tab shows the **Request status** as **Under review**. When the request completes, the status becomes **Succeeded**.

Add an AWS Region

You can add an Region where buyers can use your product.

To add a Region

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and then sign in to your seller account.
2. Go to the **Server products** page, and on the **Current server product** tab, select the product that you want to modify.
3. From the **Request changes** dropdown, choose **Add Region**.
4. Select the Region that you want to add from the list of available Regions.
5. Choose **Submit request** to submit your request for review.
6. Verify that the **Requests** tab shows the **Request status** as **Under review**. When the request completes, the status becomes **Succeeded**.

Restrict an AWS Region

To prevent new buyers from using your product in a specific AWS Region, you can restrict the Region. You can add the Region back at a later time. Existing subscribers of the product in the Region can continue using the product from the Region as long as they're subscribed.

To restrict a Region

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and then sign in to your seller account.

2. Go to the [Server products](#) page, and on the **Current server product** tab, select the product that you want to modify.
3. From the **Request changes** dropdown, choose **Restrict Region**.
4. Select the dropdown menu to view the list of Regions in which your product is currently available.
5. Select the Regions that you want to restrict.
6. The Regions you have selected appear as tokens. Review the list of Regions that you're restricting, and enter X for Regions that you don't want to restrict.
7. Choose **Submit change request** to submit your request for review.
8. Verify that the **Requests** tab shows the **Request status** as **Under review**. When the request completes, the status becomes **Succeeded**.

If your request is successful, your existing users receive the following email message notifying them of the Region to be restricted. They can continue using your product as long as they remain subscribed, but they can't re-subscribe if they cancel the subscription.

Greetings from AWS Marketplace,

This message is a notification detailing a recent change for <ProductName>. {{{sellerName}}} has opted to restrict the <ProductType> product in <Restricted Region(s)> beginning <DateOfChange>.

This impacts you in the following ways:

1. As long as you're subscribed to the product, you can continue using the software product in the restricted Region.
2. You can't begin new instances of the software product in the restricted Region.
3. You can continue using the software product in all available AWS Regions.

Regards,
The AWS Marketplace Team

Amazon Web Services, Inc. is a subsidiary of Amazon.com, Inc. Amazon.com (<http://amazon.com/>) is a registered trademark of Amazon.com, Inc. This message was produced and distributed by Amazon Web Services Inc., 410 Terry Ave. North, Seattle, WA 98109-5210.

Update support for future AWS Regions

If you want your product to be onboarded to newly launched AWS Regions, you can use **Update future Region support**.

To update future Region support

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and then sign in to your seller account.
2. Go to the [Server products](#) page, and on the **Current server product** tab, select the product that you want to modify.
3. From the **Request changes** dropdown, choose **Update future Region support**.
4. You can choose to activate future Region support to allow AWS Marketplace to onboard your product to newly launched AWS Regions on your behalf.
5. After activating the feature, you can choose between all future Regions or limit to US Regions only.
6. Choose **Submit change request** to submit your request for review.
7. Verify that the **Requests** tab shows the **Request status** as **Under review**. When the request completes, the status becomes **Succeeded**.

Add an instance

You can add a new instance for which buyers can use a single-AMI.

To add an instance

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and then sign in to your seller account.
2. Go to the [Server products](#) page, and on the **Current server product** tab, select the product that you want to modify.
3. From the **Request changes** dropdown, choose **Add instance**.
4. Select the instances that you want to add from the list of available instances. .
5. Choose **Submit request** to submit your request for review.
6. Verify that the **Requests** tab shows the **Request status** as **Under review**. When the request completes, the status becomes **Succeeded**.

Note

If your current pricing model is not free or uses a Bring Your Own License (BYOL) model, there is an additional step to add prices.

If you created an **Add instance** with a price for the new instance or **Update pricing** to increase price, you can't use self-service to **Add instance** in the 90 days starting from the day you made the change. To make these changes, contact the [AWS Marketplace Seller Operations team](#).

Restrict an instance

If you want to prevent new buyers from using your single AMI product from a specific instance, you can restrict the instance. You can add the instance back at a later time, if needed. Existing users of the single AMI on the restricted instance can continue to use the product from the Region as long as they're subscribed.

To restrict an instance

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and then sign in to your seller account.
2. Go to the [Server products](#) page, and on the **Current server product** tab, select the product that you want to modify.
3. From the **Request changes** dropdown, choose **Restrict instance**.
4. Select the instances that you want to restrict, and choose **Restrict**.
5. Choose **Submit change request** to submit your request for review.
6. Verify that the **Requests** tab shows the **Request status** as **Under review**. When the request completes, the status becomes **Succeeded**.

Note

If the check box is shaded, this means the instance is associated with one to several versions as a recommended instance type. To restrict such instances, use **Update versions** to choose a different recommended instance type. After the change requests

complete and the instance you want to restrict is no longer a recommended instance type, you can return to **Restrict instance** to restrict your chosen instance.

Update version information

After a version is created, it can be helpful to provide updated information to your buyers by modifying the information associated with the version. For example, if you plan to restrict version 1.0 after version 1.1 is released, you can update the description of version 1.0 to direct buyers to version 1.1, with the date that the version will be restricted. You update the version information from the AWS Marketplace Management Portal.

To update version information

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and then sign in to your seller account.
2. Go to the [Current server product](#) page, on the **Server products** tab, then select the product that you want to modify.
3. From the **Request changes** dropdown, choose **Update version information**.
4. On the **Update version** page, select the version that you want to update.
5. Update any of the following information that you need to modify:
 - **Release notes**
 - **Usage instructions**
 - **64-bit (x86) Amazon Machine Image (AMI)** – Details on usage and security group
6. Select **Submit**.
7. Verify that the request appears on the **Requests** tab with the **Under review** status.

Note

You can't use this procedure to update the version title, or the AMI associated with the version. Instead, [create a new version](#) and [restrict the previous version](#).

You can check the status of your request at any time from the **Requests** tab of the [Server Products](#) page. For more information, see [Get the status of a change request](#).

Add a new version

You can add a new version of your product when you make changes to the product, the base image, or any other time you need to modify the AMI for the product. Add a new version of your product from the AWS Marketplace Management Portal.

Note

For information about creating an AMI for AWS Marketplace, see [Best practices for building AMIs](#).

To add a new version

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and then sign in to your seller account.
2. Go to the [Server products](#) page, on the **Current server product** tab, then select the product that you want to modify.
3. From the **Request changes** dropdown, choose **Add new version**. The **Add a new version** form appears, pre-populated with the information from your most recent version.
4. In the **Version information** section, provide the following information:
 - **Version title** – Enter a valid string (for example *1.1* or *Version 2.0*). It must be unique across the product.
 - **Release notes** – Enter text to describe details about this version.
5. In the **New Amazon Machine Image (AMI)** section, provide the following information:
 - **Amazon Machine Image ID** – Enter the AMI ID for the AMI that you want to use for this version. You can find the AMI ID from the [list of AMIs in the console](#). The AMI must exist in the US East (N. Virginia) Region, and in your AWS Marketplace Seller account. The snapshot associated with this AMI can't be encrypted.
 - **IAM access role ARN** – Enter the Amazon Resource Name (ARN) for an AWS Identity and Access Management (IAM) role that allows AWS Marketplace to gain access to your AMI. For instructions on how to create the IAM role, see [Give AWS Marketplace access to your AMI](#). Use the standard format for an IAM ARN, for example: *arn:aws:iam::123456789012:role/RoleName*. The ARN must exist in your AWS Marketplace Seller account.

- **OS user name** – For Linux-based AMIs, enter the name of a user that can be used to sign into the instance. We recommend using *ec2-user*.
 - **Scanning port** – Enter the port number that can be used to log into the operating system: the SSH port for a Linux AMI or the RDP port for a Windows AMI.
6. If it is not already, expand the **Configuration settings to publish the AMI to the AWS Marketplace customer website** section, then provide the following information:
- **Usage instructions** – Enter instructions for using the AMI or a link to more information about using the AMI. For example: *To get started with the product, navigate to <https://example.com/usage.htm>.*
 - **Endpoint URL** – Provide information about how the buyer can access the software after they create an instance. Enter the **Protocol** (*https* or *http*), the **Relative URL** (for example, */index.html*), and the **Port** (for example, *443*) that buyers can use to access your product. (The host name depends on the EC2 instance, so you only need to provide the relative path).
 - **Operating system (OS)** – Enter the name of the OS used by the AMI (for example, *Amazon Linux*).
 - **OS version** – Enter the specific version of the OS in the AMI.
 - **Recommended instance type** – Choose the instance type that buyers get by default.
 - **Security group recommendations** – Enter the information for one or more recommendations, including the protocol (*TCP* or *UDP*), range of ports to allow, and list of IPv4 CIDR IPs (in the form *xxx.xxx.xxx.xxx/nn*, for example, *192.0.2.0/24*).
7. Select **Submit** to submit the request to add your new version.
8. Verify that the request appears on the **Requests** tab with the **Under review** status. If there are errors to fix, the page displays the errors in a table at the top of the page, and the specific fields that need to be updated display in red.

You can check the status of your request at any time from the **Requests** tab of the [Server products](#) page. The new version will be reviewed and, if successful, published as a new public version of your product. If there is an issue, the status might be **Action required**. Select the request to see details, including any issues.

If your request is successful, your existing users receive the following email message. The message notifies them that the new version is available, links to the version's release notes, and suggests that they upgrade to the latest version. As the AWS account root user, you also receive a copy of the email message in the email account that's associated with your AWS account.

Greetings from AWS Marketplace,

Thank you for subscribing to <product-title>

We are writing to inform you that <seller-name> has added a new version to <product-title> on AWS Marketplace.

As an existing customer, your subscription to the product, any running instances and access to previous versions are unaffected. However, <seller-name> does recommend you to update to the latest version, <product-title>/<version-title> by visiting <product-detail-page-of-new-listing>.

For additional questions or upgrade information, please contact <seller-name> directly. Click here <link of seller page on MP> to visit the seller's profile page on AWS Marketplace.

Release notes for <product-title>/<version-title>:

<release-notes>

Thank you,
The AWS Marketplace Team
<https://aws.amazon.com/marketplace>

Amazon Web Services, Inc. is a subsidiary of Amazon.com, Inc. Amazon.com is a registered trademark of Amazon.com, Inc.

This message was produced and distributed by Amazon Web Services Inc., 410 Terry Ave. North, Seattle, WA 98109-5210

Restrict a version

If you want to prevent buyers from accessing a specific version of your public product, you can restrict that version.

Note

All subscribers can use the current version regardless of the restriction status. AWS Marketplace guidelines require that you continue to offer support to existing buyers for 90 days after restricting the version. Your AMI will be marked as deprecated after the version is restricted. For more information, see [Deprecate an AMI](#) in the *Amazon Elastic Compute Cloud User Guide for Windows Instances*.

To restrict a version

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and then sign in to your seller account.
2. Go to the [Server products](#) page, on the **Current server product** tab, then select the product that you want to modify.
3. From the **Request changes** dropdown, choose **Restrict version**.
4. On the **Restrict version** page, select the version (or versions) that you want to restrict.
5. Select **Submit** to submit your request for review.
6. Verify that the **Requests** tab shows the **Request status** as **Under review**. When the request completes, the status is **Succeeded**.

Note

You can't restrict all versions of a product. If you try to restrict the last remaining public version of a product, you will receive an error. To completely remove a product, see [the section called "Remove a product from AWS Marketplace"](#).

You can check the status of your request at any time from the **Requests** tab of the [Server products](#) page. For more information, see [Get the status of a change request](#).

Note

Restricting a version can take up to 3 days to complete.

If your request is successful, your existing users receive the following email message that notifies them of the version restriction and suggests they use the most recent version available. As the AWS account root user, you also receive a copy of the email message in the email account that's associated with your AWS account.

Greetings from AWS Marketplace,

Thank you for subscribing to <product-title>.

We are writing to inform you that, as of <Version-Restriction-Date>, <Seller Name> will no longer offer version(s) "<version-title>" to new subscribers. Your use and subscription is unaffected for this version(s), however it is recommended that users upgrade to the latest version on AWS Marketplace.

For additional questions or upgrade information, please contact <seller-name> directly. Click here<link of seller page on MP> to visit the seller's profile page on AWS Marketplace.

Thank you,
The AWS Marketplace Team
<https://aws.amazon.com/marketplace>

Amazon Web Services, Inc. is a subsidiary of Amazon.com, Inc. Amazon.com is a registered trademark of Amazon.com, Inc. This message was produced and distributed by Amazon Web Services Inc., 410 Terry Ave. North, Seattle, WA 98109-5210

Update pricing

If you want to change the pricing per instance type on your single-AMI product, then you can update pricing.

Note

If you made an **Add instance** change request with a price for the new instance or **Update pricing** to increase price (not decrease price), you can't use self-service to make pricing updates in the 90-days starting from the day you made the change. For additional support, contact the [AWS Marketplace Seller Operations team](#) to make these changes.

To update product pricing

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and then sign in to your seller account.
2. Go to the [Server products](#) page, on the **Current server product** tab, then select the product that you want to modify.
3. From the **Request changes** dropdown, choose **Update pricing**.
4. Text boxes are pre-populated with the current pricing information. You can delete the current price and then enter your new price. We recommend that you review the prices you're requesting to verify correct pricing.

Note

A pricing increase for any instance will result in **Add Instance** and **Update Pricing** being locked for the next 90-days.

5. Choose **Submit change request** to submit your request for review.
6. Verify that the **Requests** tab shows the **Request status** as **Under review**. When the request completes, the status becomes **Succeeded**.

Note

If you created an **Add instance** with a price for the new instance or **Update pricing** to increase price (not decrease price), you can't use self-service to make pricing updates in the 90 days starting from the day you made the change. For additional support, contact the [AWS Marketplace Seller Operations team](#) to make these changes.

Update availability by country

If you want to change the countries in which your product can be subscribed to and offered, you can use **Update availability**.

To update availability by country

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and then sign in to your seller account.
2. Go to the [Server products](#) page, on the **Current server product** tab, then select the product that you want to modify.
3. From the **Request changes** dropdown, choose **Update availability**.
4. Choose one of the following options:
 1. **All countries** – Available in all supported countries.
 2. **All countries with exclusions** – Available in all supported countries except in selected countries.
 3. **Custom list** – Specific list of countries where the product is available.
5. Choose **Submit change request** to submit your request for review.

6. Verify that the **Requests** tab shows the **Request status** as **Under review**. When the request completes, the status becomes **Succeeded**.

Update your EULA

If you want to change your end user license agreement (EULA), you can use **Update EULA**. This action updates the EULA for new users subscribing to your product and for product renewals.

To update a EULA

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and then sign in to your seller account.
2. Choose the **Server products** tab, on the **Current server product** tab, select the product that you want to modify.
3. From the **Request changes** dropdown, choose **Update end-user license agreement**.
4. You can select the **Standard Contract for AWS Marketplace (SCMP)** or submit your own custom EULA. For a custom EULA, you must provide the URL for your custom contract from an Amazon S3 bucket.

Note

Public accessibility must be enabled on your Amazon S3 bucket.

5. Choose **Submit change request** to submit your request for review.
6. Verify that the **Requests** tab shows the **Request status** as **Under review**. When the request completes, the status becomes **Succeeded**.

Update the refund policy

If you want to change the refund policy for your product, you can use **Update refund policy**.

To update the refund policy

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and then sign in to your seller account.
2. Go to the **Server products** page, on the **Current server product** tab, then select the product that you want to modify.

3. From the **Request changes** dropdown, choose **Update refund policy**.
4. The current refund policy details are provided in the text box for you to edit. Submitting the request overwrites the current refund policy.
5. Choose **Submit change request** to submit your request for review.
6. Verify that the **Requests** tab shows the **Request status** as **Under review**. When the request completes, the status becomes **Succeeded**.

Give AWS Marketplace access to your AMI

When you create a request that includes adding a new AMI to AWS Marketplace, the AMI must be copied into the AWS Marketplace system and then scanned for security issues. You must give AWS Marketplace access to the AMI by creating an AWS Identity and Access Management (IAM) role with permissions to perform actions on your AMI and a trust policy that allows AWS Marketplace to assume the role. You only need to create the IAM role once.

To create a role for AWS Marketplace AMI assets ingestion

1. Sign in to the AWS Management Console, open the IAM console and go to the [Roles page](#).
2. Select **Create role**.
3. On the **Create role** page, make the following selections:
 - **Select type of trusted entity** – Choose **AWS Service**.
 - **Choose a use case** – Choose **AWS Marketplace**.
 - **Select your use case** – Choose **Marketplace – AMI Assets Ingestion**.
 - To move to the next page, select **Next: Permissions**.
4. Select the **AWSMarketplaceAmiIngestion** policy. Add a permissions boundary if required, and then select **Next: Tags** to continue.

Note

You can use permissions boundaries to limit the access that you give AWS Marketplace with this role. For more information, see [Permissions boundaries for IAM entities](#) in the *AWS Identity and Access Management User Guide*.

5. To continue, select **Next: Review**.
6. Provide a name for the role, and select **Create role**.

7. You should see "The role *rolename* has been created" at the top of the page, and the role should appear in the list of roles.

On this page, when you select the role that you just created, you can see its ARN in the form *arn:aws:iam::123456789012:role/exampleRole*. Use the ARN for the **IAM access role ARN** when you create change requests, for example, when [adding a new version](#) to your product.

Remove a product from AWS Marketplace

After your product is published, you can remove (also referred to as *sunset*) it from AWS Marketplace. To remove a product, identify the product and submit a request to remove it, along with a reason for removal and a contact email address for you. You can also provide a replacement product ID if you're replacing the current product with a new one. After you request product removal, new customers will no longer be able to subscribe to the product. You're required to support any existing customers for a minimum of 90 days. We process requests for product removal from AWS Marketplace with the following conditions:

- The product is removed from AWS Marketplace search, browse, and other discovery tools. Any **Subscribe** button or functionality is disabled, and messaging on the page clearly indicates the product is no longer available. The product detail page is still accessible using the URL and might be indexed in public search engines.
- A reason for removal must be specified (for example, end of support, end of product updates, or replacement product). For the requirements for continuing support for removed products, see [Terms and Conditions for AWS Marketplace Sellers](#).
- AWS Marketplace contacts current buyers through an email message informing them of the product removal, reasons for the removal, and to provide seller contact information.
- Current buyers *do* retain access to the software until they cancel their subscription. They aren't affected in any way by the product's removal.

To remove a product created using the AWS Marketplace Management Portal

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and then sign in to your seller account.
2. Choose the **Products** tab, and then choose **Server**.
3. On your product page, under **Server products**, locate the product that you want to remove. From the **Request changes** dropdown list, choose **Update product visibility**.

4. On the **Update product visibility** page, select **Restricted**.
5. (Optional) Provide a **Replacement Product ID**, if there is another product that will take the place of the product you are removing.
6. Review the information for accuracy, and then choose **Submit**.

A **What's next** informational page displays after you submit the product removal request. The AWS Marketplace Seller Operations reviews and processes your request. Check the status of your submission by viewing **Requests**.

After your product is removed, the product appears in the **Current Products** list in the AWS Marketplace Management Portal. In **Current Products**, the only action that you can perform is downloading the spreadsheet for the product. You can't edit or submit another sunset request.

If you have questions about product removals, contact the [AWS Marketplace Seller Operations](#) team.

Troubleshooting common errors when submitting change requests

When you make changes to your product's information, you might run into errors. Following are some common issues and suggestions for how to fix them:

- **Scanning your AMI** – Several issues could happen when scanning your AMI:
 - You have not granted AWS Marketplace permissions to scan your AMI. Grant AWS Marketplace permissions to access it. Or you have granted permissions, but the permissions boundary is too restrictive. For more information, see [Give AWS Marketplace access to your AMI](#).
 - If scanning finds security issues or Common Vulnerabilities and Exposures (CVEs) in your AMI, make sure you're using the latest patches for the operating system in your image. For more information, see [AMI-based product requirements](#).

For general guidelines about building an AMI, see [Best practices for building AMIs](#).

- **AWS Marketplace Management Portal fields** – Some fields in the AWS Marketplace Management Portal require very specific information:
 - If you are unsure about what the field is requesting, try checking the details in the console. Most fields have text descriptions above the field, and formatting requirements below the field.
 - If you try to submit a form with one or more invalid fields, a list of issues is shown. A recommended action is provided to help you fix the issue.

- If you're asked to provide an ARN, you will typically find it elsewhere in the console. For example, the ARN for the IAM role that you created to give AWS Marketplace access to your AMI is found on the [Roles page](#) in the IAM console. ARNs all have a similar format. For example, an IAM role ARN is in the form `arn:aws:iam::123456789012:role/exampleRole`.
- Your logos and videos must be provided as a URL directly to the content. For more information about logo formats, see [Company and product logo requirements](#).

For more information about submitting products and version change requests, see [Submitting your product for publication](#).

- **Product Load Form (PLF) issues** – PLFs contain instructions that are included in the spreadsheet. Overall instructions are provided in the Instructions table. Each field has instructions for how to fill it out—select the field to reveal the instructions.
- **Request in Progress** – Some requests can't happen in parallel. You can only have one request to update specific information in progress for a product at a time. You can see all of your requests still under review on the **Requests** tab of the **Server products** page in AWS Marketplace Management Portal. If you have a pending request that you did not intend, you can cancel it and then submit a new request with the change that you want to make.
 - You can't update version information when an update (to add or restrict) a version is ongoing.
 - If there is a request pending from the AWS Marketplace Seller Operations team, you can't submit any new changes.
- **Unexplained error** – If your submission fails with no explanation, try again. Occasionally, server load causes a submission to fail.

If you're still having problems with a change request, contact the [AWS Marketplace Seller Operations](#) team.

AMI-based delivery using AWS CloudFormation

Important

AWS Marketplace will discontinue the delivery method for multiple Amazon Machine Image (AMI) products using AWS CloudFormation templates in August 2024. The delivery method is only available to existing subscribers until it's discontinued. Other AWS Marketplace products using CloudFormation, such as single AMI with CloudFormation, won't be

affected. For more information, see [AWS CloudFormation template](#) in the *AWS Marketplace Buyer Guide*.

AWS Marketplace sellers can list AMI-based products that are delivered to AWS Marketplace buyers by using AWS CloudFormation templates. You can use the templates to define a cluster or distributed architecture for the products or to select different AMI combinations or product configurations. The CloudFormation templates can be configured to deliver a single Amazon Machine Image (AMI) or multiple AMIs along with associated configuration files and Lambda functions. Buyers can browse the selection of solutions on AWS Marketplace, buy with one click, and deploy by using CloudFormation templates that you provide.

Multi-AMI solutions can contain up to 20 AMIs and up to 20 CloudFormation templates. Each CloudFormation template can reference any combination or subset of the AMIs contained in the solution. The buyer purchases a single solution that entitles them to all of the AMIs in that product. When the product has multiple AMIs, each AMI has its own unique product code and can be priced and metered separately. However, individual components of a solution aren't discoverable or procurable outside the context of the product.

If you have existing single-AMI products, you can't migrate or combine them into a new multi-AMI listing. However, your new solution can feature the same software or copies of AMIs used by existing products. Each listing created on AWS Marketplace is a listing with new product codes.

You can also include Lambda functions in a serverless application with your AMI so that buyers can deploy them through CloudFormation. For instructions on how to include Lambda functions and serverless applications with your AMI, see [Adding serverless application components](#).

Building your product listing

To submit your product, you need to prepare and validate your AMIs, create your AWS CloudFormation templates, create an architectural diagram, complete the product load form, and submit the materials to AWS Marketplace. We recommend that you start by creating and validating your AMIs and then complete and validate the CloudFormation templates. After you complete those steps, you should create an architectural diagram and estimate the software and infrastructure price. AWS Marketplace validates your submission and works with you to make your product public. Use [AWS Pricing Calculator](#) to help estimate the infrastructure cost for your template. Provide AWS Marketplace with a link to your saved calculator configuration. The following are limitations of multi-AMI solution products:

- Updating existing AWS Marketplace products from a standalone product to a multi-AMI product isn't supported. To make a product available in a multi-AMI product, copy the AMI and submit it as a component to a new multi-AMI product. The resulting AMI has a unique product code that's different from the previous product's code.
- Multi-AMI solutions aren't visible on the **AWS Marketplace** tab of the **Launch Instance** page in the Amazon Elastic Compute Cloud (Amazon EC2) console.
- A CloudFormation template must not launch AMIs outside of those listed in the multi-AMI solution.
- CloudFormation templates must be submitted in the form of a public URL. All nested template URLs contained in the template must also be publicly accessible.

Preparing your CloudFormation template

To build your CloudFormation templates, you must meet the template prerequisites and provide the required input and security parameters. When submitting your CloudFormation template, use the guidelines in the following sections.

Template prerequisites

- Verify that the template is launched successfully through the AWS CloudFormation console **in all AWS Regions enabled for your product**. You can use the [TaskCat tool](#) to test your templates.
- If you are creating a single-AMI product, the template must contain only one AMI.
- AMIs must be in a [mapping table](#) for each Region. The AWS Marketplace team updates the AMI IDs after they're cloned. Your source AMI must be in us-east-1 and the other Regions can use placeholders. See the following YAML example.

```
Mappings:
  RegionMap:
    us-east-1:
      ImageId: ami-0123456789abcdef0
    us-west-1:
      ImageId: ami-xxxxxxxxxxxxxxxxxxx
    eu-west-1:
      ImageId: ami-xxxxxxxxxxxxxxxxxxx
    ap-southeast-1:
      ImageId: ami-xxxxxxxxxxxxxxxxxxx
```

- Build templates so that they do not depend on the use in a particular Availability Zone (AZ). Not all customers have access to all AZs, and AZs are mapped differently for different accounts.
- You can include dependencies such as Lambda functions, configuration files, and scripts with your AMI. For more information, see [Create a serverless application](#).
- If you're building a clustered solution using an Auto Scaling group, we recommend that you account for a scaling event. The new node should join the running cluster automatically.
- Even for single-node products, we recommend using an [Auto Scaling group](#).
- If your solution involves a cluster of multiple instances, consider using placement groups if you want low network latency, high network throughput, or both among the instances.
- If your solution involves Docker containers, you must incorporate the Docker images into the AMI.
- For ease of review by the AWS Marketplace team and transparency to the customer, we recommend that you add comments in your **UserData** section.

Template input parameters

- Input parameters to the template must not include the AWS Marketplace customer's AWS credentials (such as passwords, public keys, private keys, or certificates).
- For sensitive input parameters such as passwords, choose the NoEcho property and enable stronger regular expression. For other input parameters, set the most common inputs along with appropriate helper text.
- Use AWS CloudFormation parameter types for inputs where available.
- Use `AWS::CloudFormation::Interface` to group and sort input parameters.
- Don't set any default values for the following input parameters:

Note

Customers must provide these as input parameters.

- Default CIDR ranges that allow ingress into remote access ports from the public internet
- Default CIDR ranges that allow ingress into database connection ports from the public internet
- Default passwords for users or databases

Network and security parameters

- Ensure that the default SSH port (22) or RDP port (3389) isn't open to 0.0.0.0.
- Instead of using the default virtual private cloud (VPC), we recommend that you build a VPC with appropriate access control lists (ACLs) and security groups.
- Enable access to the customer's AWS environment by using an AWS Identity and Access Management (IAM) role to call [AssumeRole](#) from the AWS Security Token Service.
- Set IAM roles and policies to [grant the least privilege](#) and enable write access only when absolutely necessary. For example, if your application needs only S3:GET, PUT, and DELETE operations, specify those actions only. We don't recommend the use of S3:* in this case.

After your template is received, AWS Marketplace validates the product configuration and information and provides feedback for any required revisions.

Getting the cost estimate for your template infrastructure

The infrastructure cost estimate for each template displayed to customers is based on an estimate that you provide by using [AWS Pricing Calculator](#). The estimation should include the list of services to be deployed as part of the template, along with the default values for a typical deployment.

After you calculate the template's estimated monthly cost, provide AWS Marketplace with the **Save and Share** link for the US East (N. Virginia) Region. This is part of the submission process.

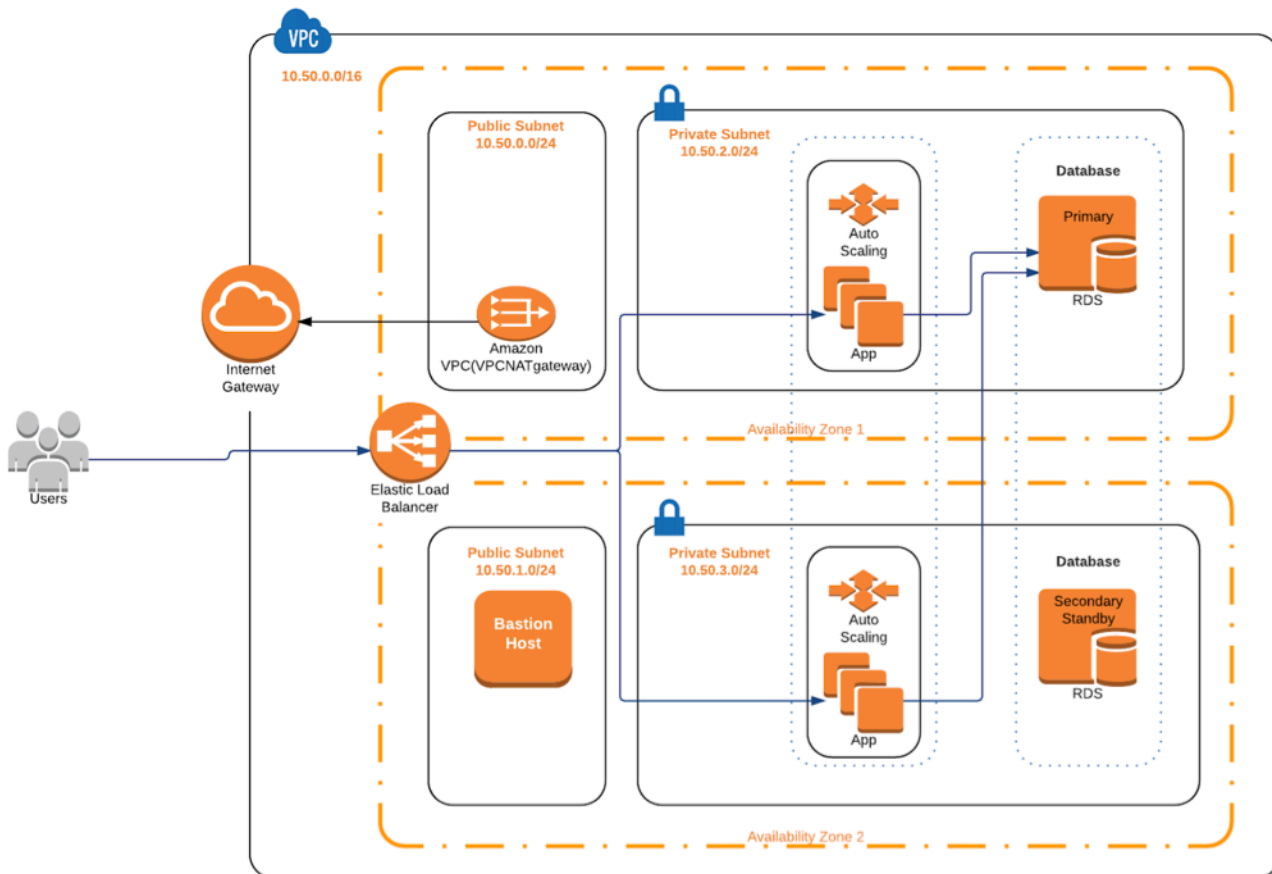
Architectural diagram

You must provide an architectural diagram for each template. The diagram must meet the following criteria:

- Illustrate standard deployment(s) on AWS
- Use the [AWS product icons](#) for each AWS service deployed through the AWS CloudFormation template
- Include metadata for all the services deployed by the AWS CloudFormation template
- Include all networks, VPCs, and subnets deployed by the AWS CloudFormation template
- Show integration points, including third party assets, APIs and on-premises, hybrid assets
- Must be 1100 x 700 pixels in size

Note

Make sure that your diagram meets this sizing requirement to avoid cropping or stretching, as shown in the following image.



Meeting the submission requirements

To submit products delivered by using AWS CloudFormation templates, you must provide the following resources:

- CloudFormation template or templates
 - A single-AMI product can have one to three CloudFormation templates
 - A multi-AMI product can have up to 20 CloudFormation templates
- The estimated infrastructure price for the default configuration of each template

- An architectural diagram and architectural metadata
- Completed product form (available from the [AWS Marketplace Management Portal](#))
 - For single-AMI products, use the [Commercial Product](#) form
 - For multi-AMI products, use the [Multi-AMI Product](#) form

The product forms include example submissions for your reference.

For each product, most of the required product data and metadata are the same as for traditional single-AMI products. Therefore, each AMI that is delivered by using an CloudFormation template must continue to meet the standards and requirements described for AWS Marketplace.

For each CloudFormation template, you must also provide the following information.

Field	Description	Restrictions
Title	Title of the architecture. This appears on the detail and fulfillment pages and the pop-up that shows the architectural details.	50 characters
Short description	This appears on the detail and fulfillment pages.	200 characters
Long description	This appears in the architectural details pop-up.	2000 characters

For multi-AMI products, the following fields are required:

- Solution title
- Solution short description
- Solution long description
- For CloudFormation templates (up to 20 per solution)
 - Deployment title (per template)
 - Short description (per template)
 - Long description (per template)

- Architecture diagram (per template)
- Infrastructure pricing estimate (per template)
- List of products/components contained in this CloudFormation template
- List of Regions supported by this CloudFormation template

Submitting your product request

Use the [AWS Marketplace Management Portal](#) to submit your product. On the **Assets** tab, choose **File Upload**. Upload files you want to submit and enter a brief description. Both YAML and JSON formats are supported. Allow three to five weeks for request processing, including:

- Review of the CloudFormation template, AMI, and metadata for the AMI and CloudFormation template
- Publication of your CloudFormation template to AWS Marketplace products

Adding serverless application components

You can create a product that includes one or more Amazon Machine Images (AMIs), delivered using one or more AWS CloudFormation templates, with serverless components incorporated into the product. For example, create a product with one AMI configured as a controller server and another AMI configured as a worker server, delivered as a AWS CloudFormation stack. The AWS CloudFormation template used to create the stack can include the definition to set up an AWS Lambda function that is triggered by an event in one of the servers.

When you use this approach to design your product, you can simplify the architecture and make it easier for your buyers to launch. This approach can also make it easier for you to update your product.

For information about creating AMIs for your product, see [AMI-based products](#). For information about completing AWS CloudFormation templates for your product, see [AMI-based delivery using AWS CloudFormation](#).

When you define your serverless application, you use an AWS Serverless Application Model (AWS SAM) template that you store in the AWS Serverless Application Repository. AWS SAM is an open-source framework for building serverless applications. During deployment, AWS SAM transforms and expands the AWS Serverless Application Model syntax into AWS CloudFormation syntax. The AWS Serverless Application Repository is a managed repository for serverless applications. It makes

it possible for you to store and share reusable applications so buyers can assemble and deploy serverless architectures. To create and offer this type of product, complete the following steps:

Steps

- [Create a serverless application](#)
- [Publish your application to the repository](#)
- [Create the CloudFormation template](#)
- [Submit your CloudFormation template and configuration files](#)
- [Update your AWS Serverless Application Repository application permissions](#)
- [Share your AMI](#)
- [Submit your CloudFormation product with AMI and serverless application](#)

AWS Marketplace reviews and validates your product before your listing is created. If there are issues you must resolve before the offer is listed, we will send you an email message.

As part of fulfilling a subscription, we copy the AMIs, serverless applications, and AWS CloudFormation templates to an AWS Marketplace-owned repository in each AWS Region. When a buyer subscribes to your product, we give them access, and also notify them when you update your software.

Create a serverless application

Your first step is to package the AWS Lambda functions used to create your serverless application. Your application is a combination of Lambda functions, event sources, and other resources that work together to perform tasks. A serverless application can be as simple as one Lambda function or contain multiple functions with other resources, such as APIs, databases, and event source mappings.

Use the AWS SAM to define a model for your serverless application. For descriptions of property names and types, see [AWS::Serverless::Application](#) in AWS Labs on GitHub. The following is an example of an AWS SAM template with a single Lambda function and AWS Identity and Access Management (IAM) role.

```
AWSTemplateFormatVersion: '2010-09-09'  
Transform: AWS::Serverless-2016-10-31  
Description: An example of SAM template with Lambda function and IAM role  
  
Resources:
```



```

SampleFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: 'com.sampleproject.SampleHandler::handleRequest'
    Runtime: java8
    CodeUri: 's3://DOC-EXAMPLE-BUCKET/2EXAMPLE-1234-4b12-ac37-515EXAMPLEe5-
lambda.zip'
    Description: Sample Lambda function
    Timeout: 120
    MemorySize: 1024
    Role:
      Fn::GetAtt: [SampleFunctionRole, Arn]

# Role to execute the Lambda function
SampleFunctionRole:
  Type: "AWS::IAM::Role"
  Properties:
    AssumeRolePolicyDocument:
      Statement:
        - Effect: "Allow"
          Principal:
            Service:
              - "lambda.amazonaws.com"
          Action: "sts:AssumeRole"
    ManagedPolicyArns:
      - "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
    Policies:
      - PolicyName: SFNXDeployWorkflowDefinitionPolicy
        PolicyDocument:
          Statement:
            - Effect: "Allow"
              Action:
                - "s3:Get*"
              Resource: "*"
    RoleName: "SampleFunctionRole"

```

Publish your application to the repository

To publish an application, you first upload the application code. Store your code artifacts (for example, Lambda functions, scripts, configuration files) in an Amazon S3 bucket that your account owns. When you upload your application, it's initially set to private, meaning that it's only available to the AWS account that created it. You must create an IAM policy that grants AWS Serverless Application Repository permissions to access the artifacts you uploaded.

To publish your serverless application to the serverless application repository

1. Open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. Choose the Amazon S3 bucket that you used to package your application.
3. Choose the **Permissions** tab.
4. Choose **Bucket Policy**.
5. Copy and paste the following example policy statement.

Note

The example policy statement will produce an error until values for `aws:SourceAccount` and `Resource` are updated in following steps.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "serverlessrepo.amazonaws.com"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        }
      }
    }
  ]
}
```

- a. Replace `DOC-EXAMPLE-BUCKET` in the `Resource` property value with the bucket name for your bucket.
- b. Replace `123456789012` in the `Condition` element with your AWS account ID. The `Condition` element ensures that the AWS Serverless Application Repository only has permission to access applications from the specified AWS account.

6. Choose **Save**.
7. Open the AWS Serverless Application Repository console at <https://console.aws.amazon.com/serverlessrepo>.
8. On the **My Applications** page, choose **Publish application**.
9. Complete the required fields and any optional field, as appropriate. The required fields are:
 - **Application name**
 - **Author**
 - **Description**
 - **Source code URL**
 - **SAM template**
10. Choose **Publish Application**.

To publish subsequent versions of your application

1. Open the AWS Serverless Application Repository console at <https://console.aws.amazon.com/serverlessrepo>.
2. In the navigation pane, from **My Applications**, choose the application.
3. Choose **Publish new version**.

For more information, see [Publishing serverless Applications Using the AWS SAM CLI](#).

Create the CloudFormation template

To build your CloudFormation templates, you must meet the template prerequisites and provide the required input and security parameters. For more information, see [Template anatomy](#) in the *AWS CloudFormation User Guide*.

In your CloudFormation template, you can reference your serverless application and your AMI. You can also use nested CloudFormation templates and reference serverless applications both in the root template and the nested templates. To reference the serverless application, you use the AWS SAM template. You can automatically generate the AWS SAM template for your application from the AWS Serverless Application Repository. The following is an example template.

```
AWSTemplateFormatVersion: '2010-09-09'  
Transform: AWS::Serverless-2016-10-31
```

```
Description: An example root template for a SAR application

Resources:
  SampleSARApplication:
    Type: AWS::Serverless::Application
    Properties:
      Location:
        ApplicationId: arn:aws:serverlessrepo:us-east-1:1234567890:applications/
TestApplication
  SemanticVersion: 1.0.0
  SampleEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: "ami-79fd7eee"
      KeyName: "testkey"
      BlockDeviceMappings:
        - DeviceName: "/dev/sdm"
          Ebs:
            VolumeType: "io1"
            Iops: "200"
            DeleteOnTermination: "false"
            VolumeSize: "20"
        - DeviceName: "/dev/sdk"
          NoDevice: {}
```

The AWS SAM template contains the following elements:

- **ApplicationID** – Your application's Amazon Resource Name (ARN). This information is located in the **My Applications** section of the AWS Serverless Application Repository.
- **SemanticVersion** – The version of your serverless application. You can find this from the **My Applications** section of the AWS Serverless Application Repository.
- **Parameter (optional)** – Application parameters.

Note

For **ApplicationID** and **SemanticVersion**, [intrinsic functions](#) aren't supported. You must hardcode those strings. The **ApplicationID** is updated when it's cloned by AWS Marketplace.

If you're planning to reference configuration and script files in your CloudFormation template, use the following format. For nested templates (AWS::CloudFormation::Stack), only TemplateURLs without intrinsic functions are supported. Note the Parameters content in the template.

```

AWSTemplateFormatVersion: '2010-09-09'
Metadata:
  Name: Seller test product
Parameters:
  CFRefFilesBucket:
    Type: String
    Default: "seller-bucket"
  CFRefFilesBucketKeyPrefix:
    Type: String
    Default: "cftsolutionFolder/additionCFfiles"
Resources:
  TestEc2:
    Type: AWS::EC2::Instance
    Metadata:
      AWS::CloudFormation::Init:
        addCloudAccount:
          files:
            /etc/cfn/set-aia-settings.sh:
              source:
                Fn::Sub:
                  - https://{CFRefFilesBucket}.${S3Region}amazonaws.com/
                    ${CFRefFilesBucketKeyPrefix}/sampleScript.sh
                  - S3Region:
                      !If
                        - GovCloudCondition
                        - s3-us-gov-west-1
                        - s3
                  owner: root
                  mode: '000700'
                  authentication: S3AccessCreds
          ..
          ..
          ..
  SampleNestedStack:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: 'https://sellerbucket.s3.amazon.com/sellerproductfolder/
nestedCft.template'

```

```
Parameters:
  SampleParameter: 'test'
Transform: AWS::Serverless-2016-10-31
```

Submit your CloudFormation template and configuration files

To submit your CloudFormation template and configuration and scripts files, grant AWS Marketplace permissions to read the Amazon S3 bucket where these files are stored. To do so, update your bucket policy to include the following permissions.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "assets.marketplace.amazonaws.com"
      },
      "Action": ["s3:GetObject", "s3:ListBucket"],
      "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET",
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"]
    }
  ]
}
```

Update your AWS Serverless Application Repository application permissions

To submit your AWS Serverless Application Repository application to AWS Marketplace, you must grant AWS Marketplace permissions to read your application. To do that, add permissions to a policy associated with your serverless application. There are two ways to update your application policy:

- Go to the [AWS Serverless Application Repository](#). Choose your serverless application from the list. Select the **Sharing** tab, and choose **Create Statement**. On the **Statement configuration** page, enter the following service principal, `assets.marketplace.amazonaws.com`, in the **Account Ids** field. Then choose **Save**.
- Use the following AWS CLI command to update your application policy.

```
aws serverlessrepo put-application-policy \  
--region region \  

```

```
--application-id application-arn \  
--statements Principals=assets.marketplace.amazonaws.com,Actions=Deploy
```

Share your AMI

All AMIs built and submitted to AWS Marketplace must adhere to all product policies. Self-service AMI scanning is available in the AWS Marketplace Management Portal. With this feature, you can initiate scans of your AMIs. You receive scanning results quickly (typically, in less than an hour) with clear feedback in a single location. After your AMI has been successfully scanned, submit the AMI for processing by the AWS Marketplace Seller Operations team by uploading your product load form.

Submit your CloudFormation product with AMI and serverless application

Keep the following in mind before you submit your product:

- You must provide an architectural diagram for each template. The diagram must use the AWS product icons for each AWS service deployed through the CloudFormation template. Also, the diagram must include metadata for the services. To download our official AWS architecture icons, see [AWS Architecture Icons](#).
- The infrastructure cost estimate for each template displayed to buyers is based on an estimate that you provide by using the [AWS Pricing Calculator](#). In the estimate, include the list of services to be deployed as part of the template, along with the default values for a typical deployment.
- Complete the product load form. You can find the product load form from the AWS Marketplace Management Portal. A different product load form is required for single AMI products and multiple AMI products. In the product load form, you will provide a public URL to your CloudFormation template. CloudFormation templates must be submitted in the form of a public URL.
- Use the AWS Marketplace Management Portal to submit your listing. From **Assets**, choose **File upload**, attach your file, and then choose **Upload**. After we receive your template and metadata, AWS starts processing your request.

After you submit your listing, AWS Marketplace reviews and validates the product load form. Additionally, AWS Marketplace regionalizes AMIs and serverless applications, and updates the regional mapping for your AWS CloudFormation template on your behalf. If any issues occur, the AWS Marketplace Seller Operations team will contact you by email.

Private images

Important

AWS Marketplace will discontinue the Private Image Build delivery method in April 2024. The delivery method is only available to existing subscribers until it's discontinued. For more information, see [Private image build](#) in the *AWS Marketplace Buyer Guide*.

You can use private image builds to let buyers purchase your installable software products through AWS Marketplace. Then, buyers can install those products on a gold image or Amazon Machine Image (AMI) that they choose from the images available to their AWS account. A *gold image* is a buyer-provided server image that includes a base operating system with modifications applied to help ensure the software adheres to the buyer's IT standards. Gold images allow buyers to better meet their internal security, compliance, and management requirements.

If you have questions about private image builds, contact the [AWS Marketplace Seller Operations](#) team.

Best practices for building AMIs

This topic provides some best practices and references to help you build Amazon Machine Images (AMIs) for use with AWS Marketplace. AMIs built and submitted to AWS Marketplace must adhere to all AWS Marketplace product policies.

Securing resell rights

You are responsible for securing resell rights for non-free Linux distributions, with the exception of AWS-provided Amazon Linux, RHEL, SUSE, and Windows AMIs.

Building an AMI

Use the following guidelines for building AMIs:

- Ensure that your AMI meets all AWS Marketplace policies, including disabling root login.
- Create your AMI in the US East (N. Virginia) Region.

- Create products from existing, well-maintained AMIs backed by Amazon Elastic Block Store (Amazon EBS) with a clearly defined lifecycle provided by trusted, reputable sources such as AWS Marketplace.
- Build AMIs using the most up-to-date operating systems, packages, and software.
- Ensure that all AMIs must start with a public AMI that uses hardware virtual machine (HVM) virtualization and 64-bit architecture.
- Develop a repeatable process for building, updating, and republishing AMIs.
- Use a consistent operating system (OS) user name across all versions and products. We recommend **ec2-user**.
- Configure a running instance from your final AMI to the end-user experience you want, and test all installation methods, features, and performance *before* submission to AWS Marketplace.
- Check port settings as follows:
 - Linux-based AMIs – Ensure that a valid SSH port is open. The default SSH port is 22.
 - Windows-based AMIs – Ensure that an RDP port is open. The default RDP port is 3389. Also, the WinRM port (5985 by default) must be open to 10.0.0.0/16 and 10.2.0.0/16.

For more information about creating an AMI, see the following resources:

[Creating Your Own AMI](#) in the *Amazon EC2 User Guide for Linux Instances*

[Creating a Custom Windows AMI](#) in the *Amazon EC2 User Guide for Windows Instances*

[How do I create an Amazon Machine Image \(AMI\) from an EBS-backed instance?](#)

[Amazon Linux AMI](#)

[Amazon EC2 Instance Types](#) and [Instance Types](#)

Preparing and securing your AMI for AWS Marketplace

We recommend the following guidelines for creating secure AMIs:

- Use the [Guidelines for Shared Linux AMIs](#) in the *Amazon EC2 User Guide for Linux Instances*
- Architect your AMI to deploy as a minimum installation to reduce the attack surface. Disable or remove unnecessary services and programs.

- Whenever possible, use end-to-end encryption for network traffic. For example, use Secure Sockets Layer (SSL) to secure HTTP sessions between you and your buyers. Ensure that your service uses only valid and up-to-date certificates.
- When adding a new version to your AMI product, configure security groups to control inbound traffic access to your instance. Ensure that your security groups are configured to allow access only to the minimum set of ports required to provide necessary functionality for your services. Allow administrative access only to the minimum set of ports and source IP address ranges necessary. For more information about how to add a new version to your AMI product, see [Add a new version](#).
- Consider performing a penetration test against your AWS computing environment at regular intervals, or consider employing a third party to conduct such tests on your behalf. For more information, including a penetration testing request form, see [AWS Penetration Testing](#).
- Be aware of the top 10 vulnerabilities for web applications, and build your applications accordingly. To learn more, see [Open Web Application Security Project \(OWASP\) - Top 10 Web Application Security Risks](#). When new internet vulnerabilities are discovered, promptly update any web applications that ship in your AMI. Examples of resources that include this information are [SecurityFocus](#) and the [NIST National Vulnerability Database](#).

For more information related to security, see the following resources:

- [AWS Cloud Security](#)
- [The Center for Internet Security \(CIS\): Security Benchmarks](#)
- [The Open Web Application Security Project \(OWASP\): Secure Coding Practices - Quick Reference Guide](#)
- [OWASP Top 10 Web Application Security Risks](#)
- [SANS \(SysAdmin, Audit, Networking, and Security\) Common Weakness Enumeration \(CWE\) Top 25 Most Dangerous Software Errors](#)
- [Security Focus](#)
- [NIST National Vulnerability Database](#)

Scanning your AMI for publishing requirements

To help verify your AMI before submitting it as a new product or version, you can use self-service scanning. The self-service scanner will check for unpatched common vulnerabilities and exposures

(CVEs) and verify security best practices are followed. For more information, see [the section called "Preparing and securing your AMI for AWS Marketplace"](#)

From the AWS Marketplace Management Portal, choose **Amazon Machine Image** from the **Assets** menu. Choose **Add AMI** to start the scanning process. You can see the scan status of AMIs by returning to this page.

Note

To learn about giving AWS Marketplace access to your AMI, see [Give AWS Marketplace access to your AMI](#).

Verifying your software is running on your AWS Marketplace AMI

You may wish to have your software verify at runtime that it is running on an Amazon EC2 instance created from your AMI product.

To verify the Amazon EC2 instance is created from your AMI product, use the instance metadata service built into Amazon EC2. The following steps take you through this validation. For more information about using the metadata service, see [Instance metadata and user data](#) in the *Amazon Elastic Compute Cloud User Guide*.

1. Obtain the instance identity document

Each running instance has an identity document accessible from the instance that provides data about the instance itself. The following example shows using curl from the instance to retrieve the instance identity document.

```
curl http://169.254.169.254/latest/dynamic/instance-identity/document
{
  "accountId" : "0123456789",
  "architecture" : "x86_64",
  "availabilityZone" : "us-east-1e",
  "billingProducts" : null,
  "devpayProductCodes" : null,
  "marketplaceProductCodes" : [ "0vg0000000000000000000000000000" ],
  "imageId" : "ami-0123456789abcdef1",
  "instanceId" : "i-0123456789abcdef0",
```

```
"instanceType" : "t2.medium",
"kernelId" : null,
"pendingTime" : "2020-02-25T20:23:14Z",
"privateIp" : "10.0.0.2",
"ramdiskId" : null,
"region" : "us-east-1",
"version" : "2017-09-30"
}
```

2. Verify the instance identity document

You can verify that the instance identity is correct using the signature. For details about this process, see [Instance identity documents](#) in the *Amazon Elastic Compute Cloud User guide*.

3. Verify the product code

When you initially submit your AMI product for publishing, your product is assigned a [product code](#) by AWS Marketplace. You can verify the product code by checking the `marketplaceProductCodes` field in the instance identity document, or you can get it directly from the metadata service:

```
curl http://169.254.169.254/latest/meta-data/product-codes
0vg0000000000000000000000000000
```

If the product code matches the one for your AMI product, then the instance was created from your product.

You may also wish to verify other information from the instance identity document, such as the `instanceId` and the `instancePrivateIp`.

AMI product pricing

AWS Marketplace has multiple pricing models for Amazon Machine Image (AMI) products. With seller private offers, there are options available for multi-year and custom duration contracts.

For more information about multi-year and custom duration contracts, see [Preparing your private offer](#) and [Installment plans](#). The following table provides general information about pricing models.

Note

You must be able to provide a W-9 tax form (for US based entities) or a W-8 form (for EU-based entities) as described in [Seller registration process](#).



AMI pricing models

The following table provides general information about pricing models for AMI-based products.

Pricing model	Description
Free	<p>Customers can run as many instances as Amazon Elastic Compute Cloud (Amazon EC2) supports with no additional software charges incurred.</p> <div data-bbox="669 886 800 926" data-label="Section-Header">Note</div> <div data-bbox="714 942 1421 1031" data-label="Text"> <p>Free Trial and Annual pricing can't be combined with Monthly pricing.</p> </div>
Bring your own license (BYOL)	<p>AWS Marketplace doesn't charge customers for usage of the software, but customers must supply a license key to activate the product. This key is purchased outside of AWS Marketplace. The entitlement and licensing enforcement, in addition to all pricing and billing, are handled by you.</p>
Paid hourly or hourly-annual	<p>Hourly – Software is charged by the hour. Each instance type can be priced differently (but it isn't required to be), and usage is rounded up to the nearest whole hour.</p> <p>Hourly with Free Trial – Customers are limited to running exactly one instance of the software without incurring a charge. You define the duration, between 5 and 30 days. The free trial applies to the most expensive instance type that is running, and any concurrent usage outside the 1 instance is billed at the hourly rate. NOTE: This is a different model than the AWS Free Tier for Amazon EC2</p>

Pricing model	Description
	<p>usage whereby customers are given 750 hours of free usage each month.</p> <p>Hourly and Monthly – Both hourly and monthly charges are applied independently. The monthly fee is charged every month regardless of usage, and the hourly fee is applied based on hourly usage only.</p> <p>Hourly with Annual – Customers have the option to purchase a year’s worth of usage upfront for one Amazon EC2 instance of one instance type. You set the pricing for each instance type and can offer net savings over the hourly price. Any customer usage above the number of annual subscriptions purchased is billed at the hourly rate you set for that instance type.</p> <p>Hourly with Multi-Annual and Custom Duration – This type of offer is only available through seller private offers. Using seller private offers, you specify a custom contract duration, up to 3 years. You can specify upfront payment, or include a flexible payment schedule. You set the pricing for each instance type. If there is a flexible payment schedule in the offer, you also set the invoice dates, payment amounts, and number of instances for each instance type included in the offer. For an active seller private offer with a flexible payment schedule, after the customer launches the specified number of instances, any additional instances launched are charged at the hourly rate specified in the seller private offer. For more information about multi-year and custom duration contracts, see Preparing your private offer and the section called “Installment plans”.</p> <p>Hourly with Annual (includes Free Trial) – This is identical to the Hourly model with an Annual option, except it includes a Free Trial allowing a customer to run one</p>

Pricing model	Description
	<p>instance of any instance type for free for a set number of days that you determine. Annual subscriptions can be purchased at any time, and they are combined with the Free Trial subscription.</p> <p>Annual with Hourly – Same as the Hourly with Annual pricing model. Customers have the option to purchase a year’s worth of usage upfront for one Amazon EC2 instance of one instance type. You set the pricing for each instance type and can offer net savings over the hourly price, but offering savings isn't required. Any customer usage above the number of annual subscriptions purchased is billed at the hourly rate you set for that instance type.</p> <p>Multi-Annual and Custom Duration with Hourly – This is only available through Preparing your private offer. Using seller private offers, you can specify a custom duration contract of up to three years. You can require upfront payment, or you can offer a flexible payment schedule to the customer. You set the pricing for each instance type for the duration of the contract, and the hourly pricing for additional instances launched. If you offer a flexible payment schedule, you also set the invoice dates, payment amounts, and number of instances for each instance type included in the offer. For an active private offer with a flexible payment schedule, after the specified number of instances have been launched, any additional instances the customer launches are charged at the hourly rate specified in the private offer. For more information about multi-year and custom duration contracts, see Preparing your private offer and the section called “Installment plans”.</p>

Pricing model	Description
	<p> Note</p> <p>Free Trial and Annual pricing can't be combined with Monthly pricing.</p>
Paid monthly	<p>Monthly – Software is paid for on a fixed monthly basis, regardless of the number of instances the customer runs. Monthly charges are pro-rated at sign-up and upon cancellation. Example: A customer who subscribes for 1 day of the month will be charged for 1/30th of the month.</p> <p>Monthly with Hourly – Both Hourly and Monthly charges are applied independently. The monthly fee is charged every month regardless of usage, and the hourly fee is applied based on hourly usage only.</p> <p> Note</p> <p>Free Trial and Annual pricing can't be combined with Monthly pricing.</p>
Paid usage pricing	<p>Software is directly charged for the value you provide along with one of four usage categories: users, data, bandwidth, or hosts. You can define up to 24 dimensions for the product. All charges are still incurred hourly by the customer.</p> <p>All usage is calculated monthly and billed monthly using the same mechanism as existing AWS Marketplace software. Usage pricing is also referred to as AWS Marketplace Metering Service.</p>
Contract pricing model	<p>AMI with contract pricing – A Single-AMI product or Single-AMI with AWS CloudFormation stack that the buyer pays an upfront fee for.</p>

AWS charges and software charges

Amazon Machine Image (AMI)-based product charges fall into two categories:

- **Infrastructure Pricing Details** – All AMI-based products incur associated AWS infrastructure charges depending on the services and infrastructure used. These rates and fees are defined and controlled by AWS, and can vary between AWS Regions. For more information, see [Amazon EC2 Pricing](#).
- **Software Pricing Details** – For Paid products, the seller defines the charges for using the software.

These two product charge categories are displayed separately on the AWS Marketplace detail pages to help buyers understand the potential cost of using the products.

Free trial for AMI hourly products

AMI hourly products are eligible for the optional Free trial program. In a Free trial, a customer can subscribe to the product and use a single instance for up to 31 days without paying software charges on the product. Applicable AWS infrastructure charges still apply. Free trials will automatically convert to a paid subscription upon expiration. Customers will be charged for additional usage above the free units provided. To offer an hourly product free trial, define the duration of the trial period and notify the [AWS Marketplace Seller Operations](#) team. The trial period can be 5–31 days.

When customers subscribe to a Free trial product, they receive a welcome email message that includes the term of the Free trial, a calculated expiration date, and details on unsubscribing. A reminder email message is sent three days before the expiration date.

If you offer a Free trial product in AWS Marketplace, you agree to the specific refund policies described under **Refund Policy**.

Custom metering pricing for AMI products

Metering service concepts

The AWS Marketplace Metering Service enables software sellers to modify their software to send metering records to an endpoint to capture usage. Sellers can select a usage category and define up to 24 dimensions of that one category. These dimensions are metered once per hour,

aggregated, and charged against a price plan defined by the seller. As a seller, you must determine which dimension you want to use. After the AMI is published, you will not be able to change it. Important service concepts include the following:

- **Usage Category** – Any software product priced through the use of the Metering Service is categorized according to one usage category, which determines the appropriate way to charge customers. Usage categories include but aren't limited to:
 - **Users** – A defined set of permissions associated with a single identifier. This category is appropriate for software in which a customer's users connect to the software directly (for example, for customer-relationship management or business intelligence reporting).
 - **Hosts** – Any server, node, instance, endpoint, or other part of a computing system. This category is appropriate for software that monitors or scans many customer-owned instances (for example, performance or security monitoring).
 - **Data** – Storage or information, measured in MB, GB, or TB. This category is appropriate for software that manages stored data or processes data in batches.
 - **Bandwidth** – Measured in Mbps or Gbps. This category is appropriate for software that allows customers to specify an amount of bandwidth to provision.
 - **Unit** – Unit of measurement; see the examples described next.
- **Usage Unit** – A software product's specific usage unit corresponds to the selected usage category. This usage unit describes the unit your software will charge on. Examples include:
 - **NodesHrs** (corresponding to the Hosts category)
 - **UserHrs** (corresponding to the User category)
 - **GBStored** (corresponding to the Data category)
- **Consumption** – Software products priced through the use of the Metering Service charge for consumption in one of three ways:
 - **Provisioned** – The software allows customers to configure a specific amount of resources for use (for example, number of users or a fixed amount of bandwidth). Each hour, customers pay for what they have provisioned.
 - **Concurrent** – The software allows any number of distinct hosts or users to connect to the software. Each hour, customers pay based on the number of hosts or users who accessed the software.
 - **Accumulated** – The software allows customers to use any amount of data, either processed or stored. Each hour, customers pay for the aggregated amount.

- **Pricing** – Software products priced through the use of the Metering Service must specify either a single price or define up to 24 dimensions, each with their own price. Details about the pricing options include:
 - **Single dimension** – This is the simplest pricing option. Customers pay a single price per resource unit per hour, regardless of size or volume (for example, \$0.014 per user per hour, or \$0.070 per host per hour).
 - **Multiple dimensions** – This pricing option is appropriate when the selected usage category varies along multiple axes. For example, for host monitoring, a different price could be set depending on the size of the host. Or, for user-based pricing, a different price could be set based on the type of user (for example, admin, power user, and read-only user).
- **Metering** – All usage is recorded as a metering event, once each hour. Your software must be configured to send the appropriate dimension and usage amount to the AWS Marketplace Metering Service.
 - **Allocations** – Optionally, you may distribute the usage into allocations by properties that you track. These allocations are represented as tags to the buyer. The tags allow the buyer to view their costs split into usage by tag. For example, if you charge by the user, and users have a "Department" property, you could create usage allocations with tags that have a key of "Department", and one allocation per value. This approach doesn't change the price, dimensions, or the total usage that you report. However, it allows your customer to view their costs by categories appropriate to your product.

Pricing your software

When pricing your software with the AWS Marketplace Metering Service, you must first decide on a usage category and how it will be consumed. The service supports six distinct pricing scenarios. You must select only one of these for your product:

- Provisioned user (per hour)
- Concurrent user (per hour)
- Provisioned host (per hour)
- Concurrent host (per hour)
- Provisioned bandwidth (per hour)
- Accumulated data (per hour)

Next, you must decide how to price the selected usage category:

- Single price
- Multiple dimensions (up to 24)

[Adding your product to AWS Marketplace](#) describes how to provide a customer-friendly description of your dimension and pricing.

Example: Provisioned bandwidth with nonlinear pricing

Imagine you offer network appliance software. You choose to bill by provisioned bandwidth. For your usage category, select **Bandwidth**. In addition to charging by bandwidth, you want to charge a different price as buyers scale up. You can define multiple dimensions within the bandwidth category. You can define a distinct price for 25 Mbps, 100 Mbps, and 1 Gbps.

Example: Concurrent hosts with multiple dimensions

Imagine you offer software that monitors other Amazon EC2 instances. You choose to bill by the number of hosts that are being monitored. For your usage category, select **Host**. In addition to charging by host, you want to charge for the extra value for monitoring larger hosts. You can use multiple dimensions within the host category. You can define a distinct price for micro, small, medium, large, x-large, 2XL, 4XL, and 8XL instances. Your software is responsible for mapping each particular host to one of your defined dimensions. Your software is responsible for sending a separate metering record for each dimension of your usage category if applicable.

Adding your product to AWS Marketplace

To take advantage of the Metering Service, you must create a new product for AWS Marketplace to list. If your product is already on the AWS Marketplace, you will need to decide whether the new AWS Marketplace Metering Service product will be made available in addition to your current product, or if it will replace your current product as the only version available to new users. If you choose replacement, the existing product will be removed from the AWS Marketplace so that it is no longer available for new buyers. Existing customers will continue to have access to their old product and instances, but they can migrate to the new product at their convenience. The new product must meter usage to the AWS Marketplace Metering Service, as described in [Modifying your software to use the Metering Service](#).

After you have your AMI, follow the standard process to share and scan your AMI using the self-service tool. In addition to using the template available on the management portal, fill out the product load form and upload it to start the ingestion process.

Use the following definitions to complete the fields of the Product Load Form for the AWS Marketplace Metering Service. On the Product Load Form, these fields are labeled as **Flexible Consumption Pricing (FCP)** to differentiate them from hourly and monthly priced products.

- **Title** – If you already have a product on AWS Marketplace and you're adding the same product with the AWS Marketplace Metering Service, include the FCP category and dimension in parentheses to differentiate them (for example, "PRODUCT TITLE (Data)").
- **Pricing Model** –From the dropdown list, choose **Usage**.
- **FCP Category** – The category in which customers are charged for paid products with a **Usage** pricing component. From the dropdown list, choose **Users, Hosts, Data, or Bandwidth**.
- **FCP Unit** –The unit of measurement on which customers are charged for paid products with a **Usage** pricing component. Options will appear in the dropdown list based on the FCP category you selected. The following table lists the valid units for each category.

Category	Valid units
Users	UserHrs
Hosts	HostHrs
Data	MB, GB, TB
Bandwidth	Mbps, Gbps

- **FCP Dimension Name** – The name used when sending metering records by calling the MeterUsage operation. It is visible in billing reports. However, because it isn't external-facing, the name doesn't need to be user-friendly. The name can be no more than 15 characters and can only include alphanumeric and underscore characters. After you set the name and make the product public, you can't change it. Changing the name requires a new AMI.
- **FCP Dimension Description** – The customer-facing statement that describes the dimension for the product. The description (can be no more than 70 characters and should be user-friendly. Examples of descriptions include: Administrators per hour and Per Mbps bandwidth provisioned. After the product is published, you can't change this description.
- **FCP Rate** – The software charge per unit for this product. This field supports three decimal places.

Notes:

- You don't need to fill out hourly and annual pricing fields.
- Free trial and annual pricing aren't compatible.
- Products that use multiple AMIs and the Clusters and AWS Resources feature can't use the AWS Marketplace Metering Service.
- Price, instance type, or AWS Region change will follow the same process as other AWS Marketplace products.
- Products with the AWS Marketplace Metering Service can't be converted to other pricing models such as hourly, monthly, or Bring Your Own License (BYOL).
- AWS Marketplace recommends adding IAM policy information in your usage instructions or document.
- You can include up to 24 FCP dimensions in total. Once created and published, you can't modify existing dimensions, but you can add new ones (up to the limit of 24).

If you have questions, contact the [AWS Marketplace Seller Operations](#) team.

Modifying your software to use the Metering Service

You will need to modify your software to record customer usage, send hourly usage reports to the Metering Service, and handle new failure modes. The software operates independently of pricing, but the software will need to know about the usage category, how it's consumed, and any dimensions.

Measuring consumption

Your software must determine how much of the selected usage category and which dimensions the customer has consumed. This value will be sent, once each hour, to the Metering Service. In all cases, it's assumed that your software has the ability to measure, record, and read consumption of resources for the purpose of sending it on an hourly basis to the Metering Service.

For provisioned consumption, this will typically be read from the software configuration as a sampled value, but might also be a maximum configured value, recorded each hour. For concurrent consumption, this might be either a periodic sample or a maximum value recorded each hour. For accumulated consumption, this will be a value that is accumulated each hour.

For pricing on multiple dimensions, multiple values must be measured and sent to the Metering Service, one per dimension. This requires your software to be programmed or configured with the known set of dimensions when you provide the AMI. The set of dimensions can't change after a product is created.

For each pricing scenario, the following table describes recommended ways for measuring consumption each hour.

Scenario	How to measure
Provisioned user	Current number of provisioned users (sampled). -OR- Maximum number of provisioned users (seen that hour).
Concurrent user	Current number of concurrent users (sampled). -OR- Maximum number of concurrent users (seen that hour). -OR- Total number of distinct users (seen that hour).
Provisioned host	Current number of provisioned hosts (sampled). -OR- Maximum number of provisioned hosts (seen that hour).
Concurrent host	Current number of concurrent hosts (sampled)

Scenario	How to measure
	-OR- Maximum number of concurrent hosts (seen that hour). -OR- Total number of distinct hosts (seen that hour).
Provisioned bandwidth	Current provisioned bandwidth setting (sampled). -OR- Maximum provisioned bandwidth (seen that hour).
Accumulated data	Current GB of data stored (sampled). -OR- Maximum GB of data stored (seen that hour). -OR- Total GB of data added or processed that hour. -OR- Total GB of data processed that hour.

Vendor-metered tagging (Optional)

Vendor-metered tagging helps Independent Software Vendors (ISVs) give the buyer more granular insight into their software usage and can help them perform cost allocation.

To tag a buyer's software usage, you need to determine how costs are allocated. First ask your buyers what they want to see in their cost allocation. Then you can split the usage across

properties that you track for the buyer's account. Examples of properties include Account ID, Business Unit, Cost Centers, and other relevant metadata for your product. These properties are exposed to the buyer as tags. Using tags, buyers can view their costs split into usage by the tag values in their AWS Billing Console (<https://console.aws.amazon.com/billing/>). Vendor-metered tagging doesn't change the price, dimensions, or the total usage that you report. It allows your customer to view their costs by categories appropriate to your product.

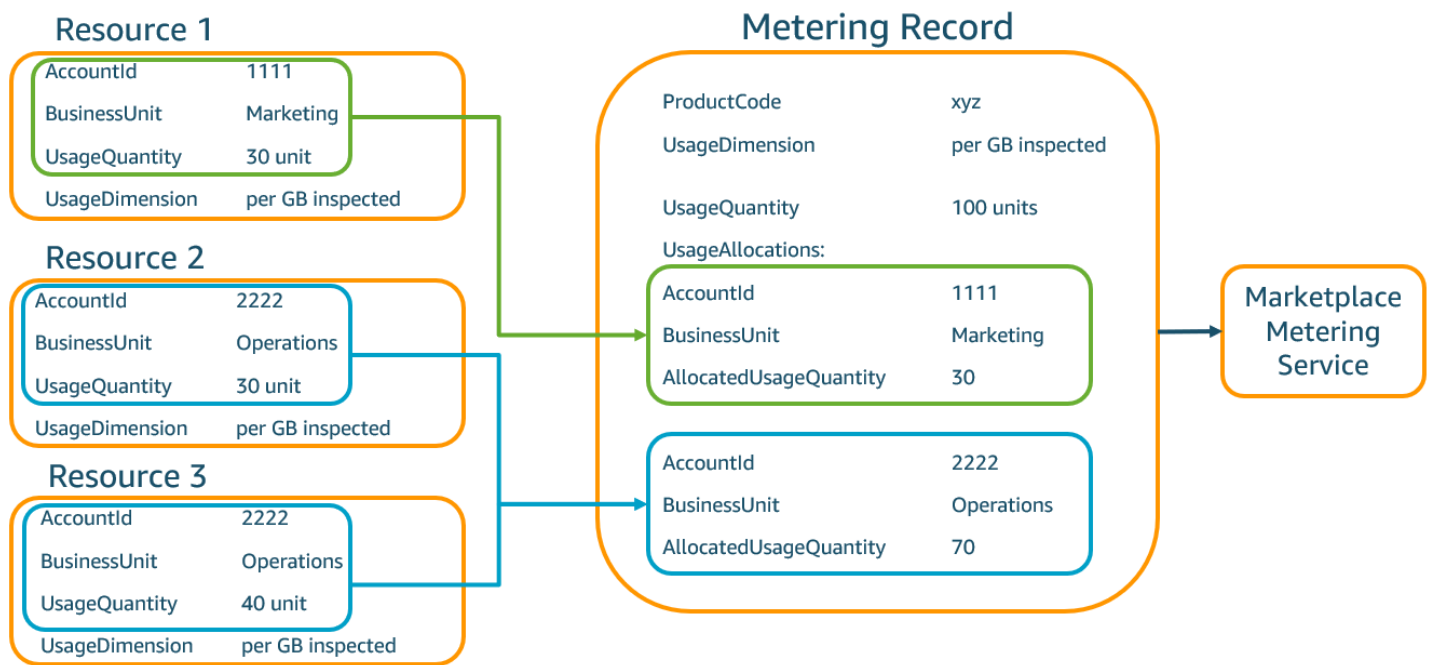
In a common use case, a buyer subscribes to your product with one AWS account. The buyer also has numerous users associated with the same product subscription. You can create usage allocations with tags that have a key of Account ID, and then allocate usage to each user. In this case, buyers can activate the Account ID tag in their Billing and Cost Management console and analyze individual user usage.

Seller experience

Sellers can aggregate the metering records for resources with the same set of tags instead of aggregating usage for all resources. For example, sellers can construct the metering record that includes different buckets of UsageAllocations. Each bucket represents UsageQuantity for a set of tags, such as AccountId and BusinessUnit.

In the following diagram, **Resource 1** has a unique set of AccountId and BusinessUnit tags, and appears in the **Metering Record** as a single entry.

Resource 2 and **Resource 3** both have the same AccountId tag, 2222, and the same BusinessUnit tag, Operations. As a result, they're combined into a single UsageAllocations entry in the **Metering Record**.



Sellers can also combine resources without tags into a single UsageAllocation with the allocated usage quantity and send it as one of the entries in UsageAllocations.

Limits include:

- Number of tags – 5
- Size of UsageAllocations (cardinality) – 2,500

Validations include:

- Characters allowed for the tag key and value – a-zA-Z0-9+ - = . : \ / @
- Maximum tags across UsageAllocation list – 5
- Two UsageAllocations can't have the same tags (that is, the same combination of tag keys and values). If that's the case, they must use the same UsageAllocation.
- The sum of AllocatedUsageQuantity of UsageAllocation must equal the UsageQuantity, which is the aggregate usage.

Buyer experience

The following table shows an example of the buyer experience after a buyer activates the AccountId and BusinessUnit vendor tags.

In this example, the buyer can see allocated usage in their **Cost Usage Report**. The vendor-metered tags use the prefix “aws:marketplace:isv”. Buyers can activate them in the Billing and Cost Management, under **Cost Allocation Tags, AWS-generated cost allocation tags**.

The first and last rows of the **Cost Usage Report** are relevant to what the Seller sends to the Metering Service (as shown in the [Seller experience](#) example).

Cost Usage Report (Simplified)

ProductCode	Buyer	UsageDimension	UsageQuantity	aws:marketplace:isv:AccountId	aws:marketplace:isv:BusinessUnit
xyz	111122223333	Network: per (GB) inspected	70	2222	Operations
xyz	111122223333	Network: per (GB) inspected	30	3333	Finance
xyz	111122223333	Network: per (GB) inspected	20	4444	IT
xyz	111122223333	Network: per (GB) inspected	20	5555	Marketing
xyz	111122223333	Network: per (GB) inspected	30	1111	Marketing

For a code example, see [MeterUsage with usage allocation tagging \(Optional\)](#)

Contract pricing for AMI products

Contract pricing for Amazon Machine Image (AMI)-based products means that the buyer pays an upfront fee for a single AMI product or single AMI with AWS CloudFormation stack. For AMI-based products with contract pricing, AWS Marketplace bills your customers upfront or by the payment schedule that you define, based on the contract between you and your customer. After that point, they're entitled to use those resources.

To set your pricing, choose one or more contract durations that you offer customers. You can enter different prices for each contract duration. Your options are 1-month, 12-months, 24-month, and 36-month durations. For private offers, you can specify a custom duration in months (up to 60 months).

Choose the category that best describes your product's pricing. The pricing category appears to customers on the AWS Marketplace website. You can choose from **Bandwidth** (GB/s, MB/s), **Data** (GB, MB, TB), **Hosts**, **Requests**, **Tiers**, or **Users**. If none of the predefined categories fit your needs, you can choose the more generic **Units** category.

The offer allows for up to 24 dimensions to be added to it. Each dimension requires the following data:

- **Contracts Category** – For contract products with no consumption-based pricing, you can choose a category which most closely resembles the category of dimension in the contract or choose **Units** if no values resemble the units for the dimension in the contract
- **Contracts Unit** – Choose one of the available values for the units that closely matches your dimensions based on the Category selected.
- **Contracts Dimension Allow Multiple Purchases** – This field is used to indicate whether an offer is a tiered pricing offer or a non-tiered offer:

Tiered offer – Allows the buyer to subscribe to only one of the available dimensions in the offer. Dimensions in a tiered offer don't have the concept of quantities. Signing a contract with a specific dimension essentially indicates that the buyer has chosen the specific feature indicated by that dimension.

Non-tiered offer – Allows the customer to procure more than one dimensions as part of the contract and allows them to procure multiple units of each such dimension.

Setting a value of *true* for this field indicates that the offer is a non-tiered offer. Setting a value of *false* for this field indicates that the offer is a tiered offer.

When using the Product Load Form (PLF) to create the contracts for your AMI product, you must define the following fields for your pricing dimensions:

- **Contracts DimensionX API Name** – The name that should appear in the license generated in the buyer's AWS License Manager account. This name is also used as the value for Name in Entitlement in the CheckoutLicense API call.
- **Contracts DimensionX Display Name** – The customer-facing name of the dimension that will be displayed on the product detail and procurement pages of the AWS Marketplace website. Create a name that is be user-friendly. The name's maximum length is 24 characters. After the listing is public, the value of Name can't be changed.
- **Contracts DimensionX Description** – The customer-facing description of a dimension that provides additional information about the dimension for the product, such as the capabilities that the specific dimension provides. The maximum length for the description is 70 characters.
- **Contracts DimensionX Quantity** – This is used to calculate proration in cases of agreement amendments to a product. This value of this field should be set to 1 for all contract offers. It should not be edited.
- **Contracts DimensionX 1-Month Rate** – The contract rate to be charged for one month of entitlements against this dimension. For non-tiered offers, this rate is charged for each unit of the dimension that is procured. This field supports three decimal places.
- **Contracts DimensionX 12-Month Rate** – The contract rate to be charged for 12 months of entitlements against the dimension. For non-tiered offers, this rate is charged for each unit of the dimension that is procured. This field supports three decimal places.
- **Contracts DimensionX 24-Month Rate** – The contract rate to be charged for 24 months of entitlements against the dimension. For non-tiered offers, this rate is charged for each unit of the dimension that is procured. This field supports three decimal places.
- **Contracts DimensionX 36-Month Rate** – The contract rate to be charged for 36 months of entitlements against the dimension. For non-tiered offers, this rate is charged for each unit of the dimension that is procured. This field supports three decimal places.

Example: Data storage application

	1-month price	12-month price	24-month price	P36-month price
Unencrypted data (GB)	\$1.50/GB	\$16.00/GB	\$30.00/GB	\$60.00/GB
Encrypted data (GB)	\$1.55/GB	\$16.60/GB	\$31.20/GB	\$61.20/GB

Example: Log monitoring product

	1-month price	12-month price	24-month price	36-month price
Basic (10 hosts monitored, 5 containers monitored)	\$100	\$1000	\$2000	\$4000
Standard (20 hosts monitored , 10 containers monitored)	\$200	\$2000	\$4000	\$8000
Pro (40 hosts monitored, 20 containers monitored)	\$400	\$4000	\$8000	\$16,000
Additional hosts monitored per hour	\$10	\$100	\$200	\$400
Additional containers monitored per hour	\$10	\$100	\$200	\$400

Note

The prices can be for the following durations: 1 month, 12 months, 24 months, or 36 months. You can choose to offer one or more of these options for your product. The durations must be the same across each dimension.

Example

For example, in a case where you have `ReadOnlyUsers` and `AdminUsers` dimensions, if you offer a yearly price for `ReadOnlyUsers`, you must offer a yearly price for `AdminUsers`, too.

Automatic renewals

When customers purchase your product through AWS Marketplace using AMI contracts, they can agree to automatically renew the contract terms. Customers continue to pay for the entitlements every month or for 1, 2, or 3 years.

Customers can modify their renewal settings at any time. For more information, see [Modifying an existing contract](#) in the *AWS Marketplace Buyer Guide*.

AMI product billing, metering, and licensing integrations

The following topics provide information about billing, metering, and licensing integrations for AMI-based products.

Topics

- [Custom metering for AMI products with AWS Marketplace Metering Service](#)
- [Contract pricing for AMI products with AWS License Manager](#)

Custom metering for AMI products with AWS Marketplace Metering Service

The AWS Marketplace Metering Service is a pricing and metering feature that sellers can use to directly charge for their software by usage category. There are five usage categories: users, data, bandwidth, hosts, or unit. You can use the Metering Service with Amazon Machine Image (AMI)-

based, container-based, and software as a service (SaaS)-based products. For more information, see the [AWS Marketplace Metering Service API Reference](#).

All AMI-based software that uses the Metering Service must meet the following requirements:

- Your software must be launched from AWS Marketplace through an Amazon Machine Image (AMI).
- If you have an existing product in AWS Marketplace, you must submit a new AMI and create a new product to enable this feature.
- All software must be provisioned with an AWS Identity and Access Management (IAM) role. The end customer must add an IAM role to the Amazon Elastic Compute Cloud (Amazon EC2) instance the user is provisioning with the software. The use of an IAM role is optional when you deploy software through AWS Marketplace. It's required when you deploy AWS Marketplace Metering Service software.
- Your software must be able to determine consumption in some way.

Products that use the Metering Service must charge customers by a single usage category, but you can define up to 24 dimensions of a single category. Depending on the category, software can be priced by provisioned resources, concurrent resources, or accumulated resource consumption. All charges are still incurred hourly by the customer. All usage is calculated and billed monthly using the same mechanism as existing AWS Marketplace software.

The AWS Marketplace Metering Service enables several new scenarios. For example, if your software monitors hosts, you can charge for each host monitored. You can have different prices based on the host size, and charge for the number of concurrent hosts monitored each hour. Similarly, if your software allows many users across an organization to sign in, you can charge by the number of users. Each hour, the customer is charged for the total number of provisioned users.


Call AWS Marketplace Metering Service

Your software must call the Metering Service hourly and record the consumption value for that hour.

When your software starts, it should record the minute-of-the-hour at which it started. This is referred to as the *start-minute*. Every hour on the start-minute, your software must determine the consumption value for that hour and call the Metering Service. For information about how to obtain this value, see [Modifying your software to use the Metering Service](#).

To wake up each hour at the start-minute, your software must use one of the following approaches:

- A thread within your software.
- A daemon process that starts up with the instance or software.
- A cron job that is configured during application startup.

 **Note**

Your software must call the AWS Marketplace Metering Service using the IAM role configured on the customer's instance and specify the consumption dimension and amount.

Your software can use the AWS SDK to call the AWS Marketplace Metering Service, similar to the following example implementation:

1. Use the instance profile to create a service client. This requires the role configured for the EC2 instance. The role credentials are refreshed by the SDK automatically.
2. Each hour, read your software configuration and state to determine consumption values for that hour. This might include collecting a value-per-dimension.
3. Call the `meterUsage` method on the SDK client with the following parameters (call additionally for each dimension that has usage):
 - `timestamp` – Timestamp of the hour being recorded (in UTC).
 - `productCode` – Product code assigned to the software.
 - `dimension` – Dimension (or dimensions) assigned to the software.
 - `quantity` – Consumption value for the hour.
 - `allocations` – (Optional) You may provide allocations for the usage across properties that you track. These allocations must add up to the total consumption in the record. To the buyer, these display as potential cost allocation tags in their billing tools (such as the AWS Billing and Cost Management console). The buyer must activate the tags in their account in order to track their cost using these tags.

In addition, your software must call an in-Region AWS Marketplace Metering Service endpoint. Your product must have a correct Regional endpoint set up, so `us-east-1` sends records to a `us-`

east-1 endpoint, and us-west-2 sends records to a us-west-2 endpoint. Making in-Region calls provides buyers with a more stable experience and prevents situations in which an unrelated Region's availability could impact software running in another Region.

When you send metering records to the service, you must connect to the AWS Marketplace Metering Service in your Region. Use the `getCurrentRegion()` helper method to determine the Region in which the EC2 instance is running, and then pass this Region information to the `MeteringServiceClient` constructor. If you don't specify an AWS Region in the SDK constructor, the default us-east-1 Region is used. If your application attempts to make cross-Region calls to the service, the calls are rejected. For more information, see [Determining an Application's Current Region](#) and [getCurrentRegion\(\)](#).

Failure handling

Your product must send metering records to the service, a public internet endpoint, so that usage can be captured and billed. Because it's possible for a customer to modify network settings in a way that prevents your metering records from being delivered, your product should account for this by choosing a failure mode.

Note

Some metering failures may be transient issues in connecting to the AWS Marketplace Metering Service. AWS Marketplace strongly recommends implementing retries for up to 30 minutes, with exponential back off, to avoid short-term outages or network issues.

Typically, software can fail open (provide a warning message but maintain full functionality) or fail closed (disable all functionality in the application until a connection has been reestablished). You can choose to fail open, closed, or something specific to your application. We strongly recommend that you refrain from failing closed after less than two hours of metering failures.

As an example of failing partially open, you could continue to allow access to the software but not allow the buyer to modify the software settings. Or, a buyer could still access the software but would not be able to create additional users. Your software is responsible for defining and enforcing this failure mode. Your software's failure mode must be included when your AMI is submitted, and it can't be changed later.

Limitations

Keep these limitations in mind when designing and submitting your Metering Service-enabled software:

- **IAM role and internet gateway requirements for your customers** – Your customers must have an internet gateway and must launch your software with an IAM role with specific permissions. For more information, see [AWS Marketplace metering and entitlement API permissions](#). Your software can't connect to the Metering Service if these two conditions aren't met.
- **Inability to add new or change usage category to existing Metering Service product** – When customers subscribe to your software product, they're agreeing to terms and conditions. Changing the usage categories in products with the Metering Service requires a new product and a new subscription.
- **Inability to change dimensions to existing Metering Service product** – When customers subscribe to your software product, they're agreeing to terms and conditions. Changing the dimensions in products with the Metering Service requires a new product and a new subscription. You *can* add new dimensions to existing products, up to the limit of 24.
- **Lack of free trial and annual subscriptions** – Metering Service products don't support free trials and annual subscriptions at launch.
- **Multi-instance or cluster-based deployment considerations** – Some software is deployed as part of a multi-instance deployment. When you design your software, consider how and where consumption is measured and where metering records are emitted.

Code example

The following code example is provided to help you integrate your AMI product with the AWS Marketplace APIs required for publishing and maintaining your product.

MeterUsage with usage allocation tagging (Optional)

The following code example is relevant for AMI products with consumption pricing models. The Python example sends a metering record with appropriate usage allocation tags to AWS Marketplace to charge your customers for pay-as-you-go fees.

```
# NOTE: Your application will need to aggregate usage for the
#       customer for the hour and set the quantity as seen below.
#       AWS Marketplace can only accept records for up to an hour in the past.
```

```
#
# productCode is supplied after the AWS Marketplace Ops team has
# published the product to limited

# Import AWS Python SDK
import boto3
import time

usageRecord = [
    {
        "AllocatedUsageQuantity": 2,
        "Tags":
            [
                { "Key": "BusinessUnit", "Value": "IT" },
                { "Key": "AccountId", "Value": "123456789" },
            ]
    },
    {
        "AllocatedUsageQuantity": 1,
        "Tags":
            [
                { "Key": "BusinessUnit", "Value": "Finance" },
                { "Key": "AccountId", "Value": "987654321" },
            ]
    }
]

marketplaceClient = boto3.client("meteringmarketplace")

response = marketplaceClient.meter_usage(
    ProductCode="testProduct",
    Timestamp=int(time.time()),
    UsageDimension="Dimension1",
    UsageQuantity=3,
    DryRun=False,
    UsageAllocations=usageRecord
)
```

For more information about MeterUsage, see [MeterUsage](#) in the *AWS Marketplace Metering Service API Reference*.

Example response

```
{ "MeteringRecordId": "string" }
```

Contract pricing for AMI products with AWS License Manager

For Amazon Machine Image (AMI)-based products with contract pricing, you use AWS License Manager to associate licenses with your product.

AWS License Manager is a license management tool that enables your application to track and update licenses (also known as entitlements) that have been purchased by a customer. This section provides information about how to integrate your product with AWS License Manager. After the integration is complete, you can publish your product listing on AWS Marketplace.

For more information about AWS License Manager, refer to the [AWS License Manager User Guide](#) and the [AWS License Manager](#) section of the *AWS CLI Command Reference*.

Note

- Customers can't launch new instances of the AMI after the contract expiry period. However, during the contract duration, they can launch any number of instances. These licenses are not node-locked or tied to particular instances.
- **Private Offer Creation**– Sellers can generate private offers for the products using the Private offer creation tool in the AWS Marketplace Management Portal.
- **Reporting** – You can set up data feeds by setting up an Amazon S3 bucket in the **Report** section in the AWS Marketplace Management Portal. For more information, refer to [Seller reports, data feeds, and dashboards](#).

License models

AWS Marketplace integration with AWS License Manager supports two license models:

- [Configurable license model](#)
- [Tiered license model](#)

Configurable license model

The configurable license model (also known as the quantifiable license model) entitles a buyer to a specific quantity of resources after a buyer has procured a license.

You set a pricing dimension and a per unit price. Then, buyer can choose the quantity of the resources that they want to purchase.

Example of pricing dimension and per unit price

You can set a pricing dimension (such as data backup) and per unit price (such as \$30/unit)

The buyer can choose to purchase 5, 10, or 20 units.

Your product tracks and meters usage to measure the quantity of resources consumed.

With the configuration model, the entitlements are counted in one of two ways:

- [Drawdown licenses](#)
- [Floating licenses](#)

Drawdown licenses

The license is drawn from the pool of allowed amount of licenses upon use. That entitlement is checked out permanently and can't be returned to the license pool.

Example of processing a limited amount of data

A user is entitled to process 500 GB of data. As they continue to process data, the quantity is drawn from the pool of 500 GB until all 500 GB licenses are consumed.

For drawdown licenses, you can use the CheckoutLicense API operation to check out license units that are consumed.

Example of backup to S3 for a number of units/year

You have a storage product that allows backup to Amazon Simple Storage Service (Amazon S3) for up to 1024 units for data for one year. Your application can be launched by using multiple Amazon EC2 instances. Your application has a mechanism to track and aggregate data. Your software calls the CheckoutLicense API operation with the Product ID upon every backup or at fixed intervals to update the consumed quantities.

In this example, your software calls `CheckoutLicense` to check out 10 units of data. When the total capacity reaches the backup limit that the customer has purchased, the API call fails.

Request

```
linux-machine ~]$ aws license-manager checkout-license\  
--product-sku "2205b290-19e6-4c76-9eea-377d6bf71a47" \  
--checkout-type "PERPETUAL" \  
--key-fingerprint "aws:294406891311:AWS/Marketplace:issuer-fingerprint" \  
--entitlements "Name=DataConsumption, Value=10, Unit=Count" \  
--client-token "AKIAIOSFODNN7EXAMPLE"
```

Response

```
{  
  "CheckoutType": "PERPETUAL",  
  "EntitlementsAllowed": [  
    {  
      "Name": "DataConsumption",  
      "Count": 10,  
      "Units": "Count",  
      "Value": "Enabled"  
    }  
  ],  
  "Expiration": "2021-04-22T19:02:36",  
  "IssuedAt": "2021-04-22T18:02:36",  
  "LicenseArn": "arn:aws:license-manager::294406891311:license:l-16bf01b...",  
  "LicenseConsumptionToken": "AKIAIOSFODNN7EXAMPLE"  
}
```

Floating licenses

The license is returned to the pool of the allowed amount of licenses after use.

Example of number of users from a fixed upper limit

A user is entitled to 500 simultaneous users on the application. As users log in and log out, the users are drawn and returned to the pool of 500 users. However, the application can't draw more than 500 users from the pool because 500 simultaneous users is the fixed upper limit.

For floating licenses, you can use the `CheckInLicense` API operation to return the license units to the entitlement pool.

Example of number of concurrent users for one year

Your product is priced based on number of concurrent users. The customer purchases a license for 10 users for one year. The customer launches the software by providing AWS Identity and Access Management (IAM) permissions. When a user logs in, your application calls the CheckoutLicense API operation to reduce the quantity by 1. When the user logs out, the application returns that license to the pool by calling the CheckInLicense API operation. If you don't call CheckInLicense, the license unit will be automatically checked in after 1 hour.

Note

In the following Request, the key-fingerprint isn't a placeholder value but the actual value of the fingerprint with which all licenses will be published.

Request

```
linux-machine ~]$ aws license-manager checkout-license\  
--product-sku "2205b290-19e6-4c76-9eea-377d6bf71a47" \  
--checkout-type "PROVISIONAL" \  
--key-fingerprint "aws:294406891311:AWS/Marketplace:issuer-fingerprint" \  
--entitlements "Name=ReadOnlyUSers, Value=10, Unit=Count" \  
--client-token "AKIAIOSFODNN7EXAMPLE"
```

Response

```
{  
  "CheckoutType": "PROVISIONAL",  
  "EntitlementsAllowed": [  
    {  
      "Name": "ReadOnlyUsers",  
      "Count": 10,  
      "Units": "Count",  
      "Value": "Enabled"  
    }  
  ],  
  "Expiration": "2021-04-22T19:02:36",  
  "IssuedAt": "2021-04-22T18:02:36",  
  "LicenseArn": "arn:aws:license-manager::294406891311:license:l-16bf01b...",  
  "LicenseConsumptionToken": "AKIAIOSFODNN7EXAMPLE"  
}
```


Tiered license model

The tiered license model entitles a buyer to a specific level, or tier, of application features after a buyer has procured a license.

You create tiers for your product, such as Basic, Intermediate, and Premium. The buyer then selects one of the predefined tiers.

The application doesn't need to track or meter usage of the application.

With the tiered license model, the entitlements aren't counted but instead signify a tier of service that was procured by the customer.

If you want to offer bundled features together, we recommend using the tiered license model.

Example of Basic, Intermediate, and Premium tiers

A customer can sign a contract for one of three possible tiers of the software: Basic, Intermediate, or Premium. Each of these tiers has its own pricing. Your software can identify the tier that the customer has signed up for by invoking the CheckoutLicense API operation and specifying all possible tiers in the request.

The response of the request contains the entitlement corresponding to the tier the customer has procured. Based on this information, the software can provision the appropriate customer experience.

Request

```
linux-machine ~]$ aws license-manager checkout-license\  
--product-sku "2205b290-19e6-4c76-9eea-377d6bf71a47" \  
--checkout-type "PROVISIONAL" \  
--key-fingerprint "aws:294406891311:AWS/Marketplace:issuer-fingerprint" \  
--entitlements "Name=BasicTier, Unit=None" "Name=IntermediateTier, Unit=None" \  
"Name=PremiumTier, Unit=None"
```

Response

```
{  
  "CheckoutType": "PROVISIONAL",  
  "EntitlementsAllowed": [  
    {  
      "Name": "IntermediateTier",
```

```
    "Units": "None"
  }
},
"Expiration": "2021-04-22T19:02:36",
"IssuedAt": "2021-04-22T18:02:36",
"LicenseArn": "arn:aws:license-manager::294406891311:license:l-16bf01b...",
"LicenseConsumptionToken": "AKIAIOSFODNN7EXAMPLE"
}
```

Integration workflow

The following steps show the workflow for integrating your AMI product with AWS License Manager:

1. Seller creates a product with AWS License Manager integration.
2. Seller lists the product on AWS Marketplace.
3. Buyer finds the product on AWS Marketplace and purchases it.
4. A license is sent to the buyer in their AWS account.
5. Buyer uses the software by launching the Amazon Elastic Compute Cloud (Amazon EC2) instance, Amazon Elastic Container Service (Amazon ECS) task, or Amazon Elastic Kubernetes Service (Amazon EKS) pod software. The customer deploys by using an IAM role.
6. Software reads the license in the buyer's AWS License Manager account, discovers the entitlements purchased, and provisions the features accordingly.

Note

License Manager doesn't do any tracking or updates; this is done by the seller's application.

License Manager integration prerequisites

Before publishing the product, you must do the following:

1. Create a new AMI product in the AWS Marketplace Management Portal, and make a note of its product code.
2. Fill out the Product Load Form (PLF) with the necessary price information, and return it to us for processing.

3. Use an IAM role for the task or pod running your application with the IAM permissions necessary to call `CheckoutLicense`, `ExtendLicenseConsumption`, and `CheckInLicense`.

The required IAM permissions are detailed in the following IAM policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "license-manager:CheckoutLicense",
        "license-manager:GetLicense",
        "license-manager:CheckInLicense",
        "license-manager:ExtendLicenseConsumption",
        "license-manager:ListReceivedLicenses"
      ],
      "Resource": "*"
    }
  ]
}
```

4. Make a test call to the `RegisterUsage` API operation with a record for all of the pricing dimensions you define.

Integrating an AMI-based product with AWS License Manager

You can integrate your AMI-based product with License Manager by using the [AWS License Manager](#) API. Launch the Amazon EC2 instances by using AWS Marketplace AMI-based products.

Note

Make sure that you have completed the [the section called “License Manager integration prerequisites”](#) before you perform the following procedure.

To integrate your AMI-based product with License Manager

1. Complete the procedure in [the section called “Creating a test license in License Manager”](#). You must create a test license in License Manager for testing your integration.

2. Run the [GetLicense](#) API operation using the license Amazon Resource Name (ARN) that you obtained in step 1. Note the value of the `KeyFingerprint` attribute of the `GetLicense` response for later use.
3. Download and include the latest public AWS SDK in your application.
4. To verify that the buyer is entitled to use a license for your application, run the [CheckoutLicense](#) API operation. Use the entitlements details and the key fingerprint of the test license that you obtained in step 1.

If there are no entitlements found for the license, or the entitlement maximum count is exceeded, the `CheckoutLicense` API operation returns `NoEntitlementsAllowedException`. If the entitlements are valid, or available to use, the `CheckoutLicense` operation returns a successful response with the requested entitlements and their values.

5. (Required for floating entitlements only) Run the [CheckinLicense](#) API operation using the `LicenseConsumptionToken` that was received in the `CheckoutLicense` response. This action releases previously checked-out entitlements back into the pool of available entitlements.
6. After you successfully verify the License Manager integration with the test license that you created in step 1, update the key fingerprint in your code to `aws:294406891311:AWS/Marketplace:issuer-fingerprint`. Now, you're ready to work with licenses issued by AWS Marketplace.

Follow the release process of building the application to an AMI product and then submit the product to AWS Marketplace following the product publishing process.

Creating a test license in License Manager

You use version 2 of the AWS Command Line Interface (AWS CLI) to create a test license in AWS License Manager. This test license is only used for verifying and testing the AWS License Manager integration. After the testing is completed, you can delete the test license. The actual license is generated by AWS Marketplace with a different key fingerprint.

AWS Marketplace supports two types of entitlements in AWS License Manager. However, only one type can be enabled for a product. When you create a license, including a test license, you must specify one of the following types of entitlements:

Tiered entitlements – The tiered license model entitles the customer to certain application features. Customers can't define the quantity of units they want to purchase. However, they can select a single predefined package or tier. Customers can modify the contract later to subscribe to another tier.

Configurable entitlements – The configurable license model grants entitlements to a certain quantity of resources when the customer procures a license. The customer chooses the quantity of units they want to purchase during the subscription process and will be billed based on the unit price. Customers can also subscribe to multiple dimensions.

The required parameters for use in the CheckoutLicense API operation are as follows:

- CheckoutType – The valid values are Perpetual or Provisional:
 - Perpetual – Used when the quantity of entitlements checked out will be exhausted from the pool. Example: Buyer is entitled to process 500 GB of data. As they continue to process data, the quantity is drawn down and exhausted from the pool of 500 GB. Gets the status of a purchased license on whether the license is expired or about to be expired to send a notification to the customer.
 - Provisional – Used for floating license entitlements where entitlements are checked out of the pool and returned back after use. Example: User is entitled to 500 simultaneous users in the application. As users log in and log out, the users are drawn and returned to the pool of 500 users. For more information about floating license entitlements, see [Seller issued licenses in AWS License Manager](#).
- ClientToken – Unique, case-sensitive identifier to ensure the exact result occurs and is the same no matter how many times attempted. We recommend that you use a random universally unique identifier (UUID) for each request.
- Entitlements – List of entitlements to be checked out.
 - For tiered entitlements, provide Name and Unit properties as follows:

```
{  
  
  "Name": "<Entitlement_Name>",  
  
  "Unit": "None"  
  
}
```

- For configurable entitlements, provide Name, Unit, and Value properties as follows:

```
{
  "Name": "<Entitlement_Name>",
  "Unit": "<Entitlement_Unit>",
  "Value": <Desired_Count>{
}
```

- **KeyFingerprint** – Use this key fingerprint to verify that the license is issued by AWS Marketplace. The key fingerprint for licenses issued by AWS Marketplace is as follows:

```
aws:294406891311:AWS/Marketplace:issuer-fingerprint
```

- **Product SKU** – Product ID with a Globally Unique Identifier (GUID) format that is associated with an AWS Marketplace product.

Example of a configurable entitlement

The following is an example of a request that uses the CheckoutLicense API operation to check out a configurable entitlement named PowerUsers.

```
aws license-manager checkout-license \
  product-sku "2205b290-19e6-4c76-9eea-377d6bf71a47" \
  checkout-type "PROVISIONAL" \
  client-token "79464194dca9429698cc774587a603a1" \ "Statement":[
  entitlements "Name=PowerUsers,Value=1,Unit=Count" \
  key-fingerprint "aws:294406891311:AWS/Marketplace:issuer-fingerprint"
```

Example of a tiered entitlement

The following is an example of a request that uses the CheckoutLicense API operation to check out a feature entitlement named EnterpriseEdition.

```
aws license-manager checkout-license \
  --product-sku "2205b290-19e6-4c76-9eea-377d6bf71a47" \
  --checkout-type "PROVISIONAL" \
  --client-token "79464194dca9429698cc774587a603a1" \
  --entitlements "Name=EnterpriseEdition,Unit=None" \
```

```
--key-fingerprint "aws:294406891311:AWS/Marketplace:issuer-fingerprint"
```

To create a test license for your AMI-based product

1. From your local environment with AWS CLI v2 installed, run the following script. The script creates the test license and configures the appropriate product details.

Note

Use a different AWS account than the test AWS account in which you are deploying and testing your software. Licenses can't be created, granted to, and checked out in the same AWS account.

```
#!/bin/bash

# Replace with intended product ID on AWS Marketplace
PRODUCT_ID=<REPLACE-WITH-PRODUCT-ID>

# Replace with license recipient's AWS Account ID
BENEFICIARY_ACCOUNT_ID=<REPLACE-WITH-BENEFICIARY-ACCOUNT-ID>

# Replace with your product's name
PRODUCT_NAME="Test Product"

# Replace with your seller name on AWS Marketplace
SELLER_OF_RECORD="Test Seller"

# Replace with intended license name
LICENSE_NAME="AWSMP Test License"

# Replace the following with desired contract dimensions
# More info here: https://docs.aws.amazon.com/license-manager/latest/APIReference/API\_Entitlement.html
# Example "configurable entitlement"
ENTITLEMENTS='[
  {
    "Name": "ReadOnly",
    "MaxCount": 5,
    "Overage": false,
    "Unit": "Count",
```

```

    "AllowCheckIn": true
  }
]'
# Example "tiered entitlement"
# ENTITLEMENTS='[
#   {
#     "Name": "EnterpriseUsage",
#     "Value": "Enabled",
#     "Unit": "None"
#   }
# ]'

# Format "yyyy-mm-ddTHH:mm:ss.SSSZ"
# This creates a validity period of 10 days starting the current day
# Can be updated to desired dates
VALIDITY_START=$(date +%Y-%m-%dT%H:%M:%S.%SZ)
VALIDITY_END=$(date --date="+10 days" +%Y-%m-%dT%H:%M:%S.%SZ)

# Configuration for consumption of the license as set on Marketplace products
CONSUMPTION_CONFIG='{
  "RenewType": "None",
  "ProvisionalConfiguration": {
    "MaxTimeToLiveInMinutes": 60
  }
}'

# License's home Region
HOME_REGION=us-east-1

# License issuer's name
ISSUER=Self

# Run AWS CLI command to create a license
aws license-manager create-license \
  --license-name "${LICENSE_NAME}" \
  --product-name "${PRODUCT_NAME}" \
  --product-sku "${PRODUCT_ID}" \
  --issuer Name="${ISSUER}" \
  --beneficiary "${BENEFICIARY_ACCOUNT_ID}" \
  --validity 'Begin="'"${VALIDITY_START}"'",End="'"${VALIDITY_END}"'"' \
  --entitlements "${ENTITLEMENTS}" \
  --home-region "${HOME_REGION}" \
  --region "${HOME_REGION}" \
  --consumption-configuration "${CONSUMPTION_CONFIG}" \

```



```
--client-token $(uuidgen)
```

2. Grant the license using the AWS License Manager console. For more information, see [distribute an entitlement](#) in the *License Manager User Guide*.
3. Sign in to the AWS account that acts as a buyer account where you will deploy and test your software. This must be a different AWS account from the AWS account that created and granted the license.
4. Go to the AWS License Manager console to accept and activate the granted licenses. For more information, see [manage your granted licenses](#) in the *License Manager User Guide*.
5. Run the following command in your environment.

```
# The following example uses a key fingerprint that should match the test license
# you created.
# When checking out an actual AWS Marketplace created license, use the following
# fingerprint:
# aws:294406891311:AWS/Marketplace:issuer-fingerprint
aws license-manager checkout-license \
  --product-sku <REPLACE-WITH-PRODUCT-ID> \
  --checkout-type PROVISIONAL \
  --key-fingerprint "aws:<ACCOUNT-ID-WHERE-YOU-CREATED-TEST-LICENSE>:Self:issuer-
fingerprint" \
  --entitlements "Name=ReadOnly,Value=1,Unit=Count" \
  --client-token $(uuidgen)
```

The previous command uses PROVISIONAL as the value for the CheckoutType parameter. If the entitlement uses a drawdown license, use PERPETUAL for the value.

License Manager API calls

To manage the licenses stored in the customer's License Manager account, your software can use the following API calls:

- **GetLicense** – Gets the status of a purchased license on whether the license is expired or about to be expired to send a notification to the customer.
- **CheckoutLicense** – Discovers licenses that the user has purchased. You can also use it to update the license quantity when the user has consumed some quantity of licenses. With CheckoutLicense, you can keep checking out the quantities of the licenses used by the customer. When the customer exhausts all the licenses, this call returns an error. For information

about the suggested cadence to run `CheckoutLicense`, see [the section called “License renewals and upgrades”](#).

- `ExtendLicenseConsumption` – In case of floating dimensions, when the software check outs a license, it will return the license to the pool automatically after 60 minutes. If you want to extend the time the license remains checked out, your software can call `ExtendLicenseConsumption` to extend the license for another 60 minutes.
- `CheckInLicense` – In case of floating dimensions, when you want to return the license to the entitlement pool, use `CheckInLicense`.
- `ListReceivedLicenses` – Lists licenses purchased by the buyer.

License renewals and upgrades

Customers can renew or upgrade their licenses on the AWS Marketplace Management Portal. After they make an additional purchase, AWS Marketplace generates a new version of the license that reflects the new entitlements. Your software reads the new entitlements using the same API calls. You don't have to do anything different in terms of License Manager Integration to handle renewals and upgrades.

Due to license renewals, upgrades, cancellations, and so on, we recommend that your product performs the `CheckoutLicense` API call at a regular cadence while the product is in use. By using the `CheckoutLicense` API operation at a regular cadence, the product can detect changes in entitlements such as upgrades and expiry.

We recommend that you perform the `CheckoutLicense` API call every 15 minutes.

Amazon SNS notifications for AMI products

To receive notifications, you subscribe to the AWS Marketplace Amazon Simple Notification Service (Amazon SNS) topics provided to you during product creation. The topics provide notifications about changes to customers' subscriptions for your products. For example, you can know when customers accept a private offer.

Note

During the product creation process, you'll receive the actual Amazon Resource Name (ARN) to the SNS topic. For example: `arn:aws:sns:us-east-1:123456789012:aws-mp-subscription-notification-PRODUCTCODE`

The following Amazon SNS topic is available for AMI products:

- [Amazon SNS topic: aws-mp-subscription-notification](#) – This topic notifies you when a buyer subscribes or unsubscribes to a product. This notification is available for hourly pricing models, including hourly and hourly with annual.

Amazon SNS topic: aws-mp-subscription-notification

Each message in the aws-mp-subscription-notification topic for the subscribe-success and subscribe-fail action has the following format.

```
{
  "action": "<action-name>",
  "customer-identifier": " X01EXAMPLEX",
  "product-code": "n0123EXAMPLEXXXXXXXXXXXXX",
  "offer-identifier": "offer-abcexample123"
}
```

The *<action-name>* will vary depending on the notification. Possible actions are:

- subscribe-success
- subscribe-fail
- unsubscribe-pending
- unsubscribe-success

The offer-identifier only appears in the notification if the offer is a *private offer*.

Subscribing an Amazon SQS queue to the Amazon SNS topic

We recommend subscribing an Amazon SQS queue to the provided SNS topics. For detailed instructions on creating an SQS queue and subscribing the queue to a topic, see [Subscribing an Amazon SQS queue to an Amazon SNS topic](#) in the *Amazon Simple Notification Service Developer Guide*.

Note

You can only subscribe to AWS Marketplace SNS topics from the AWS account used to sell the products. However, you can forward the messages to a different account. For more

information, see [Sending Amazon SNS messages to an Amazon SQS queue in a different account](#) in the *Amazon Simple Notification Service Developer Guide*.

Polling the SQS queue for notifications

After you subscribe your SQS queue to an SNS topic, the messages are stored in SQS. You must define a service that continually polls the queue, looks for messages, and handles them accordingly.

AMI product checklist

Before submitting your Amazon Machine Image (AMI) product request to AWS Marketplace, review this checklist. Validating this information will help to make sure your submission goes through the publication process smoothly.

Product usage:

- Your AMI must be production-ready.
- Your AMI can't restrict product usage by time or any other measurements.
- Your AMI must be compatible with the 1-Click fulfillment experience.
- Everything required to use the product is in the software, including client applications. Products that require external dependencies, such as software packages or client applications, must follow the [the section called "Product usage policies"](#) which include proper disclosure.
- The default user uses a randomized password, or creating the initial user requires verification that the buyer is authorized to use the instance using a value unique to the instance such as instance ID.

For free or paid products:

- No additional license is required to use the product.
- The buyer doesn't have to provide personally identifiable information (for example, their email address) to use the product.

AMI preparation:

- Your product name and description must match the **Description** field of the AMI product that you're providing.
- Uses Hardware Virtual Machine (HVM) virtualization and 64-bit architecture.
- Doesn't contain any known vulnerabilities, malware, or viruses.
- Buyers have operating system-level administration access to the AMI.
- Run your AMI through AMI Self-Service Scanning.

For Windows AMIs:

- When using Windows Server 2012 and later operating systems, use the latest version of [EC2Launch v2](#).
- If your AMI is built with EC2Config on top of Windows Server 2012 and 2012 R2, use the latest version of [EC2Config](#).
- If you're using EC2Launch v2, ensure you complete the following:
 - In [Amazon EC2Launch settings](#), choose **Random** under **Set administrator account**.
 - In [Amazon EC2Launch settings](#), choose the **check box** under **Start SSM service**.
 - Add **UserData** to the [EC2 v2 task configuration](#).
- If you're using EC2Config, enable the following [settings files](#) within your AMI: Ec2SetPassword, Ec2WindowsActivate, and Ec2HandleUserData.
- Ensure that there are no guest accounts or remote desktop users present.

For Linux AMIs:

- Root login is locked or disabled.
- No authorized keys, default passwords, or other credentials are included.

Product Load Form or **Product** tab:

- All required fields are completed.
- All values are within specified character limits.
- All URLs load without error.
- The product image is at least 110 pixels wide and between a 1:1 and 2:1 ratio.

- Pricing is specified for all enabled instance types (for hourly, hourly-based monthly pricing, and hourly-based annual pricing models).
- Monthly pricing is specified (for hourly-based monthly and monthly pricing models).

AMI-based product requirements

AWS Marketplace maintains the following policies for all Amazon Machine Image (AMI) products and offerings in AWS Marketplace. The policies promote a safe, secure, and trustworthy platform for our customers.

All products and their related metadata are reviewed when they're submitted to ensure that they meet or exceed current AWS Marketplace policies. These policies are reviewed and adjusted to meet evolving security guidelines. AWS Marketplace continuously scans your products to verify that they meet changes to the security guidelines. If products fall out of compliance, AWS Marketplace will contact you to update your AMI product to meet new standards. Likewise, if a newly discovered vulnerability is found to affect the AMI, we will ask you to provide an updated AMI with the relevant updates in place. You must use the [self-service AMI scanning tool](#) before submitting your AMI. This tool helps ensure that the AMI meets AWS Marketplace policies.

Security policies

All AMIs must adhere to the following security policies:

- AMIs must not contain any known vulnerabilities, malware, or viruses as detected by the [self-service AMI scanning tool](#) or AWS Security.
- AMIs must use currently supported operating systems and other software packages. Any version of an AMI with an End-of-Life (EoL) operating system or other software packages will be delisted from the AWS Marketplace. You can build a new AMI with updated packages and publish it as a new version to AWS Marketplace.
- All instance authentication must use key pair access, not password-based authentication, even if the password is generated, reset, or defined by the user at launch. AMIs must not contain passwords, authentication keys, key pairs, security keys, or other credentials for any reason.
- AMIs must not request or use access or secret keys from users to access AWS resources. If your AMI application requires access to the user, it must be achieved through an AWS Identity and Access Management (IAM) role instantiated through AWS CloudFormation, which creates the instance and associates the appropriate role. When single-AMI launch is enabled for products with an AWS CloudFormation delivery method, corresponding usage instructions must include

clear guidance for creating minimally privileged IAM roles. For more information, see [the section called “AMI-based delivery using CloudFormation”](#).

- Linux-based AMIs must not allow SSH password authentication. Disable password authentication via your `sshd_config` file by setting `PasswordAuthentication` to `NO`.

Access policies

There are three categories of access policies: general, Linux-specific, and Windows-specific policies.

General access policies

All AMIs must adhere to the following general access policies:

- AMIs must allow operating system (OS)-level administration capabilities to allow for compliance requirements, vulnerability updates, and log file access. Linux-based AMIs use SSH, and Windows-based AMIs use RDP.
- AMIs must not contain authorized passwords or authorized keys.
- AMIs must not use fixed passwords for administrative access. AMIs must use a randomized password instead. An alternative implementation is to retrieve the instance metadata and use the `instance_id` as the password. The administrator must be prompted for this randomized password before being permitted to set or change their own credentials. For information about retrieving instance metadata, see [Instance Metadata and User Data](#) in the *Amazon EC2 User Guide for Linux Instances*.
- You must not have access to the customer's running instances. The customer has to explicitly enable any outside access, and any accessibility built into the AMI must be off by default.

Linux-specific (or Unix-like) access policies

Linux-based or Unix-like AMIs must adhere to the following access policies, as well as the general access policies:

- AMIs must [disable password-based remote logins](#).
- AMIs must disable remote logins for root.
- AMIs must allow users to gain administrator control to perform root function. For example, allow `sudo` access for Linux-based OS. For other systems, allow full privileged-level access.
- AMIs must log the root activity for an audit trail.

- AMIs must not contain authorized passwords for OS users.
- AMIs must not contain authorized keys.
- AMIs must not have blank or null root passwords.

Windows-specific access policies

Windows-based AMIs must adhere to the following access policies, as well as the general access policies:

- For Windows Server 2016 and later, use EC2Launch.
- For Windows Server 2012 R2 and earlier, use the most recent version of Ec2ConfigService and enable Ec2SetPassword, Ec2WindowsActivate, and Ec2HandleUserData.
- Remove guest accounts and remote desktop users, none of which are allowed.

Customer information policies

All AMIs must adhere to the following customer information policies:

- Software must not collect or export customer data without the customer's knowledge and express consent except as required by BYOL (Bring Your Own License). Applications that collect or export customer data must follow these guidelines:
 - The collection of the customer data must be self-service, automated, and secure. Buyers must not need to wait for sellers to approve to deploy the software.
 - The requirements for customer data must be clearly stated in the description or the usage instructions of the listing. This includes what is collected, the location of where the customer data will be stored, and how it will be used. For example, *This product collects your name and email address. This information is sent to and stored by the <company name>. This information will only be used to contact the buyer in regards to the <product name>.*
 - Payment information must not be collected.

Product usage policies

All AMIs must adhere to the following product usage policies:

- Products must not restrict access to the product or product functionality by time, number of users, or other restrictions. Beta and prerelease products, or products whose sole purpose is

to offer trial or evaluation functionality, are not supported. Developer, Community, and BYOL editions of commercial software are supported, provided an equivalent paid version is also available in AWS Marketplace.

- All AMIs must be compatible with either the Launch from Website experience or AMI-based delivery through AWS CloudFormation. For Launch from Website, the AMI can't require customer or user data at instance creation to function correctly.
- AMIs and their software must be deployable in a self-service manner and must not require additional payment methods or costs. Applications that require external dependencies on deployment must follow these guidelines:
 - The requirement must be disclosed in the description or the usage instructions of the listing. For example, *This product requires an internet connection to deploy properly. The following packages are downloaded on deployment: <list of package>*.
 - Sellers are responsible for the use of and ensuring the availability and security of all external dependencies.
 - If the external dependencies are no longer available, the product must be removed from AWS Marketplace as well.
 - The external dependencies must not require additional payment methods or costs.
- AMIs that require an ongoing connection to external resources not under the direct control of the buyer—for example, external APIs or AWS services managed by the seller or a third party—must follow these guidelines:
 - The requirement must be disclosed in the description or the usage instructions of the listing. For example, *This product requires an ongoing internet connection. The following ongoing external services are required to properly function: <list of resources>*.
 - Sellers are responsible for the use of and ensuring the availability and security of all external resources.
 - If the external resources are no longer available, the product must be removed from AWS Marketplace as well.
 - The external resources must not require additional payment methods or costs and the setup of the connection must be automated.
- Product software and metadata must not contain language that redirects users to other cloud platforms, additional products, or upsell services that aren't available in AWS Marketplace.
- If your product is an add-on to another product or another ISV's product, your product description must indicate that it extends the functionality of the other product and that without it, your product has very limited utility. For example, *This product extends the functionality of*

<product name> and without it, this product has very limited utility. Please note that <product name> might require its own license for full functionality with this listing.

Architecture policies

All AMIs must adhere to the following architecture policies:

- Source AMIs for AWS Marketplace must be provided in the US East (N. Virginia) Region.
- AMIs must use HVM virtualization.
- AMIs must use 64-bit or 64-bit ARM architecture.
- AMIs must be AMIs backed by Amazon Elastic Block Store (Amazon EBS). We don't support AMIs backed by Amazon Simple Storage Service (Amazon S3).
- AMIs must not use encrypted EBS snapshots.
- AMIs must not use encrypted file systems.
- AMIs must be built so that they can run in all AWS Regions and are Region-agnostic. AMIs built differently for different Regions aren't allowed.

AMI product usage instructions

When creating usage instructions for your AMI product, please follow the steps and guidance located in [the section called "AMI and container product usage instructions"](#).

Container-based products

AWS Marketplace supports software products that use Docker containers. Container products consist of delivery options that are a set of container images and deployment templates that go together. You submit at least one delivery option for your product, with up to a maximum of four. For each delivery option, you provide a set of container images, usage instructions, and links to deployment templates for customers to launch that delivery option.

AWS Marketplace buyers see the available delivery options on the published product detail pages that are available to them. After they subscribe to the product and choose their preferred delivery option, buyers see information and instructions for launching and using the product. For Container image delivery options, buyers see links to the available deployment templates and container image URLs. They also receive instructions for how to pull the individual container images. For Helm chart delivery options, buyers will see step-by-step instructions for launching using Helm.

For a walkthrough of the buying experience, you can refer to this video: [Deploying AWS Marketplace Containers on Amazon ECS Clusters](#) (3:34).

You can find, subscribe to, and deploy third-party Kubernetes applications from AWS Marketplace on any Kubernetes cluster in any environment. You can deploy third-party Kubernetes applications on Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), AWS Fargate, and on-premises using Amazon EKS Anywhere (EKS Anywhere). You can also deploy them on self-managed Kubernetes clusters on-premises or in Amazon Elastic Compute Cloud (Amazon EC2).

You can run Free and Bring Your Own License model (BYOL) container products on any Docker-compatible runtime.

Topics

- [Getting help](#)
- [Getting started with container products](#)
- [Container-based product requirements](#)
- [Container products pricing](#)
- [Container product billing, metering, and licensing integrations](#)
- [Amazon SNS notifications for container products](#)

Getting help

For assistance with your container products, contact your business development partner for AWS Marketplace or the [AWS Marketplace Seller Operations](#) team.

Getting started with container products

This topic describes all the steps related to creating, testing, and publishing your first container product for AWS Marketplace.

Topics

- [Prerequisites](#)
- [Creating a container product](#)
- [Product lifecycle](#)
- [Updating a container product \(legacy\)](#)
- [Updating product visibility](#)
- [Updating the allowlist of AWS account IDs](#)
- [Adding a pricing dimension](#)
- [Updating dimension information](#)
- [Updating pricing terms](#)
- [Updating availability by country](#)
- [Updating end-user license agreement](#)
- [Updating the refund policy of a product](#)
- [Creating the product ID and product code for your container product](#)
- [Creating an initial listing](#)
- [Creating or updating pricing details for container products \(legacy\)](#)
- [Integrating AWS Marketplace Metering Service for your container product](#)
- [Integrating AWS License Manager for your container product](#)
- [Adding a new version of your product](#)
- [Testing and releasing your product](#)
- [Updating version information](#)

- [Restricting a version of your Amazon EKS add-on](#)
- [Creating or updating product information for your container product](#)
- [Publishing container products \(legacy\)](#)
- [Container product scans for security issues](#)

Prerequisites

Before you get started, you must complete the following prerequisites:

1. Access and use the [AWS Marketplace Management Portal](#). This is the tool that you use to register as a seller and manage the products that you sell on AWS Marketplace. For more information, see [AWS Marketplace Management Portal](#).
2. Register as a seller, and submit your tax and banking information. For more information, see [Seller registration process](#).
3. Create at least one container in Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), or AWS Fargate. Make sure that you have links for the associated images.
4. Plan how you'll create and integrate your container product in AWS Marketplace.

We recommend that you plan your pricing, entitlement, and metering strategy well in advance of publicly publishing your product.


- For information about the requirements for container-based products, see [Container-based product requirements](#).
- For information about setting the pricing for your product, see [Container products pricing](#).
- For information about custom metering for your paid container-based product, see [Hourly and custom metering with AWS Marketplace Metering Service](#).

Creating a container product

Creating a container product involves the following steps:

1. [Create the product ID and code](#).
2. [the section called "Creating an initial listing"](#).
3. [Add an initial version of your product](#).
4. For paid products, [integrate metering into your product](#).

5. [the section called “Updating product visibility”](#).

 **Note**

For information on the product lifecycle, see [the section called “Product lifecycle”](#).

Product lifecycle

When you create a product in AWS Marketplace, it's initially published with limited visibility so that accounts on the allow list can see it, including the account that created the product. When you're ready, you can publish it to the AWS Marketplace catalog to allow buyers to subscribe and purchase your product.

On the [Server product](#) page, you can view the list of your products. Depending on what stage it is at, the product will have one of the following statuses.

- **Staging** – An incomplete product for which you're still adding information. At the first **Save and exit** from the self-service experience, the successful change request creates an unpublished product with information from the completed steps that you submitted. From this status, you can continue adding information to the product or change already submitted details through change requests.
- **Limited** – A product is complete after it is submitted to the system and passes all validation in the system. Then the product is released to a **Limited** status. At this point, the product has a detail page that's only accessible to your account and whoever you have allowlisted. You can test your product through the detail page. For more information or help, contact the [AWS Marketplace Seller Operations](#) team.
- **Public** – When you're ready to publish the product so that buyers can view and subscribe to the product, you use the **Update visibility** change request. This request initiates a workflow for the AWS Marketplace Seller Operations team to review and audit your product against AWS policies. After the product is approved and the change request is processed, the product is moved from a status of **Limited** to **Public**. For information about AWS guidelines, see [Container-based product requirements](#).
- **Restricted** – If you want to stop new users from subscribing to your product, you can restrict the product by using the **Update visibility** change request. A **Restricted** status means that existing allowlisted users can continue to use the product. However, the product will no longer be visible to the public or be available to new users.

Note

You can update your product when it's in the Staging, Limited, or Public status. For more information, see [Updating a container product](#).

Updating a container product (legacy)

Updating a container product involves the following steps:

1. [Add a new version of your product](#), including:
 - a. Add repositories for your containers.
 - b. Upload the final containers into the repositories.
 - c. Create the first version of the product with your first container images.
2. [Update the product version information](#).
3. [Publish the product for buyers](#).

Updating product visibility

To change which buyers can view your product in AWS Marketplace, you can use **Update visibility**.

To update visibility

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and then sign in to your seller account.
2. Go to the [Server products](#) page, on the **Current server product** tab, select the container-based product that you want to modify.
3. From the **Request changes** dropdown, choose **Update visibility**.

Note

You can request that the product be moved from a **Limited** status to a **Public** status by using this change request. However, the change request must go through an AWS Marketplace Seller Operations team approval process to be moved to **Public**.

4. Choose **Submit** to submit your request for review.

5. Verify that the **Requests** tab shows the **Request status** as **Under review**. When the request completes, the status becomes **Succeeded**.

Updating the allowlist of AWS account IDs

You can change the list of AWS account IDs that can view your product in a limited state. Allow-listed accounts display a Limited badge alongside the product version on the product detail page.

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/> and sign in to your seller account.
2. From the **Server Products** page, select the container product that you want to modify.
3. From the **Request changes** dropdown list, select **Update allowlist**. The current list of accounts that are allowlisted is shown.
4. In the **Allowlisted AWS accounts** field, enter the AWS account IDs and separate them using a comma.
5. Choose **Submit** to submit your request for review.
6. Verify that the **Requests** tab shows the **Request status** as **Under review**. When the request completes, the status will update to **Succeeded** or **Failed**.

Adding a pricing dimension

You can add a pricing dimension to the pricing model that you use for your product billing. For more information about pricing models, see [Container pricing models](#).

Note

Adding a pricing dimension for a contract with consumption pricing (for example, pay-as-you-go pricing for additional usage) isn't available on the AWS Marketplace Management Portal.

You can't change your pricing model between contract, usage, and contract with consumption pricing. Contact the [AWS Marketplace Seller Operations](#) team for assistance.

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/> and sign in to your seller account.

2. From the [Server Products](#) page, select the container product that you want to modify.
3. From the **Request changes** dropdown list, select **Update pricing dimensions**, and then select **Add pricing dimensions**.
4. Depending on the pricing model, you can add contract dimensions or usage dimensions by providing information for **API identifier**, **Display name**, and **Description**.
5. Choose **Next**, and enter your contract dimension pricing.
6. Choose **Submit** to submit your request for review.
7. Verify that the **Requests** tab shows the **Request status** as **Under review**. When the request completes, the status will update to **Succeeded** or **Failed**.

Updating dimension information

You can change the dimensions information for your product. For more information about pricing models, see [Container pricing models](#).

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/> and sign in to your seller account.
2. From the [Server Products](#) page, select the container product that you want to modify.
3. From the **Request changes** dropdown list, select **Update pricing dimensions**, and then select **Update dimension information**.
4. You can add dimension information by providing information for **Display name** and **Description** of the dimension you want to update.
5. Choose **Submit** to submit your request for review.
6. Verify that the **Requests** tab shows the **Request status** as **Under review**. When the request completes, the status will update to **Succeeded** or **Failed**.

Updating pricing terms

You can change the pricing terms for your product. Pricing terms might need updating if you change the countries in which your product is offered.

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/> and sign in to your seller account.
2. From the [Server Products](#) page, select the container product that you want to modify.

3. From the **Request changes** dropdown list, select **Update public offer**, and then select **Update pricing terms**.
4. Current pricing is prepopulated for you to edit. You can delete the current pricing and add your new price. We recommend that you review the prices you're requesting before submitting your request for review.
5. Choose **Submit** to submit your request for review.
6. Verify that the **Requests** tab shows the **Request status** as **Under review**. When the request completes, the status will update to **Succeeded** or **Failed**.

Note

If you increase the price of a dimension using **Update pricing terms**, you won't be able to update pricing for 90 days. A price increase locks the price for 90 days from the day you initiate the change. This price lock is only in effect for a price increase not a price decrease.

Updating availability by country

You can change the countries in which your product can be offered and subscribed to. For more information, see [Countries](#).

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/> and sign in to your seller account.
2. From the **Server Products** page, select the container product that you want to modify.
3. From the **Request changes** dropdown list, select **Update public offer**, and then select **Update availability by country**.
4. Select one of the following options:
 - **All countries** – Available in all supported countries.
 - **All countries with exclusions** – Available in all supported countries except in selected countries.
 - **Allowlisted countries only** – Available for buyers to purchase only in the countries you specify.
5. Choose **Submit** to submit your request for review.

6. Verify that the **Requests** tab shows the **Request status** as **Under review**. When the request completes, the status will update to **Succeeded** or **Failed**.

Updating end-user license agreement

You can update the end-user license agreement (EULA) to use either the [Standard Contract for AWS Marketplace](#) or a custom EULA. Updates made to the EULA take effect for new users subscribing to your product and product renewals.

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/> and sign in to your seller account.
2. From the [Server Products](#) page, select the container product that you want to modify.
3. From the **Request changes** dropdown list, select **Update public offer**, and then select **Update EULA**.
4. Select **Standard Contract for AWS Marketplace** or submit your custom EULA. For a custom EULA, you must provide the contract from an Amazon S3 bucket.
5. Choose **Submit** to submit your request for review.
6. Verify that the **Requests** tab shows the **Request status** as **Under review**. When the request completes, the status will update to **Succeeded** or **Failed**.

Updating the refund policy of a product

You can update the refund policy for your product. Updates to the refund policy take effect for all users. For more information, see [Product refunds in AWS Marketplace](#).

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/> and sign in to your seller account.
2. From the [Server Products](#) page, select the container product that you want to modify.
3. From the **Request changes** dropdown list, select **Update public offer**, and then select **Update refund policy**.
4. Current refund policy is prepopulated for you to edit. You can delete the current refund policy and add your new refund policy. We recommend that you review the refund policy you're requesting before submitting your request for review. Submitting the request overwrites the current refund policy.
5. Choose **Submit** to submit your request for review.

6. Verify that the **Requests** tab shows the **Request status** as **Under review**. When the request completes, the status will update to **Succeeded** or **Failed**.

Creating the product ID and product code for your container product

To get started with a container product, you must create a product ID and product code record in AWS Marketplace. The product ID is used to track your product throughout its lifecycle.

Use the following procedure to create a new container product in the AWS Marketplace Management Portal, and generate the product ID.

Note

This process also creates a public key for your container that pairs with your product.

To create the container product ID

1. Open a web browser and sign into the [AWS Marketplace Management Portal](#).
2. From the menu bar, select **Product**, and choose **Server**.
3. Choose **Create server product** and then choose **Container**.
4. Generate a container product ID and code.

Note

(Optional) You can tag your product for tag-based authorization. For more information, see [Tagging your AWS resources](#).

5. Choose **Continue** to continue creating your product.

Creating an initial listing

After generating the product ID, product code, and public key, you'll use a wizard to create an initial listing.

1. Provide product information for your product listing.
2. Determine the pricing model for your product.

Note

For more information, see [Container products pricing](#).

Note

For paid products, your product will start with \$0.01 pricing to allow you and AWS Marketplace Seller Operations team to test the product without incurring a high cost. You'll provide the actual price when you go public.

3. Provide additional offer information, including a refund policy, EULA, and offer availability.
4. Add an initial repository for your container product.
5. Choose **Submit** on the last step to move the product to Limited visibility.

Note

Your container product is initially created with a placeholder version. You'll add the final version when the product has a Limited visibility.

Creating or updating pricing details for container products (legacy)

To update the pricing details for your container product, you must use a product load form (PLF). The PLF for your product is a spreadsheet that contains information about your product. The following procedure outlines using the PLF to update information about your product, including pricing details.

Note

For more information about pricing models for container products, see [Container products pricing](#).

Your pricing and metering must be aligned. For more information about metering with container products, see [Hourly and custom metering with AWS Marketplace Metering Service](#).

To update pricing for your container product by using the product load form

1. Open a web browser and sign into the [AWS Marketplace Management Portal](#).
2. From the menu bar, expand **Assets**, and choose **File upload**.
3. From **Product load forms and seller guides** on the right side, choose **Containers Product Load Form**.
4. Open the PLF spreadsheet on your computer, and fill out the fields to define your product. This information includes your product ID that you made note of when you created your container product.

Tip

When viewing the PLF in Microsoft Excel, hover over each of the fields to show comments that provide guidance about how to fill in each field.

Provide pricing and metering dimensions, based on your pricing model for your product. For more information, see the following:

- [Product load form for custom metering](#)
- [Product load form for hourly metering](#)

Note

Required fields have a red header with the word **REQUIRED** in the spreadsheet. Make sure that all of these fields are filled out to avoid delays in processing your request.

5. Save your PLF.
6. If it's not still open, open a web browser and sign into the [AWS Marketplace Management Portal](#).
7. From the menu bar, expand **Assets**, and choose **File Upload**.
8. In **Upload File**, browse your computer and choose the PLF you saved for this container product.
9. Provide a brief description for this PLF to help you identify it among the other PLFs you upload.
10. Choose **Upload**. Your uploaded PLF appears in a table at the bottom of the page.

Your pricing details are reviewed and updated manually by the AWS Marketplace Seller Operations team. It typically takes a few business days to complete the update. You can check the status by choosing **Container** from the **Assets** menu in the AWS Marketplace Management Portal. An email message is sent to you when the review of your product pricing details is complete.

Note

Your container product is now created, in a limited state. Your account can view the product for testing and modify it. To make it visible to other test accounts, or when it's ready to be made publicly available, see [Publishing container products \(legacy\)](#).

You can edit your container product pricing by following this same procedure, until you publicly publish the product.

After you create the pricing details for your product, you can add other product details, integrate metering into your product, and create a software version for your product.

Integrating AWS Marketplace Metering Service for your container product

For container-based products with usage pricing, you use the [AWS Marketplace Metering Service](#) for both checking entitlement to use your product and metering usage for billing. You must meter for the pricing model that you created when setting your pricing information. For more information, see [Hourly and custom metering with AWS Marketplace Metering Service](#).

Integrating AWS License Manager for your container product

For container-based products with contract pricing, you use the AWS License Manager to associate licenses with your product.

For more information about integrating with AWS License Manager, see [Contract pricing for Container products with AWS License Manager](#).

Adding a new version of your product

Your product might have several versions over its lifetime. Each version has a set of container images that are specific to that version.

Note

You can't add a version to your product until you have created the product ID and the pricing for your product. For more information about these steps, see [Creating the product ID and product code for your container product](#), and [Creating or updating pricing details for container products \(legacy\)](#).

Creating a version of your product involves the following steps:

Topics

- [Step 1: Adding repositories](#)
- [Step 2: Uploading container images and artifacts to repositories](#)
- [Step 3: Adding a new version to your container product](#)

Your container images and other artifacts for your product are stored in repositories in AWS Marketplace. Typically, you create one repository for each artifact needed, but the repository can store multiple versions of the artifact (with different tags).

Note

All images in your product deployment must use images from the AWS Marketplace repositories.


Step 1: Adding repositories

The following procedure describes how to add any needed repositories in AWS Marketplace.

To add repositories


1. Sign in to the [AWS Marketplace Management Portal](#).
2. Select **Server** from the **Products** menu.
3. On the **Server products** tab, select the product you want to modify, and then choose **Add repositories** from the **Request changes** dropdown.

4. Enter the name for the repository that you want to create. If you want to create more than one new repository, choose **Add new repository** for each additional repository, and give it a unique name.

 **Note**

The repository will have this structure: `<repositoryID>.dkr.ecr.us-east-1.amazonaws.com/<sellerName>/<repositoryName>`. When you add items to the repository (in the following procedure), they will get a tag and have this structure: `<repositoryID>.dkr.ecr.us-east-1.amazonaws.com/<sellerName>/<repositoryName>:<tag>`. The `repositoryID` is an internal ID for AWS Marketplace. The `sellerName` is based on the name you created for your seller account. You define the `repositoryName` in this step. The tag is set when you upload an artifact to the repository.

5. Select **Submit**.

 **Note**

You can have up to 50 repositories per product.

A new request is created and shown on the **Requests** tab. When it's completed, within minutes, you can start adding container images and other artifacts to the repositories you have created.

Step 2: Uploading container images and artifacts to repositories

To upload container images and artifacts to repositories

1. Sign in to the [AWS Marketplace Management Portal](#).
2. From the **Products** menu, choose **Server**.
3. On the **Server products** tab, select the product you want to modify.
4. Choose **Add repositories** from the **Request changes** dropdown.
5. Choose **View existing repositories**.
6. Select the repository to which you want to upload.

7. Select **View push commands** to open a list of instructions, including commands you can use to push Docker container images and Helm charts to that repository.


For general information about how to push container images and other artifacts to repositories, refer to [Pushing an image](#) in the *Amazon Elastic Container Registry User Guide*.

 **Note**

You can use the following Amazon Elastic Container Registry (Amazon ECR) API operations when calling `docker pull` or `docker push`:


- `DescribeImages` – Use this to review the metadata about the images in a repository.
- `GetAuthorizationToken` – Use to authenticate before uploading artifacts to the repository, then use `docker pull` or `docker push` commands.
- `ListImages` – Use to view a list of images you pushed.

8. Use the commands listed to push any needed artifacts from your local repository to the AWS Marketplace repository for your product.

 **Note**

The **tag** that you provide in the push commands is used to differentiate the version of the artifact that you are uploading to the repository. Use a tag that makes sense for the version the artifacts are a part of.

9. Repeat for each container image or artifact you need in your version.

 **Note**

Your version can include up to 50 container images or artifacts in each delivery option. Refer to the following procedure for more information about delivery options.

After you upload your artifacts, you're ready to create the version of your product.

Note

Your container images are scanned automatically to see if they meet the [Container-based product requirements](#). For more information, refer to [Container product scans for security issues](#).

Adding a new delivery option

Each version of your container product would need a delivery option. Delivery option specifies the deployment options available for the buyer. Depending on one of the delivery options below, you would need to upload the appropriate artifacts into the repositories.

- For a **Container image** delivery option, upload all the container images required for the product installation into the Amazon Elastic Container Registry (Amazon ECR) repository created in the AWS Marketplace console.
- For a **Helm chart** delivery option, upload the Helm chart and container images into the Amazon ECR repository created in the AWS Marketplace console.
- For an **Amazon EKS console add-on** delivery option, upload the Helm chart and container images into the Amazon ECR repository created in the AWS Marketplace console.

Step 3: Adding a new version to your container product

Note

If you receive any errors when adding a new version to your container, see the [Add a new version Asynchronous Errors table](#) in the *AWS Marketplace Catalog API Reference*.

To add a new version to your container product

1. Sign in to the [AWS Marketplace Management Portal](#).
2. Choose **Server** from the **Products** menu.
3. On the **Server products** tab, select the product you want to add a version to. Then choose **Add new version** from the **Request changes** dropdown.
4. On the **Add new version** page, enter the **Version title** and **Release notes** for your version.

5. After entering the version details, the next step is to add delivery options. Delivery options are sets of instructions and information that buyers can use to launch the software from your product version. Delivery options are known as *fulfillment options* to buyers.


 **Note**

Your product can support multiple platforms with different container images (for example, Kubernetes and Ubuntu deployments). You can create one delivery option for each way that customers can set up your product, up to four delivery options per version of the product.

- a. If the product already has delivery options in other versions, you can use the existing option as a template to add a delivery option to the new version. In **Delivery options**, choose the delivery option that you want to add from the list. You can edit the option using the instructions in the following steps.
 - b. To add a new delivery option, choose **New delivery option**. After adding an option, follow the instructions in the following steps to configure it.
6. Choose a delivery method for the delivery option. The delivery method determines how buyers will launch your software.
 - For a **Container image** delivery option, provide paths to container images in an Amazon Elastic Container Registry (Amazon ECR) repository that was created in the AWS Marketplace console. Buyers use the container image paths to launch the software by pulling the images directly into their environments.
 - For a **Helm chart** delivery option, provide paths to Helm charts in an Amazon ECR repository that was created in the AWS Marketplace console. Buyers install the Helm charts in their deployment environment to launch the software.
 - For an **Amazon EKS console add-on** delivery option, provide paths to Helm charts in an Amazon ECR repository that was created in the AWS Marketplace console. Buyers install the container using the Amazon EKS console or native Amazon EKS add-on APIs to launch the software. For more information, see [Available Amazon EKS add-ons from Amazon EKS](#).
 - a. To add a **Container image** delivery option, perform the following steps:

- i. In **Container images**, add the Amazon ECR URL to the container images that contain the product version software.
 - ii. In **Delivery option title** and **Deployment option description**, enter a title and description for this delivery option.
 - iii. In **Usage instructions**, enter detailed information to help your buyers use your software after launching it.
 - iv. In **Supported services**, select the environments that buyers can launch the software in.
 - v. In **Deployment templates**, add resources that buyers can use to launch the software. Enter a title and a URL to the resource for each template.
- b. To add a **Helm chart** delivery option, perform the following steps:
- i. In **Helm chart**, add the Amazon ECR URL to the Helm chart that buyers will install in their deployment environment to launch your software.
 - ii. In **Container images**, add the Amazon ECR URL to the container images that contain the product version software.
 - iii. In **Delivery option title** and **Deployment option description**, enter a title and description for this delivery option.
 - iv. In **Usage instructions**, enter detailed information to help your buyers use your software after launching it.
 - v. In **Supported services**, select the environments that buyers can launch the software in.
 - vi. *Optional* - In **Helm release name**, enter the name of the Kubernetes namespace where the Helm chart will be installed.
 - vii. *Optional* - In **Helm installation namespace**, enter the name for the Helm release that will be used by the `helm install` command.
 - viii. *Optional* - In **Kubernetes service account name**, enter the name of the Kubernetes service account that will be used to connect to AWS Identity and Access Management (IAM). The Kubernetes service account calls AWS services such as licensing or metering.
 - ix. Choose to enable **QuickLaunch** on this product version. QuickLaunch is a feature in AWS Marketplace. Buyers can use QuickLaunch to create an Amazon EKS cluster quickly and launch your software on it by using AWS CloudFormation. For more information, see [QuickLaunch in AWS Marketplace](#).

- x. In **Override parameters**, enter parameters that will be used in the Helm CLI commands that launch the software. These parameters allow buyers to override the provided default values. If you have enabled QuickLaunch, also enter a parameter name and description for the CloudFormation form. There is a limit of 15 parameters when using the AWS Marketplace Management Console, but there is no limit when using the AWS Marketplace Catalog API. For more information, see [Adding a new version to a container-based product](#).

 **Note**

Some **Override parameters** are required. Amazon EKS Anywhere products require an **Override parameter** for license secret with a `DefaultValue` of `"${AWSMP_LICENSE_SECRET}"`. For paid products, you must provide one **Override parameter** for service account configuration with the `DefaultValue` of `"${AWSMP_SERVICE_ACCOUNT}"`.

- xi. Choose **Hide passwords and secrets** to mask sensitive information in consoles, command line tools, and APIs. For more information, see the `NoEcho` parameter documentation in [Parameters](#) in the *AWS CloudFormation User Guide*.
- c. To add an **Amazon EKS console add-on** delivery option, make sure that artifacts conform to [Requirements for Amazon EKS add-on products](#), and then perform the following steps:

 **Note**

Only one Amazon EKS add-on delivery option is supported per version. You aren't able to add a new version until the current version you're working with is published on the Amazon EKS console.

- i. In **Helm chart**, add the Amazon ECR URL to the Helm chart that buyers will install in their deployment environment to launch your software.
- ii. In **Container images**, add the Amazon ECR URL to the container images that contain the product version software. Make sure that all images within the Helm chart are listed.
- iii. In **Delivery option title** and **Deployment option description**, enter a title and description for this delivery option.

- iv. In **Visibility**, keep the default value of **Limited selected**.
 - v. In **Add-on name**, enter a unique name for this add-on. The add-on name that you enter will be appended with the seller's name while being displayed in the Amazon EKS console.
 - vi. In **Add-on version**, enter the version of the add-on that will be visible when installing or upgrading this add-on. Follow the format `major.minor.patch`.
 - vii. In **Add-on type**, select a category for your add-on from the dropdown list.
 - viii. In **Kubernetes Version**, select all the Kubernetes versions that your add-on will support.
 - ix. In **Architecture**, select the platform architectures that your add-on supports. The options are **AMD64** and **ARM64**. We recommend supporting both architectures to maximize compatibility. If your add-on doesn't support ARM64 devices, you must specify a planned date for adding support before your product can be published in all commercial AWS Regions.
 - x. In **Namespace**, enter a unique Kubernetes namespace where your add-on will be installed. The `default`, `kube-system`, and `kube-public` namespaces aren't supported for installing third-party add-ons.
 - xi. In **Environment Override parameters**, you can select up to 2 environment parameters from the Amazon EKS add-on framework. You can map parameter names from your `values.yaml` to these environment variables, which are `${AWS_REGION}` and `${AWS_EKS_CLUSTER_NAME}`.
7. To add additional delivery options, choose **New delivery option** and repeat the instructions in the previous steps to configure them.
 8. Choose **Submit**.

Testing and releasing your product

This section provides guidance on the next steps after publishing a version for your container product. It outlines the specific steps and process required for testing and releasing your product to public.

Container image and Helm chart delivery options

This section provides guidance on the releasing your Container image and Helm chart.

Your request for a new version is created and should complete within minutes. You can track the request from the **Requests** tab of the **Server products** page. If you receive any errors when testing or releasing your add-on, see the Asynchronous Errors table in [Add a new version](#) in the *AWS Marketplace Catalog API Reference*.

Note

If your product is currently set to limited availability, only the buyers that the product is available for can access the product version. If your product is currently set to public availability, all AWS Marketplace buyers can access the product version.

If this was your first version set, your product is now ready to be published. For information about how to publish a product, see [Publishing container products \(legacy\)](#).

Amazon EKS add-on delivery option

This section provides guidance on testing and releasing your Amazon EKS add-on.

Test your add-on

- After you submit your add-on, AWS Marketplace processes your request and publishes your add-on in a limited state for you to validate in the Amazon EKS add-on catalog. You can track the request from the **Requests** tab of the **Server products** page in the AWS Marketplace Management Portal. Ingestion times will vary from 5-10 business days depending on the volume of requests we are handling.

When your request is in **Under review** status, the add-on is still being published by AWS team from AWS Marketplace into Amazon EKS add-on catalog. Request status changes to **Success** once the add-on is published onto **Limited** state. You can start the testing of your add-on after this.

- After your add-on is available, you can find it in the Asia Pacific (Seoul) Region for testing purposes. AWS Marketplace relies on your expertise to verify the functionality of your software. To test your add-on, you must create an Amazon EKS cluster in the Asia Pacific (Seoul) Region in your seller account where your add-on is allowlisted. To test your add-on, follow [these detailed instructions](#). Make sure to test on each Kubernetes version that your software supports.
- If you're offering a paid product, create a private offer to the following internal AWS accounts. These accounts help integrate your software into the Amazon EKS console in all commercial AWS Regions.

288092140294, 288092140294, 408202761791

- Keep your test cluster with the add-on active until AWS Marketplace approves and moves your add-on version to public.

Note

AWS Marketplace will not bear the AWS infrastructure costs incurred during testing of your container product on your Amazon EKS clusters. You can follow right sizing mechanisms to tone down the nodes to a minimal operating costs while we verify the testing results.

Release your add-on to public

After you have validated your software via Amazon EKS cluster as an add-on, you can submit a request to release the version of your Amazon EKS add-on to public using the [AWS Marketplace Management Portal](#) or AWS Marketplace Catalog API.

For more information, see [Update the visibility for an Amazon EKS add-on](#) in the *AWS Marketplace Catalog API Reference*.

You can track the request from the **Requests** tab of the **Server products** page in the AWS Marketplace Management Portal. Ingestion times will vary.

Updating version information

After a version is created, it can be helpful to provide updated information to your buyers by modifying the information associated with the version. For example, if you plan to restrict version 1.0 after version 1.1 is released, you can update the description of version 1.0 to direct buyers to version 1.1. Provide the date that version 1.0 will be restricted. You update the version information from the AWS Marketplace Management Portal.

To update version information

1. Sign in to the [AWS Marketplace Management Portal](#).
2. Select **Server** from the **Products** menu.
3. On the **Server products** tab, select the product that you want to modify.

4. From the **Request changes** dropdown, choose **Update version information**.
5. On the **Update version** page, select the version that you want to update.
6. Make updates to the selected version. The fields that are available for updating depend on the status of the product version or delivery option.
 - a. For all versions, you can update the **Release notes**.
 - b. For versions that are not yet publicly available, you can update the **Version title**.
 - c. For delivery options that haven't been restricted, you can update the following fields:
 - **Description**
 - **Usage instructions**
 - **Supported services**
 - d. For delivery options in versions that are not yet publicly available, you can update the following fields:
 - **Delivery option titles**
 - **Helm chart** (for **Helm chart** delivery options only)
 - **Container images**
 - **Deployment resources**
 - **AddOn Name**
 - **AddOn Version**
 - **AddOn Type**
 - **Helm Chart URI**
 - **CompatibleKubernetesVersions**
 - **SupportedArchitectures**
 - **Namespace**
 - **EnvironmentOverrideParameters**
 - e. For delivery options in versions that are publicly available, you can update **SupportedArchitectures**.
7. Choose **Submit**.
8. Verify that the request appears on the **Requests** tab with the **Under review** status.

You can check the status of your request at any time from the **Requests** tab of the [Server Products](#) page.

Restricting a version of your Amazon EKS add-on

To restrict a version of your container product published as an add-on, contact the AWS Marketplace operations team using the contact us form at the bottom of the [AWS Marketplace Management Portal](#).

Creating or updating product information for your container product

After you have created your product ID and set the pricing, you can edit your product information, including what customers will see about your container product in the AWS Marketplace. The following procedure outlines creating the product details for your product.

To create or update product details for your container product

1. Sign in to the [AWS Marketplace Management Portal](#).
2. Select **Server** from the **Products** menu.
3. On the **Server products** tab, select the product that you want to modify.
4. From the **Request changes** dropdown, choose **Update product information**.
5. Update any of the following fields that you want to change:
 - **Product title**
 - **SKU**
 - **Short description**
 - **Long description**
 - **Product logo image URL**
 - **Highlights**
 - **Product categories**
 - **Keywords**
 - **Product video URL**
 - **Resources**
 - **Support information**

Note

Image URLs must be in an Amazon S3 bucket that is publicly accessible. For more details about the logo format, see [Company and product logo requirements](#).

6. Choose **Submit**.
7. Verify that the request appears on the **Requests** tab with the **Under review** status. You might need to refresh the page to see the request on the list.

You can check the status of your request at any time from the **Requests** tab of the [Server Products](#) page.

Publishing container products (legacy)

When you initially create your product, its availability is limited to just your account. Once your product is ready for testing (including having product details filled in and the first version created), you can make it available to other accounts for testing, or to all accounts as a public product.

Note

Before publishing publicly, you should review your product to ensure accuracy, including image links, deployment templates, descriptions, and pricing. Your pricing model can't be changed for publicly published products.

To publish your limited product to additional accounts or for public availability, contact the [AWS Marketplace Seller Operations](#) team. In your request, provide the product ID and describe the changes that you want to make.

Note

You can also choose to restrict a version that you no longer want available to buyers. You can include this in a request to publish a product publicly, to avoid test versions appearing in public products.

You can't restrict a version if it will leave your public product with no public versions.

When you publicly publish a container product, you make it visible to all AWS customers who can then subscribe and launch your product. The AWS Marketplace Seller Operations team reviews the data in your product information, as well as your test calls to the AWS Marketplace Metering Service.

Container product scans for security issues

When you create a change request to add a new version to your container product, we scan the container images included in that new version and check for security vulnerabilities. To do this, we perform a layer-by-layer static scan on the image. If we find critical vulnerabilities with remotely exploitable risk vectors, we provide you with a list of found issues. We strongly recommend that you perform your own security analysis using a container image scanner such as Clair, Twistlock, Aqua Security, or Trend Micro to avoid delays in the ingestion and publishing process.

Your choice of base image for building your container images can have a significant influence on the security profile of the final image. If you choose a base image that already has known critical vulnerabilities, they will be flagged because of the base layer, even if your application software layers are clean. We recommend that you verify that you're starting with a base container that is free of vulnerabilities before you build your images and submit them to AWS Marketplace.

Container-based product requirements

AWS Marketplace maintains the following requirements for all container-based products and offerings on AWS Marketplace. These requirements help to promote a safe, secure, and trustworthy catalog for our customers. We also encourage sellers to review implementation of additional controls and protocols as applicable to meet the needs of their specific products.

All products and their related metadata are reviewed when submitted to ensure that they meet or exceed current AWS Marketplace requirements. We review and adjust these policies to meet our evolving security and other usage requirements. AWS Marketplace continuously verifies that existing products continue to meet any changes to these requirements. If products fall out of compliance, AWS Marketplace will contact you to update your product. In some cases, your product might temporarily be unavailable to new subscribers until issues are resolved.

Security requirements

All container-based products must adhere to the following security requirements:

- Docker container images must be free from any known malware, viruses, or vulnerabilities. When you [add a new version](#) to your container product, the container images included in the version are scanned.
- If your container-based products requires access to manage AWS resources, access must be achieved through [IAM roles for service accounts](#) (if run through Amazon Elastic Kubernetes Service (Amazon EKS)) or [IAM roles for tasks](#) (if run through Amazon Elastic Container Service (Amazon ECS)) instead of requesting an access key from users.
- Container-based products must only require least privileges to run. For more information, see [ECS security](#) and [EKS security](#).
- Container images should be configured to run with non-root privileges by default.

Access requirements

All container-based products must adhere to the following access requirements:

- Container-based products must use an initial randomized password. Container-based products must not use initial fixed or blank passwords for external administrative access (for example, to log in to the application via a web interface). The buyer must be prompted for this randomized password before being permitted to set or change their own credentials.
- Any outside access to the application must be explicitly agreed to and enabled by customers.

Customer information requirements

All container-based products must adhere to the following customer information requirements:

- Software must not collect or export customer data without the customer's knowledge and express consent except as required by BYOL (Bring Your Own License). Applications that collect or export customer data must follow these guidelines:
 - The collection of the customer data must be self-service, automated, and secure. Buyers must not need to wait for sellers to approve to deploy the software.
 - The requirements for customer data must be clearly stated in the description or the usage instructions of the listing. This includes what is collected, the location of where the customer data will be stored, and how it will be used. For example, *This product collects your name and email address. This information is sent to and stored by the <company name>. This information will only be used to contact the buyer in regards to the <product name>.*

- Payment information must not be collected.

Product usage requirements

All container-based products must adhere to the following product usage requirements:

- Sellers can only list fully functioning products. Beta or prerelease products for trial or evaluation purposes are not allowed. Developer, community, and BYOL editions of commercial software are supported if the seller provides an equivalent paid version on AWS Marketplace within 90 days of providing the free edition.
- All of a container-based product's usage instructions must include all steps to deploy container-based products. Usage instructions must provide commands and deployment resources pointing to the corresponding container images on AWS Marketplace.
- Container-based products must include all container images that a subscriber needs to use the software. In addition, container-based products must not require a user to launch the product using any images from outside AWS Marketplace (for example, container images from third-party repositories).
- Containers and their software must be deployable in a self-service manner and must not require additional payment methods or costs. Applications that require external dependencies on deployment must follow these guidelines:
 - The requirement must be disclosed in the description or the usage instructions of the listing. For example, *This product requires an internet connection to deploy properly. The following packages are downloaded on deployment: <list of package>*.
 - Sellers are responsible for the use of and ensuring the availability and security of all external dependencies.
 - If the external dependencies are no longer available, the product must be removed from AWS Marketplace as well.
 - The external dependencies must not require additional payment methods or costs.
- Containers that require an ongoing connection to external resources not under the direct control of the buyer—for example, external APIs or AWS services managed by the seller or a third party—must follow these guidelines:
 - The requirement must be disclosed in the description or the usage instructions of the listing. For example, *This product requires an ongoing internet connection. The following ongoing external services are required to properly function: <list of resources>*.

- Sellers are responsible for the use of and ensuring the availability and security of all external resources.
- If the external resources are no longer available, the product must be removed from AWS Marketplace as well.
- The external resources must not require additional payment methods or costs and the setup of the connection must be automated.
- Product software and metadata must not contain language that redirects users to other cloud platforms, additional products, or upsell services that aren't available on AWS Marketplace.
- If your product is an add-on to another product or another ISV's product, your product description must indicate that it extends the functionality of the other product and that without it, your product has very limited utility. For example, *This product extends the functionality of <product name> and without it, this product has very limited utility. Please note that <product name> might require its own license for full functionality with this listing.*

Architecture requirements

All container-based products must adhere to the following architecture requirements:

- Source container images for AWS Marketplace must be pushed to the Amazon Elastic Container Registry (Amazon ECR) repository owned by AWS Marketplace. You can create these repositories in the AWS Marketplace Management Portal under server products for each of your container product listings.
- Container images must be based on Linux.
- Paid container-based products must be able to be deployed on [Amazon ECS](#), [Amazon EKS](#), or [AWS Fargate](#).
- Paid container-based products with contract pricing and an integration with AWS License Manager should deploy on Amazon EKS, Amazon ECS, AWS Fargate, Amazon EKS Anywhere, Amazon ECS Anywhere, Red Hat OpenShift Service on AWS (ROSA), self-managed Kubernetes clusters on-premises, or on Amazon Elastic Compute Cloud.

Container product usage instructions

When creating usage instructions for your container product, follow the steps and guidance in [the section called "AMI and container product usage instructions"](#).

Requirements for Amazon EKS add-on products

An Amazon EKS add-on is software that provides operational capabilities to Kubernetes applications but isn't specific to the application. For example, an Amazon EKS add-on includes observability agents or Kubernetes drivers that allow the cluster to interact with underlying AWS resources for networking, compute, and storage.

As a seller of container products, you can choose among several deployment options including Amazon EKS. You can publish a version of your product as an AWS Marketplace add-on into the Amazon EKS add-on catalog. Your add-on appears in the Amazon EKS console next to add-ons maintained by AWS and other vendors. Your buyers can deploy your software as an add-on just as easily as they do the other add-ons.

For more information, see [Amazon EKS add-ons](#) in the *Amazon EKS User Guide*.

Preparing your container product as an AWS Marketplace add-on

To publish your container product as an AWS Marketplace add-on, it must meet the following requirements:

- Your container product must be published in AWS Marketplace.
- Your container product must be built compatible for both AMD64 and ARM64 architectures.
- Your container product must not use the Bring Your Own License (BYOL) [pricing model](#).

Note

BYOL is not supported for Amazon EKS add-on delivery.

- You must adhere to all [container-based product requirements](#) including pushing all container images and Helm charts into AWS Marketplace managed Amazon ECR repositories. This requirement includes open-source images, for example, `nginx`. Images and charts can't be hosted in other external repositories including, but not limited to, [Amazon ECR Public Gallery](#), Docker Hub, and Quay.
- **Helm charts** - Prepare your software to be deployed through a Helm chart. The Amazon EKS add-on framework converts a Helm chart into a manifest. Some Helm features are not supported within Amazon EKS systems. The following list describes the requirements that must be met before onboarding. In this list, all Helm commands use Helm version 3.8.1:

- All Capabilities objects are supported, with an exception for .APIVersions. .APIVersions is not supported for non-built-in custom Kubernetes APIs.
- Only the Release.Name and Release.Namespace objects are supported.
- Helm hooks and the lookup function are not supported.
- All dependent charts must be located within the main Helm chart (specified with repository path file://...).
- The Helm chart must successfully pass Helm Lint and Helm Template with no errors. The commands are as follows:

- Helm Lint – `helm lint helm-chart`

Common issues include undeclared charts in the parent chart's metadata. For example, chart metadata is missing these dependencies: chart-base Error: 1 chart(s) linted, 1 chart(s) failed

- Helm Template – `helm template chart-name chart-location --set k8version=Kubernetes-version --kube-version Kubernetes-version --namespace addon-namespace --include-crds --no-hooks -f any-overridden-values`

Pass any overridden configurations with the `--f` flag.

- Store all container binaries in AWS Marketplace Amazon ECR repos. To create a manifest, use the Helm template command that's shown earlier. Search the manifest for any external image references such as busybox or gcr images. Upload all container images along with dependencies into AWS Marketplace Amazon ECR repos created by using the **Add Repository** option in the request dropdown.
- **Custom configuration** – You can add custom variables during the deployment. For information about how to identify the end user experience, name the software `aws_mp_configuration_schema.json`, and package into a wrapper with the Helm chart, see [Amazon EKS add-ons: Advanced configuration](#).

According to [The "\\$schema" Keyword](#), `$schema` must be a URI that points to a valid `application/schema+json` resource.

This file must not accept any sensitive information such as passwords, license keys, and certificates.

To handle secrets and certificate installations, you can provide post- or pre-Add-on installation steps to end users. The product should not rely on any external licenses. The product should work based on AWS Marketplace entitlements.

For more information about limitations for `aws_mp_configuration_schema.json`, see [Add-on configuration requirements and best practices for add-on providers](#).

- **Identify and create the namespace that the software will be deployed in** – In the first release of your product, you must identify the namespace that the software will be deployed in by adding a templated namespace.
- **Create the serviceAccount if applicable** – If the software is either a paid software on AWS Marketplace or must connect with other AWS services, make sure that the Helm chart creates `serviceAccount` by default. If the `serviceAccount` creation is handled by a parameter in a `values.yaml` file, set the parameter value to `true`. For example, `serviceAccount.create = true`. This is required because the customer might choose to install the add-on by inheriting permissions from the underlying node instance which already has the required permissions. If the Helm chart doesn't create the `serviceAccount`, then the permissions can't be tied to the `serviceAccount`.
- **Traceable Deployments or Daemonsets** – Make sure your Helm chart has a daemonset or deployment. Amazon EKS addon framework tracks deployment of your Amazon EKS resources using them. Without a traceable deployment or daemonset, your add-on will face a deployment error. If your add-on does not have a deployment or daemonset, for example, if your add-on deploys a bunch of Custom resources or a Kubernetes job which are not traceable, add a dummy deployment or daemonset object.
- **Support for AMD and ARM architectures** – Many Amazon EKS customers use ARM64 today to use AWS Graviton instances. Third-party software must support both architectures.
- **Integrate with licensing or metering APIs from AWS Marketplace** – AWS Marketplace supports multiple billing models. For more information, see [Container product billing, metering, and licensing integrations](#). If you want to sell your product through PAYG mechanisms, see [Custom metering for container products with AWS Marketplace Metering Service](#). If you want to sell your product through an upfront or contract model, see [Contract pricing for Container products with AWS License Manager](#).
- **Upload the software and all the artifacts and dependencies** – The Helm chart must be self-contained, and it must not require dependencies from external sources, for example, GitHub. If the software requires external dependencies, then the dependencies must be pushed to AWS Marketplace private Amazon ECR repositories under the same AWS Marketplace listing.

- **Provide deployment instructions on your website** – We request that you host a deployment guide for customers to identify how to deploy your software through the [create-addon](#) command.
- **IAM roles** – List all the AWS Identity and Access Management (IAM) policies required for your software to function or connect with other AWS services.
- **Version updates** – Amazon EKS releases new Kubernetes versions a few weeks after the upstream release. As new Amazon EKS cluster versions become generally available, vendors have 45 days to certify or update their software to be compatible with the new Amazon EKS cluster version release. If your current versions of the add-on supports the new Kubernetes version, validate and certify the same so that we can update the version compatibility matrix. If a new add-on version is needed to support the new Kubernetes version release, then please submit the new version for onboarding.
- Partner's software must fall into one of the following types or be an operational software that will enhance Kubernetes or Amazon EKS: Gitops | monitoring | logging | cert-management | policy-management | cost-management | autoscaling | storage | kubernetes-management | service-mesh | etcd-backup | ingress-service-type | load-balancer | local-registry| networking | Security | backup | ingress-controller | observability
- Software cannot be [Container Network Interface \(CNI\)](#).
- Software must be sold through AWS Marketplace and integrated with Licensing and metering APIs for paid products. BYOL products are not accepted.

Add-on configuration requirements and best practices for add-on providers

Amazon EKS requires configuration as a [Helm JSON schema](#) string from add-on providers. Add-ons that either need required configurations or allow optional configurations must include a `aws_mp_configuration_schema.json` file with the Helm Chart submitted to AWS Marketplace. Amazon EKS will use this schema to validate the configuration input from customers and reject API calls with input values that do not conform to the schema. Add-on configurations typically fall under two categories:

- Configuration for general Kubernetes properties like labels, tolerations, nodeSelector, etc.
- Configurations that are add-on specific like license key, feature enablement, URLs, etc.

This section is focused on the first category related to general Kubernetes properties.

Amazon EKS recommends following best practices around configuration of Amazon EKS add-ons.

- [Schema requirements](#)
- [Common parameters that are allowed for configuration](#)
- [Common parameters that aren't allowed for configuration](#)

Schema requirements

When defining the json schema, ensure you use a version of jsonschema that is supported by Amazon EKS add-ons.

The list of supported schemas:

- <https://json-schema.org/draft-04/schema>
- <https://json-schema.org/draft-06/schema>
- <https://json-schema.org/draft-07/schema>
- <https://json-schema.org/draft/2019-09/schema>

Using any other json schema version is incompatible with Amazon EKS add-ons and will cause the add-on to be unable to be released until this is fixed.

Example Helm schema file

```
{
  "$schema": "http://json-schema.org/schema#",
  "type": "object",
  "properties": {
    "podAnnotations": {
      "description": "Pod Annotations"
      "type": "object"
    },
    "podLabels": {
      "description": "Pod Labels"
      "type": "string"
    },
    "resources": {
      "type": "object"
      "description": "Resources"
    },
    "logLevel": {
      "description": "Logging Level"
    }
  }
}
```

```
"type": "string",
  "enum": [
    "info",
    "debug"
  ],
  "config": {
    "description": "Custom Configuration"
  }
}
```

camelCase

Configuration parameters are required to be camelCase, and will be rejected if not adhering to this format.

Descriptions are required

Always include meaningful descriptions for schema properties. This description will be used to render label names in Amazon EKS console for each configuration parameter.

RBAC definition

Add-on providers need to define and supply the RBAC permissions needed to successfully install the add-on using the principle of least privilege. If RBAC permissions need to change for newer versions of add-on or any fixes to address a CVE, add-on providers will need to inform the Amazon EKS team about this change. Required permissions for each Kubernetes resource should be restricted to the resource name of the object.

```
apiGroups: ["apps"]
resources: ["daemonsets"]
resourceNames: ["ebs-csi-node"]
verbs: ["create", "delete", "get", "list", "patch", "update", "watch"]
```

Secrets Management

This section only applies to add-ons that need customers to configure secret information like application key, API key, password, etc. Currently, Amazon EKS APIs do not support passing in secret information in plain text due to the security implications. However, customers can use configuration to pass in the name of the Kubernetes Secret that holds the keys needed by the

add-on. Customers will be required to create Kubernetes Secret objects containing the keys with the same namespace as a pre-requisite step and then pass in the name of the Secret using configuration blob when creating the add-on. We recommend that add-on providers name the schema properties so that customers do not accidentally mistake it for the actual key. For example: `appSecretName`, `connectionSecretName` etc.

In summary, add-on providers can leverage the schema to allow customers to pass in the name of the secret but not the keys which will actually hold the secret itself.

Example configuration values

You can include configuration examples in your schema to help customers with configuration of add-ons. The following example is from the schema of AWS Distro for OpenTelemetry add-on.

```
"examples": [  
  {  
    "admissionWebhooks": {  
      "namespaceSelector": {},  
      "objectSelector": {}  
    },  
    "affinity": {},  
    "collector": {  
      "amp": {  
        "enabled": true,  
        "remoteWriteEndpoint": "https://aps-workspaces.us-west-2.amazonaws.com/  
workspaces/ws-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx/api/v1/remote_write"  
      },  
      "cloudwatch": {  
        "enabled": true  
      },  
      "mode": "deployment",  
      "replicas": 1,  
      "resources": {  
        "limits": {  
          "cpu": "256m",  
          "memory": "512Mi"  
        },  
        "requests": {  
          "cpu": "64m",  
          "memory": "128Mi"  
        }  
      },  
      "serviceAccount": {
```

```
    "annotations": {},
    "create": true,
    "name": "adot-collector"
  },
  "xray": {
    "enabled": true
  }
},
"kubeRBACProxy": {
  "enabled": true,
  "resources": {
    "limits": {
      "cpu": "500m",
      "memory": "128Mi"
    },
    "requests": {
      "cpu": "5m",
      "memory": "64Mi"
    }
  }
},
"manager": {
  "env": {},
  "resources": {
    "limits": {
      "cpu": "100m",
      "memory": "128Mi"
    },
    "requests": {
      "cpu": "100m",
      "memory": "64Mi"
    }
  }
},
"nodeSelector": {},
"replicaCount": 1,
"tolerations": []
}
]
```

Common parameters that are allowed for configuration

The following are recommended parameters in a customer facing Helm schema file.

Parameter	Description	Should have a default?
additionalLabels	Add Kubernetes labels to all Kubernetes objects managed by the add-on.	No
additionalAnnotations	Add Kubernetes annotations to all Kubernetes objects managed by the add-on.	No
podLabels	Add Kubernetes labels to pods managed by the add-on.	No
podAnnotations	Add Kubernetes annotations to pods managed by the add-on.	No
logLevel	Log level for components managed by the add-on.	Yes
nodeSelector	Simplest recommended form of node selection constraint. You can add the nodeSelector field to your Pod specification and specify the node labels you want the target node to have.	Potentially, for example Linux nodes only
tolerations	Tolerations are applied to pods. Tolerations allow the scheduler to schedule pods with matching taints. Tolerations allow scheduling but don't guarantee scheduling.	Maybe, more common with daemonsets
affinity	The affinity feature consists of two types of affinity: Node affinity functions like the nodeSelector field but is more expressive and allows you to specify soft rules, Inter-pod	Maybe

Parameter	Description	Should have a default?
	affinity/anti-affinity allows you to constrain Pods against labels on other Pods.	
topologySpreadConstraints	You can use topology spread constraints to control how Pods are spread across your cluster among failure-domains such as regions, zones, nodes, and other user-defined topology domains. This can help to achieve high availability as well as efficient resource utilization.	Maybe
resource request/limits	Specify how much cpu/memory each container needs. Requests are strongly recommended to be set. Limits are optional.	Yes
replicas	Number of replicas of the pods managed by the add-on. Not applicable for daemonsets.	Yes

Note

For workload scheduling configuration parameters, you may need to separate out top level components in the Schema where necessary. Example, Amazon EBS CSI driver contains two main components, controller and node agent - customers require different node selectors/tolerations for each component.

Note

The default values defined in the JSON schema is purely for user documentation purpose only and does not replace the need to have the rightful default in the `values.yaml` file. If

using the default property, please ensure that the default in `values.yaml` matches that in the schema and the two artifacts (`values.schema.json` and `values.yaml`) remain in sync whenever changes are made to the Helm Chart.

```
"affinity": {
  "default": {
    "affinity": {
      "nodeAffinity": {
        "preferredDuringSchedulingIgnoredDuringExecution": [
          {
            "preference": {
              "matchExpressions": [
                {
                  "key": "eks.amazonaws.com/compute-type",
                  "operator": "NotIn",
                  "values": [
                    "fargate"
                  ]
                }
              ]
            },
            "weight": 1
          }
        ]
      },
      "podAntiAffinity": {
        "preferredDuringSchedulingIgnoredDuringExecution": [
          {
            "podAffinityTerm": {
              "labelSelector": {
                "matchExpressions": [
                  {
                    "key": "app",
                    "operator": "In",
                    "values": [
                      "ebs-csi-controller"
                    ]
                  }
                ]
              },
              "topologyKey": "kubernetes.io/hostname"
            }
          }
        ]
      }
    }
  }
}
```

```
        },
        "weight": 100
      }
    ]
  }
},
"description": "Affinity of the controller pod",
"type": [
  "object",
  "null"
]
```

Common parameters that aren't allowed for configuration

Cluster metadata parameters such `clusterName`, `region`, `vpcId`, `accountId`, and others may be required by various add-ons (for example, Elastic Load Balancing Controller). Any parameter similar to these that is known by the Amazon EKS service will be automatically injected by Amazon EKS add-ons, and not put on the responsibility of the user to specify as a configuration option. These parameters include:

- AWS region
- Amazon EKS cluster name
- VPC ID of the cluster
- Container registry, specifically for build-prod accounts, which is used by networking add-ons
- DNS cluster IP, specifically for `coredns` add-on
- Amazon EKS cluster API endpoint
- IPv4 enabled on cluster
- IPv6 enabled on cluster
- Prefix delegation for IPv6 enabled on cluster

Add-on providers need to ensure you have templating defined for such applicable parameters. Each of the above parameters will have a pre-defined `parameterType` attribute defined by Amazon EKS. The release metadata will specify the mapping between the `parameterType` and the name/path of the parameter in the template. This way, the values can be dynamically passed-in by Amazon EKS without requiring customers to specify these through configurations and also

gives flexibility to add-on providers to define their own template name/path. Parameters such as the above that Amazon EKS needs to inject dynamically should be excluded from the schema file.

Example mapping from release metadata

```
"defaultConfiguration": [
  {
    "key": "image.containerRegistry",
    "parameterType": "CONTAINER_REGISTRY"
  }
]
```

The following are parameters not recommended to be configurable in a customer facing Helm schema file. Either the parameters should have non-modifiable defaults, or not be included at all in the add-on template.

Parameter	Description	Should have a default?
image	Container image that will be deployed on the Kubernetes cluster.	No, managed through add-on definition
imagePullSecrets	Configuring a pod to use a secret to pull from a private registry.	N/A
livenessProbe	The Kubelet process uses liveness probes to know when to restart a container. For example, liveness probes could catch a deadlock, where an application is running, but unable to make progress. Restarting a container in such a state can help to make the application more available despite bugs.	Yes
readinessProbe	It is important that you have a readiness probe for your containers. This way the Kubelet process	Yes

Parameter	Description	Should have a default?
	running on your data plane will know when the container is ready to serve traffic. A Pod is considered ready when all of its containers are ready. One use of this signal is to control which Pods are used as backends for Services. When a Pod is not ready, it is removed from Service load balancers.	
startupProbe	The kubelet uses startup probes to know when a container application has started. If such a probe is configured, it disables liveness and readiness checks until it succeeds, making sure those probes don't interfere with the application startup. This can be used to adopt liveness checks on slow starting containers, avoiding them getting killed by the kubelet before they are up and running.	Optional

Parameter	Description	Should have a default?
podDisruptionBudget	Define a Pod Disruption Budget (PDB) to ensure a minimum number of PODS keep running during voluntary disruptions. A PDB limits the number of Pods of a replicated application that are down simultaneously from voluntary disruptions. For example, a quorum-based application would like to ensure that the number of replicas running is never brought below the number needed for a quorum. A web front end might want to ensure that the number of replicas serving load never falls below a certain percentage of the total.	Yes, if defaulting to more than two replicas
serviceAccount (name)	Name of the service account pods will run under.	Yes

Parameter	Description	Should have a default?
serviceAccount (annotations)	Annotations applied to the service account. Typically used for IAM Roles for Service Accounts feature	No, IAM service account role ARN is set in top level Amazon EKS add-ons API. An exception to this rule is if your add-on has multiple deployments/controllers (such as Flux) and requires separate IRSA role ARNs.
priorityClassName	Priority indicates the importance of a Pod relative to other Pods. If a Pod cannot be scheduled, the scheduler tries to preempt (evict) lower priority Pods to make scheduling of the pending Pod possible.	Yes. Most add-ons are critical to cluster functionality, and should have a priority class set by default.
podSecurityContext	A security context defines privilege and access control settings for a Pod or Container. Typically used to set fsGroup - which was required for IRSA in v1.19 and lower clusters.	Unlikely, given Amazon EKS no longer supports Kubernetes v1.19
securityContext	A security context defines privilege and access control settings for a Pod or Container.	Yes

Parameter	Description	Should have a default?
updateStrategy	Specifies the strategy used to replace old Pods by new ones.	Yes
nameOverride	Override name of pods.	No
podSecurityPolicy	Enforce restrictions on parameters.	No - PSPs are deprecated
extraVolumeMounts/extraVolumes	Used for IRSA in non Amazon EKS clusters.	No

Container products pricing

This section outlines the available pricing models for container products. You can list free products, Bring Your Own License model (BYOL) products, and paid products for Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), and AWS Fargate. You can set only one price per product.

Note

You use the [AWS Marketplace Metering Service](#) to enforce entitlement and meter usage for your paid products. For per task or per pod pricing, usage is metered automatically by AWS.

The price you set for a container product applies to all AWS Regions. Whenever you lower the price for a container product, the new price is implemented for your buyers immediately. For price increases, existing buyers are notified about the change 90 days before it impacts their billing. New buyers are billed the new amount.

Note

For new subscribers, the price change is effective immediately. For existing subscribers, the price change is effective on the first day of the month following a 90-day period that begins on the date that the price change notification is sent. For example, say you send a price change notification on March 16. June 16 is about 90 days after March 16. Because

the price change happens on the first day of the month that follows the 90-day period, the effective date of the change is July 1.

Container pricing models

AWS Marketplace has multiple pricing models for container products.

The following table provides general information about pricing models for container-based products.

Pricing models for container products

Pricing model	Description
Bring Your Own License (BYOL)	BYOL is managed outside of AWS Marketplace through an external billing relationship that you maintain with the buyer.
Monthly	<p>Fixed monthly price</p> <p>A fixed monthly price that provides users with unlimited product usage during the following month.</p> <p>Example: You set the price for your product at \$99 per month. Your product includes three different container images that are deployed using an Amazon ECS task definition.</p> <p>After a buyer subscribes to your product, they're immediately charged \$99, which repeats each month until they cancel the subscription. The buyer also gets unlimited usage of the product. The buyer also pays separately for any infrastructure that the tasks run on. While subscribed, they can access your container images. They can launch and run any number of containers from those images on Amazon ECS or Amazon EKS in any configuration.</p> <p>If the buyer cancels their subscription in the middle of a month, they lose access to the Amazon ECR repository where AWS Marketplace stores the container images. The buyer might have pulled and stored the original images. However, they can no longer pull new container image versions that you make available through AWS Marketplace. The buyer is refunded</p>

Pricing model	Description
	for the unused portion of the final month. You're paid based on the buyer's usage minus the agreed-to AWS Marketplace fee.
Custom metric pricing dimensions	<p>Custom metered prices based off of dimensions you define (for example users, nodes, repositories, or GB), up to 24 dimensions per product.</p> <p>Example: Your product charges by users. You have admin users and regular users, and you define the pricing as \$2 for admin users and \$1 for regular users. You can set them up as separate dimensions when listing your product. You charge by users logged in per day and you meter that usage per day.</p>

Pricing model	Description
Per task or per pod hourly price	<p>Amazon ECS task or Amazon EKS pod</p> <p>Per Amazon ECS task or per Amazon EKS pod pricing that we measure to the second with the price set per hour.</p> <p>Example: Your product includes three different container images: a controller node, a worker node, and an analytics node. Because your product isn't functional or useful without the controller node, you decide that is the image that you want to charge usage for. You set a price of \$6 per hour.</p> <p>You modify the software in the container image for the controller node to integrate with the AWS Marketplace Metering Service RegisterUsage API operation. This ensures that only buyers with an active subscription can launch and run that container image and that its usage is metered based on how long it runs.</p> <p>The buyer is charged \$6 per hour of usage for each Amazon EKS controller pod running. If the buyer launches five Amazon EKS controller pods that include the controller node container, they're charged \$30 per hour (\$6 per pod). The buyer also pays separately for any infrastructure that the pods run on.</p> <p>For hourly pricing, billing is per-second with a 1-minute minimum. If the customer runs this controller container for 20 minutes and 30 seconds, they're charged $20 \times (\\$6/60) + 30 \times (\\$6/60/60) = \\$2 + \\$0.05 = \\$2.05$. You're paid based on the buyer's usage minus the agreed-to AWS Marketplace fee.</p>

Pricing model	Description
Hourly/usage with long-term contract	<p>A long-term contract, at a reduced price, paid up front or in regular installments. A long-term contract can be added to an existing product that has custom metered pricing, or per task and per pod pricing. Buyers pay the metered prices when they consume more than what they purchased in the long term contract.</p> <p>Example: For metered pricing models, you can add a long-term contract price for buyers to get a discount for committing upfront. Say that you normally charge \$1 per some unit consumed. A buyer using 1 unit per hour would pay \$8760 per year (365 days x 24 hours x \$1 per hour). You could enable a contract that enables the buyer to use 1 unit per hour for those 365 days at half that price (\$4380). In this case, the buyer commits to pay upfront for the one-year contract, and the price drops from \$1 per unit to \$0.5 per unit. You could also enable the buyer to purchase multiple of these contracts. If the quantity that is metered showed that the buyer consumed 10 units in an hour, and they had two contracts, then 2 units will be included in the 2 contracts. The 8 additional units would be billed at the regular \$1 per hour, for a total of \$8 in that hour.</p> <p>For the per task or per pod example, you can also add a long-term contract price for buyers to get a discount for committing upfront. If you normally charge \$6 per pod, you could set a long-term contract duration of 365 days with a price of \$13,140 (365 days x 24 hours x \$3 per pod per hour). One contract would then entitle the customer to 1 pod per hour during those 365 days. Customers can choose to purchase multiple contracts. For example, a customer can purchase two contracts which entitles them to 2 pods per hour. If the customer runs more pods per hour than the entitled contracts, then excess pods will be billed at your normal hourly price.</p> <p>In both cases, buyers that purchase long-term contracts will be billed upfront, either as a one-time payment or regularly scheduled future payments. Buyers will also be billed for any additional usage above their contract at the metered rate.</p>

Pricing model	Description
Container contract pricing	AMI with contract pricing – A container-based product that the buyer pays an upfront fee for.

Contract pricing for container products

For container-based products with contract pricing, AWS Marketplace bills your customers upfront or by the payment schedule that you define, based on the contract between you and your customer. After that point, they're entitled to use those resources.

To set your pricing, choose one or more contract durations that you offer customers. You can enter different prices for each contract duration. Your options are 1-month, 12-months, 24-month, and 36-month durations. For private offers, you can specify a custom duration in months (up to 60 months).

Choose the category that best describes your product's pricing. The pricing category appears to customers on the AWS Marketplace website. You can choose from **Bandwidth** (GB/s, MB/s), **Data** (GB, MB, TB), **Hosts**, **Requests**, **Tiers**, or **Users**. If none of the predefined categories fit your needs, you can choose the more generic **Units** category.

The offer allows for up to 24 dimensions to be added to it. Each dimension requires the following data:

- **Contracts Category** – The contract category is used to measure or meter your product if the product supports consumption based metering on top of contract pricing. For contract products with no consumption based pricing, you can choose a category which most closely resembles the category of dimension in the contract. If no values resemble the units for the dimension in the contract, choose **Units**.
- **Contracts Unit** – The contract unit is used along with category for metering if the product supports consumption based metering. Choose one of the available values for the units that closely matches your dimensions based on the category selected.
- **Contracts Dimension Allow Multiple Purchases** – This field is used to indicate whether an offer is a tiered pricing offer or a non-tiered offer which allows for purchase of multiple dimensions.

Tiered offer – Allows the buyer to subscribe to only one of the available dimensions in the offer. Dimensions in a tiered offer don't have the concept of quantities. Signing a contract with a

specific dimension essentially indicates that the buyer has chosen the specific feature indicated by that dimension.

Non-tiered offer – Allows the customer to procure more than one dimensions as part of the contract and allows them to procure multiple units of each such dimension.

Setting a value of *true* for this field indicates that the offer is a non-tiered offer. Setting a value of *false* for this field indicates that the offer is a tiered offer.

When using the Product Load Form (PLF) to create the contracts for your Container product, you must define the following fields for your pricing dimensions:

- **Contracts DimensionX API Name** – The name that should appear in the license generated in the buyer's AWS License Manager account. This name is also used as the value for Name in Entitlement in the CheckoutLicense API call.
- **Contracts DimensionX Display Name** – The customer-facing name of the dimension that will be displayed on the product detail and procurement pages of the AWS Marketplace website. Create a name that is user-friendly The name's maximum length is 24 characters. After the listing is public, the value of Name can't be changed.
- **Contracts DimensionX Description** – The customer-facing description of a dimension that provides additional information about the dimension for the product, such as the capabilities that the specific dimension provides. The maximum length for the description is 70 characters.
- **Contracts DimensionX Quantity** – This is used to calculate proration in cases of agreement amendments to a product. This value of this field should be set to 1 for all contract offers. It should not be edited.
- **Contracts DimensionX 1-Month Rate** – The contract rate to be charged for 1-month of entitlements against this dimension. For non-tiered offers, this rate is charged for each unit of the dimension that is procured. This field supports three decimal places.
- **Contracts DimensionX 12-Month Rate** – The contract rate to be charged for 12 months of entitlements against the dimension. For non-tiered offers, this rate is charged for each unit of the dimension that is procured. This field supports three decimal places.
- **Contracts DimensionX 24-Month Rate** – The contract rate to be charged for 24 months of entitlements against the dimension. For non-tiered offers, this rate is charged for each unit of the dimension that is procured. This field supports three decimal places.

- **Contracts DimensionX 36-Month Rate** – The contract rate to be charged for 36 months of entitlements against the dimension. For non-tiered offers, this rate is charged for each unit of the dimension that is procured. This field supports three decimal places.

Example: Data storage application

	1-month price	12-month price	24-month price	P36-month price
Unencrypted data (GB)	\$1.50/GB	\$16.00/GB	\$30.00/GB	\$60.00/GB
Encrypted data (GB)	\$1.55/GB	\$16.60/GB	\$31.20/GB	\$61.20/GB

Example: Log monitoring product

	1-month price	12-month price	24-month price	36-month price
Basic (10 hosts monitored, 5 containers monitored)	\$100	\$1000	\$2000	\$4000
Standard (20 hosts monitored, 10 containers monitored)	\$200	\$2000	\$4000	\$8000
Pro (40 hosts monitored, 20 containers monitored)	\$400	\$4000	\$8000	\$16,000
Additional hosts monitored per hour	\$10	\$100	\$200	\$400

	1-month price	12-month price	24-month price	36-month price
Additional containers monitored per hour	\$10	\$100	\$200	\$400

Note

The prices can be for the following durations: 1 month, 12 months, 24 months, or 36 months. You can choose to offer one or more of these options for your product. The durations must be the same across each dimension.

Example

For example, in a case where you have `ReadOnlyUsers` and `AdminUsers` dimensions, if you offer a yearly price for `ReadOnlyUsers`, you must offer a yearly price for `AdminUsers`, too.

Automatic renewals

When customers purchase your product through AWS Marketplace using container contracts, they can agree to automatically renew the contract terms. Customers continue to pay for the entitlements every month or for 1, 2, or 3 years.

Customers can modify their renewal settings at any time. For more information, see [Modifying an existing contract](#) in the *AWS Marketplace Buyer Guide*.

When a container contract ends

A container contract product has a contract expiry. When a contract ends, the following events occur:

1. Your container product receives an entitlement-updated notification indicating that the buyer's entitlement has changed, and the AWS Marketplace Entitlement Service returns an empty response.

2. You have one hour to meter any remaining usage for the customer. After this you can no longer send metering records for this customer.

Container product billing, metering, and licensing integrations

AWS Marketplace integrates with other AWS services to provide both metering and contract-based pricing for your container product.

Hourly and custom metering with AWS Marketplace Metering Service

To both check entitlement to use your product and to meter usage for billing, use the [AWS Marketplace Metering Service](#). If you want to define your own pricing units and meter that usage to us for billing, integrate by using the [MeterUsage](#) API operation. If you want to price your product based on number of tasks or pods used and have AWS meter that usage automatically, integrate by using the [RegisterUsage](#) API operation. For both types of pricing, you can add a long-term contract price without changing how you integrate with the AWS Marketplace Metering Service.

When you create a new container product in the AWS Marketplace Management Portal, we provide a set of product identifiers (the product code and public key) that are used to integrate your product with the AWS Marketplace Metering Service.

Entitlement

Integrating with the AWS Marketplace Metering Service allows you to verify that the customer running your paid software is subscribed to your product on AWS Marketplace, guarding you against unauthorized use at container startup. To verify entitlement, use the [MeterUsage](#) or [RegisterUsage](#) API operations, depending on your pricing model. For hourly and fixed monthly pricing models, use the `RegisterUsage` API operation. For custom metering pricing models, use the `MeterUsage` API operation.

If a buyer isn't entitled to your product, these API operations return the `CustomerNotEntitledException` exception.

Note

If a buyer unsubscribes from your product while running it, they are entitled to continue running it. However, they can't launch additional containers for your product.

Integration guidelines

As you create and publish your container products and use the `MeterUsage` or `RegisterUsage` API operations for entitlement and metering, keep the following guidelines in mind:

- Don't configure AWS credentials within your software or the Docker container image. AWS credentials for the buyer are automatically obtained at runtime when your container image is running within an Amazon ECS task or Amazon EKS pod.
- To call the `MeterUsage` or `RegisterUsage` API operations from Amazon EKS, you must [use a supported AWS SDK](#). To test `MeterUsage` or `RegisterUsage` integration of Amazon EKS, you must run an Amazon EKS cluster running Kubernetes 1.13.x or greater. Kubernetes 1.13 is required for AWS Identity and Access Management (IAM) roles for pod support. IAM roles are required for the running pod to obtain the AWS credentials required to invoke these actions on Amazon EKS.
- You can do local development, but you will get a `PlatformNotSupportedException` exception. This exception won't occur when you launch the container on AWS container services (Amazon ECS, Amazon EKS, and Fargate).

Supported AWS Regions

For a list of all AWS Marketplace supported AWS Regions, see [Region Table](#) on the Global Infrastructure website.

Obtaining the AWS Region for metering

When integrating your container for metering with either the `MeterUsage` or `RegisterUsage` API operation, don't configure the AWS SDK to use a specific AWS Region. The Region must be obtained dynamically at runtime.

Example

For example, a customer launches an Amazon ECS task or Amazon EKS pod. The `RegisterUsage` API operation is called in a Region that differs from the Region where the Amazon ECS task or Amazon EKS pod was launched. Therefore, the `RegisterUsage` API operation throws an `InvalidRegionException` error.

AWS SDK languages don't determine the `AWS_REGION` in a consistent manner. If your SDK does not automatically pick up the `AWS_REGION`, software needs to be written manually to determine

the `AWS_Region`. For example, the AWS SDK for Java automatically uses [Amazon EC2 instance metadata](#) (specifically, `ec2InstanceMetadata`) to obtain the Region when environment variables or other configuration aren't present. In this instance, only call `ec2InstanceMetadata` if the `AWS_REGION` environment variable isn't present.

For information about how to dynamically obtain an AWS Region at runtime, refer to the [AWS SDK Developer Guide](#) for your programming language.

Preventing metering modification

Introducing ways for buyers to modify or override calls to `RegisterUsage` or `MeterUsage` might result in undesirable billing and payment issues. We strongly recommend that you integrate the metering and entitlement logic.

When engineering your product to prevent metering modification, keep the following in mind:


- If buyers can insert new image layers that contain `CMD` or `ENTRYPOINT` instructions, directly integrate `RegisterUsage` or `MeterUsage` into the software that the buyer is running through your container image. Otherwise, calls to `RegisterUsage` or `MeterUsage` executed via `CMD` or `ENTRYPOINT` from the base image will likely be overridden by the buyer.
- We recommend that you manage the AWS Marketplace product codes that your software uses as input to `RegisterUsage` or `MeterUsage` in a manner buyers can't modify. However, if your product manages product codes in a manner customers can override, such as AWS CloudFormation, Helm chart, or Kubernetes manifest, you must maintain a list of *trusted* AWS Marketplace product codes. This is to ensure that the product code your software passes as input to `RegisterUsage` or `MeterUsage` is valid.
- If any of your trusted product codes are for free products, ensure that they can't be used in place of a paid product code.

Contract pricing with AWS License Manager

For container-based products with contract pricing, you use AWS License Manager to associate licenses with your product.

AWS License Manager is a license management tool that enables your application to track and update licenses (also known as entitlements) that have been purchased by a customer. This section provides information about how to integrate your product with AWS License Manager. After the integration is complete, you can publish your product listing on AWS Marketplace.

For more information about AWS License Manager, see the [AWS License Manager User Guide](#) and the [AWS License Manager](#) section of the *AWS CLI Command Reference*.

 **Note**

- Customers can't launch new instances of the container after the contract expiry period. However, during the contract duration, they can launch any number of instances. These licenses are not bound to a specific node or instance. Any software running on any container on any node can checkout the license as long as it has the assigned AWS credentials.
- **Private Offer Creation** – Sellers can generate private offers for the products using the Private offer creation tool in the AWS Marketplace Management Portal.
- **Reporting** – You can set up data feeds by setting up an Amazon S3 bucket in the **Report** section in the AWS Marketplace Management Portal. For more information, see [Seller reports, data feeds, and dashboards](#).

Integration workflow

The following steps show the workflow for integrating your container product with AWS License Manager:

1. Seller creates a product with AWS License Manager integration.
2. Seller lists the product on AWS Marketplace.
3. Buyer finds the product on AWS Marketplace and purchases it.
4. A license is sent to the buyer in their AWS account.
5. Buyer uses the software by launching the Amazon EC2 instance, Amazon ECS task, or Amazon EKS pod software. The customer deploys using an IAM role.
6. Software reads the license in the buyer's AWS License Manager account, discovers the entitlements purchased, and provisions the features accordingly.

Note

License Manager doesn't do any tracking or updates; this is done by the seller's application.

Hourly metering with AWS Marketplace Metering Service

If your container product uses per-hour per-task or per-pod pricing instead of custom metered pricing dimensions, you don't have to define custom metering dimensions.

The `RegisterUsage` API operation meters software use per Amazon Elastic Container Service (Amazon ECS) task or per Amazon Elastic Kubernetes Service (Amazon EKS) pod, per hour, with usage prorated to the second. A minimum of 1 minute of usage applies to tasks or pods that are short lived. Continuous metering for software use is automatically handled by the AWS Marketplace Metering Control Plane. Your software isn't required to perform any metering specific actions except calling `RegisterUsage` once for metering of software use to commence.

`RegisterUsage` must be called immediately at the time of launching a container. If you don't register the container in the first 6 hours of the container launch, AWS Marketplace Metering Service doesn't provide any metering guarantees for previous months. However, the metering will continue for the current month forward until the container ends.

The AWS Marketplace Metering Control Plane continues to bill customers for running Amazon ECS tasks and Amazon EKS pods, regardless of the customer's subscription state. This removes the need for your software to perform entitlement checks after the initial successful launch of the task or pod.

Hourly metering prerequisites

Before publishing the product, you must do the following:

1. Create a new container product in the AWS Marketplace Management Portal, and make a note of its product code.

For more information, see [Creating a container product](#).

2. Fill out the product load form (PLF) with the necessary hourly price information, and return it to us for processing.

For more information, see [Creating or updating pricing details for container products \(legacy\)](#).

3. Use an AWS Identity and Access Management (IAM) role for the task or pod running your application with the IAM permissions necessary to call `RegisterUsage`. The IAM managed policy `AWSMarketplaceMeteringRegisterUsage` has these permissions.
4. (Optional) If you want to see logging, we recommend that you enable AWS CloudTrail logging in the task or pod definition.
5. Make a test call to the `RegisterUsage` API operation with a record for all of the pricing dimensions you define.

Product load form for hourly metering

When filling out the product load form for hourly metering, fill out the following fields for your product, in addition to the other required and optional fields that define your product:

- **Hourly Price** – The price for your product, per hour.
- **Dimension Long Term Rate** – The total software price over a long-term contract when buyers pay upfront.
- **Long Term Duration (Days)** – The duration, in days, for the long-term contract.

Testing integration and preview mode for RegisterUsage

Use the `RegisterUsage` API operation to test your integration before submitting your image to AWS Marketplace for publishing.

Preview mode operates identically to production mode, except preview mode does not verify entitlement to use your product. To call `RegisterUsage` in preview mode, call `RegisterUsage` from the container image by running your product on Amazon ECS or Amazon EKS. Use the AWS account that you're using to list the product on AWS Marketplace. Your metering integration must dynamically set the AWS Region, rather than hardcoding it. However, when testing, launch at least one Amazon ECS task or Amazon EKS pod containing your paid container in the US East (N. Virginia) Region. By doing this, the AWS Marketplace operations team can verify your work with the logs in that Region.

Note

If your product supports both Amazon ECS and Amazon EKS, you only need to launch in Amazon EKS for us to validate your integration.

You can't fully test the integration until your product is published with all the required metadata and pricing information. If requested, the AWS Marketplace catalog operations team can verify receipt of your metering records in preview mode.

Error handling for RegisterUsage

If your container image integrates with the AWS Marketplace Metering Service and receives an exception other than `ThrottlingException` at container startup, you should terminate the container to prevent unauthorized use.

Exceptions other than `ThrottlingException` are thrown only on the initial call to the `RegisterUsage` API operation. Subsequent calls from the same Amazon ECS task or Amazon EKS pod don't throw `CustomerNotSubscribedException` even if the customer unsubscribes while the task or pod is still running. These customers are still charged for running containers after they unsubscribe, and their usage is tracked.

The following table describes the errors that the `RegisterUsage` API operation might throw. Each AWS SDK programming language has a set of error handling guidelines that you can refer to for additional information.

Error	Description
<code>InternalServerErrorException</code>	<code>RegisterUsage</code> isn't available.
<code>CustomerNotEntitledException</code>	The customer doesn't have a valid subscription for the product.
<code>InvalidProductCodeException</code>	The <code>ProductCode</code> value passed in as part of the request doesn't exist.
<code>InvalidPublicKeyException</code>	The <code>PublicKeyVersion</code> value passed in as part of the request doesn't exist.

Error	Description
PlatformNotSupportedException	AWS Marketplace doesn't support metering usage from the underlying platform. Only Amazon ECS, Amazon EKS, and AWS Fargate are supported.
ThrottlingException	The calls to <code>RegisterUsage</code> are throttled.
InvalidRegionException	<code>RegisterUsage</code> must be called in the same AWS Region that the Amazon ECS task or Amazon EKS pod was launched in. This prevents a container from choosing a Region (for example, <code>withRegion("us-east-1")</code>) when calling <code>RegisterUsage</code> .

Integrating your container product with the AWS Marketplace Metering Service using the AWS SDK for Java

The following steps outline an example implementation using the AWS SDK for Java to integrate with the [AWS Marketplace Metering Service](#)'s `RegisterUsage` action. For the full source code, see [RegisterUsage Java example](#). Many of these steps apply regardless of the language.

Example steps for AWS Marketplace Metering Service integration

1. Sign into the [AWS Marketplace Management Portal](#).
2. From **Assets** choose **Containers** to start creating a new container product. Creating the product generates the product code for the product to integrate with your container image. For more information about publishing, see [Publishing container products \(legacy\)](#). For information about setting IAM permissions, see [the section called "AWS Marketplace metering and entitlement API permissions"](#).
3. Download the public [AWS Java SDK](#).

⚠ Important

To call the metering APIs from Amazon EKS, you must [use a supported AWS SDK](#) and run on an Amazon EKS cluster running Kubernetes 1.13 or later.

4. (Optional) If you're integrating with the RegisterUsage action and you want to perform digital signature verification, you need to configure the [BouncyCastle](#) signature verification library in your application classpath.

If you want to use JSON Web Token (JWT), you must also include [JWT Java](#) libraries in your application classpath. Using JWT provides a simpler approach to signature verification but is not required, and you can use standalone BouncyCastle instead. Whether you use JWT or BouncyCastle, you need to use a build system such as Maven to include transitive dependencies of BouncyCastle or JWT in your application classpath.

```
// Required for signature verification using code sample
<dependency>
  <groupId>org.bouncycastle</groupId>
  <artifactId>bcpkix-jdk15on</artifactId>
  <version>1.60</version>
</dependency>

// This one is only required for JWT
<dependency>
  <groupId>com.nimbusds</groupId>
  <artifactId>nimbus-jose-jwt</artifactId>
  <version>6.0</version>
</dependency>
```

5. Call RegisterUsage from each paid container image in your product offering. ProductCode and PublicKeyVersion are required parameters, and all other inputs are optional. The following is an example payload for RegisterUsage.

```
{
  "ProductCode" : "string", // (required)
  "PublicKeyVersion": 1,    // (required)
  "Nonce": "string",       // (optional) to scope down the registration
                           // to a specific running software
                           // instance and guard against
                           // replay attacks
```

```
}

```

Note

It is possible to see transient issues in connecting to the AWS Marketplace Metering Service. AWS Marketplace strongly recommends implementing retries for up to 30 minutes, with exponential back off, to avoid short-term outages or network issues.

6. RegisterUsage generates an RSA-PSS digital signature using SHA-256 that you can use to verify request authenticity. The signature includes the following fields: ProductCode, PublicKeyVersion, and Nonce. To verify the digital signature, you must retain these fields from the request. The following code is an example response to a RegisterUsage call.

```
{
  "Signature": "<<JWT Token>>"
}

// Where the JWT Token is composed of 3 dot-separated,
// base-64 URL Encoded sections.
// e.g. eyJhbGcVCj9.eyJzdWIMzkwMjJ9.rr09Qw0SXRWTe

// Section 1: Header/Algorithm
{
  "alg": "PS256",
  "typ": "JWT"
}

// Section 2: Payload
{
  "ProductCode" : "string",
  "PublicKeyVersion": 1,
  "Nonce": "string",
  "iat": date // JWT issued at claim
}

// Section 3: RSA-PSS SHA256 signature
"rr09Q4FEi3gweH3X4lrt2okf5zwIatUUwERlw016wTy_21Nv8S..."

```

7. Rebuild a new version of your container image that includes the RegisterUsage call, tag the container, and push it to any container registry that is compatible with Amazon ECS or Amazon EKS, such as Amazon ECR or Amazon ECR Public. If you are using Amazon ECR, ensure that the

account launching the Amazon ECS task or Amazon EKS pod has permissions on the Amazon ECR repository. Otherwise, the launch fails.

8. Create an [IAM](#) role that grants permission for your container to call `RegisterUsage`, as defined in the following code. You must supply this IAM role in the [Task Role](#) parameter of the Amazon ECS task or Amazon EKS pod definition.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "aws-marketplace:RegisterUsage"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

9. Create an Amazon ECS task or Amazon EKS pod definition that references the container that has integrated with AWS Marketplace and references the IAM role that you created in step 7. You should enable AWS CloudTrail logging in the task definition if you want to see logging.
10. Create an Amazon ECS or Amazon EKS cluster to execute your task or pod. For more information about creating an Amazon ECS cluster, see [Creating a Cluster](#) in the *Amazon Elastic Container Service Developer Guide*. For more information about creating an Amazon EKS cluster (using Kubernetes version 1.13.x or later), see [Creating an Amazon EKS Cluster](#).
11. Configure the Amazon ECS or Amazon EKS cluster and launch the Amazon ECS task definition or Amazon EKS pod that you created, in the us-east-1 AWS Region. It's only during this testing process, before the product is live, that you have to use this region.
12. When you get a valid response back from `RegisterUsage`, you can begin creating your container product. For questions, contact the [AWS Marketplace Seller Operations](#) team.

RegisterUsage Java example

The following example uses the AWS SDK for Java and AWS Marketplace Metering Service to call the `RegisterUsage` operation. Signature verification is optional, but if you want to perform signature verification, you must include the required digital signature verification libraries. This example is for illustrative purposes only.

```
import com.amazonaws.auth.PEM;
import com.amazonaws.services.marketplacemetering.AWSMarketplaceMetering;
import com.amazonaws.services.marketplacemetering.AWSMarketplaceMeteringClientBuilder;
import com.amazonaws.services.marketplacemetering.model.RegisterUsageRequest;
import com.amazonaws.services.marketplacemetering.model.RegisterUsageResult;
import com.amazonaws.util.json.Jackson;
import com.fasterxml.jackson.databind.JsonNode;
import com.nimbusds.jose.JWSObject;
import com.nimbusds.jose.JWSVerifier;
import com.nimbusds.jose.crypto.RSASSAVerifier;
import java.io.ByteArrayInputStream;
import java.nio.charset.StandardCharsets;
import java.security.PublicKey;
import java.security.Security;
import java.security.Signature;
import java.security.interfaces.RSAPublicKey;
import java.util.Base64;
import java.util.Optional;
import java.util.UUID;
import org.bouncycastle.jce.provider.BouncyCastleProvider;

/**
 * Class for making calls out to AWS Marketplace Metering Service.
 */
class RegisterUsage {

    private static final String PRODUCT_CODE = ".....";

    private final AWSMarketplaceMetering registerUsageClient;
    private final SignatureVerifier signatureVerifier;
    private final int publicKeyVersion;

    public RegisterUsage(final SignatureVerifier signatureVerifier) {
        this.signatureVerifier = signatureVerifier;
        this.publicKeyVersion = PublicKeyProvider.PUBLIC_KEY_VERSION;
        this.registerUsageClient =
AWSMarketplaceMeteringClientBuilder.standard().build();
    }

    /**
     * Shows how to call RegisterUsage client and verify digital signature.
     */
    public void callRegisterUsage() {
```

```
RegisterUsageRequest request = new RegisterUsageRequest()
    .withProductCode(PRODUCT_CODE)
    .withPublicKeyVersion(publicKeyVersion)
    .withNonce(UUID.randomUUID().toString());

// Execute call to RegisterUsage (only need to call once at container startup)
RegisterUsageResult result = this.registerUsageClient.registerUsage(request);

// Verify Digital Signature w/o JWT
boolean isSignatureValid = this.signatureVerifier.verify(request, result);
if (!isSignatureValid) {
    throw new RuntimeException("Revoke entitlement, digital signature
invalid.");
}
}
}

/**
 * Signature verification class with both a JWT-library based verification
 * and a non-library based implementation.
 */
class SignatureVerifier {
    private static BouncyCastleProvider BC = new BouncyCastleProvider();

    private static final String SIGNATURE_ALGORITHM = "SHA256withRSA/PSS";

    private final PublicKey publicKey;

    public SignatureVerifier(PublicKeyProvider publicKeyProvider) {
        this.publicKey = publicKeyProvider.getPublicKey().orElse(null);
        Security.addProvider(BC);
    }

    /**
     * Example signature verification using the NimbusJOSEJWT library to verify the JWT
     * Token.
     *
     * @param request RegisterUsage Request.
     * @param result RegisterUsage Result.
     * @return true if the token matches.
     */
    public boolean verifyUsingNimbusJOSEJWT(final RegisterUsageRequest request, final
RegisterUsageResult result) {
        if (!getPublicKey().isPresent()) {
```

```
        return false;
    }

    try {
        JWSVerifier verifier = new RSASSAVerifier((RSAPublicKey)
getPublicKey().get());
        JWSObject jwsObject = JWSObject.parse(result.getSignature());
        return jwsObject.verify(verifier) &&
validatePayload(jwsObject.getPayload().toString(), request, result);
    } catch (Exception e) {
        // log error
        return false;
    }
}

/**
 * Example signature verification without any JWT library support.
 *
 * @param request RegisterUsage Request.
 * @param result RegisterUsage Result.
 * @return true if the token matches.
 */
public boolean verify(final RegisterUsageRequest request, final RegisterUsageResult
result) {
    if (!getPublicKey().isPresent()) {
        return false;
    }
    try {
        String[] jwtParts = result.getSignature().split("\\.");
        String header = jwtParts[0];
        String payload = jwtParts[1];
        String payloadSignature = jwtParts[2];

        Signature signature = Signature.getInstance(SIGNATURE_ALGORITHM, BC);
        signature.initVerify(getPublicKey().get());
        signature.update(String.format("%s.%s", header,
payload).getBytes(StandardCharsets.UTF_8));
        boolean verified = signature.verify(Base64.getUrlDecoder()
                .decode(payloadSignature.getBytes(StandardCharsets.UTF_8)));

        String decodedPayload = new String(Base64.getUrlDecoder().decode(payload));
        return verified && validatePayload(decodedPayload, request, result);
    } catch (Exception e) {
        // log error
    }
}
```

```

        return false;
    }
}

/**
 * Validate each value in the returned payload matches values originally
 * supplied in the request to RegisterUsage. TimeToLiveInMillis and
 * PublicKeyExpirationTimestamp will have the values in the payload compared
 * to values in the signature
 */
private boolean validatePayload(final String payload, final RegisterUsageRequest
request,
                                final RegisterUsageResult result) {
    try {
        JsonNode payloadJson = Jackson.getObjectMapper().readTree(payload);
        boolean matches = payloadJson.get("productCode")
            .asText()
            .equals(request.getProductCode());
        matches = matches && payloadJson.get("nonce")
            .asText()
            .equals(request.getNonce());
        return matches = matches && payloadJson.get("publicKeyVersion")
            .asText()
            .equals(String.valueOf(request.getPublicKeyVersion()));
    } catch (Exception ex) {
        // log error
        return false;
    }
}

private Optional<PublicKey> getPublicKey() {
    return Optional.ofNullable(this.publicKey);
}
}

/**
 * Public key provider taking advantage of the AWS PEM Utility.
 */
class PublicKeyProvider {
    // Replace with your public key. Ensure there are new-lines ("\n") in the
    // string after "-----BEGIN PUBLIC KEY-----\n" and before "\n-----END PUBLIC
    KEY-----".
    private static final String PUBLIC_KEY =

```



```

        "-----BEGIN PUBLIC KEY-----\n"
            + "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDdlatRjRjogo3WojgGHFHYLugd
\n"
            + "UWAY9iR3fy4arWNA1KoS8kVw33cJibXr8bvWUAUparCwlvdbH6dvE0fou0/gCFQs
\n"
            + "HUfQrSDv+MuSUMAe8jzKE4qW+jK+xQU9a03GUnKHkkle+Q0pX/g6jXZ7r1/xAK5D
\n"
            + "o2kQ+X5xK9cipRgEKwIDAQAB\n"
            + "-----END PUBLIC KEY-----";

public static final int PUBLIC_KEY_VERSION = 1;

public Optional<PublicKey> getPublicKey() {
    try {
        return Optional.of(PEM.readPublicKey(new ByteArrayInputStream(
            PUBLIC_KEY.getBytes(StandardCharsets.UTF_8))));
    } catch (Exception e) {
        // log error
        return Optional.empty();
    }
}
}
}

```

Custom metering for container products with AWS Marketplace Metering Service

AWS Marketplace container products can have custom metering on up to 24 different pricing dimensions per product. Each dimension can have a long-term contract price associated with it. To enable custom metering, integrate your container product with AWS Marketplace Metering Service. You can define your own pricing units and custom metering for that usage to AWS for billing using the [MeterUsage](#) API operation.

Price dimensions are defined in two locations, once in the product load form and once through the `MeterUsage` operation. This two-factor method ensures that the subsequent offers are working as intended before they're made available to the public.

To set up custom metering, you'll need to choose the usage category, the unit type, and pricing dimensions:

- **Usage category** – The usage category helps buyers understand what your product is and how to use it.

- **Unit type** – The unit type defines the unit of measure for billing. For example, bandwidth measured in GBps or MBps, the number of hosts, or data measured in MB, GB, or TB.
- **Pricing dimensions** – The pricing dimensions represents a feature or service that you've set a per-unit price for (for example, users, scans, vCPUs, or deployed agents). Pricing dimensions are public. However, you can still define private and Bring Your Own License (BYOL) offers for public products. Don't send pricing in the metering records. You meter the quantity of units, and we use that along with the prices you defined when creating your product to compute the buyer's bill.

If your product pricing doesn't fit with any of the predefined categories or unit types, you can choose the generic **Units** category. Then, use the dimension description to describe what the unit is.

Optionally, you may distribute the usage into allocations by properties that you track. The allocations are represented as tags to the buyer. These tags allow the buyer to view their costs split into usage by tag values. For example, if you charge by the user, and users have a "Department" property, you could create usage allocations with tags that have a key of "Department", and one allocation per value. This does not change the price, dimensions, or the total usage that you report, but allows your customer to view their costs by categories appropriate to your product.

We recommend that you send a metering record every hour. However, you can aggregate usage over daily or monthly periods as well. If you experience an outage, you can aggregate buyer software use and send it in the following hours metering. You can't send more than one record per hour.

Important

Free trial and prepaid entitlement are tracked on an hourly level. As a result, sending these records in separately might lead to the buyer being overcharged.

Custom metering prerequisites

Before publishing the product, you must do the following:

1. Create a new container product in the AWS Marketplace Management Portal, and make a note of its product code.
2. Fill out the product load form with the necessary dimension information, and return it to us for processing.

3. Use an AWS Identity and Access Management (IAM) role for the task or pod running your application with the IAM permissions necessary to call `MeterUsage`. The IAM managed policy `AWSMarketplaceMeteringRegisterUsage` has these permissions.
4. (Optional) We recommend that you enable AWS CloudTrail logging in the task or pod definition if you want to see logging.
5. Make a test call to the `MeterUsage` API operation with a record for all of the pricing dimensions you define.

Product load form for custom metering

When filling out the product load form for custom metering, each product can have up to 24 dimensions. The dimensions are defined in the following fields:

- **Dimension Name** – The name used when your container application is sending metering records to the AWS Marketplace Metering Service. This name indicates which dimension your buyer will use. This name is visible in billing reports. After you set the name, you can't change it.
- **Dimension Description** – The buyer-facing description for the dimension. The description can't exceed 70 characters. After the product is published publicly to buyers, this field can't be changed.
- **Dimension Rate** – The software price per unit for this product when buyers pay as they go. This field supports three decimal places.
- **Dimension Long Term Rate** – The total software price over a long-term contract when buyers pay upfront.
- **Long Term Duration (Days)** – The duration, in days, for the long-term contract.

Testing `MeterUsage` integration and preview mode

Use the `MeterUsage` operation to test your integration before submitting your image to AWS Marketplace for publishing.

Preview mode operates identically to production mode, except preview mode does not verify entitlement to use your product. To call `MeterUsage` in preview mode, call `MeterUsage` from the container images by running your product on Amazon Elastic Container Service (Amazon ECS) or Amazon Elastic Kubernetes Service (Amazon EKS) with the AWS account you are using to list the product on AWS Marketplace. Your metering integration must dynamically set the AWS Region, rather than hard coding it. However, when testing, launch at least one Amazon ECS task or

Amazon EKS pod containing your paid container in the US East (N. Virginia) Region so that the AWS Marketplace operations team can verify your work with the logs in that Region.

Note

- If your product supports both Amazon ECS and Amazon EKS, you only need to launch in Amazon EKS for us to validate your integration.
- Test every dimension before launching your product to the public and after adding a new dimension. If you don't send a metering record for each dimension associated with a container product, it will result in an error with the request failing.

You can't fully test the integration until your product is published with all the required metadata and pricing information. If requested, the AWS Marketplace catalog operations team can verify receipt of your metering records in preview mode.

Error handling for `MeterUsage`

If your container image integrates with the `MeterUsage` operation and receives an exception other than `ThrottlingException` at container startup, you should terminate the container to prevent unauthorized use.

Exceptions other than `ThrottlingException` are thrown only on the initial call to `MeterUsage`. Subsequent calls from the same Amazon ECS task or Amazon EKS pod do not throw `CustomerNotSubscribedException` even if the customer unsubscribes while the task or pod is still running. These customers are still charged for running containers after they unsubscribe and their usage is tracked.

See [MeterUsage](#) in the *AWS Marketplace Metering Service API Reference* for detailed descriptions of common errors for `MeterUsage`. Each AWS SDK programming language has a set of error handling guidelines that you can refer to for additional information.

Vendor-metered tagging (Optional)

Vendor-metered tagging helps Independent Software Vendors (ISVs) give the buyer more granular insight into their software usage and can help them perform cost allocation.

There are many ways to tag a buyer's software usage. One way is to first ask your buyers what they want to see in their cost allocation. Then you can split the usage across properties that you

track for the buyer's account. Examples of properties include `AccountId`, `Business Unit`, `Cost Centers`, and other relevant metadata for your product. These properties are exposed to the buyer as tags. Using tags, buyers can view their costs split into usage by the tag values in their AWS Billing Console (<https://console.aws.amazon.com/billing/>). Vendor-metered tagging doesn't change the price, dimensions, or the total usage that you report. It allows your customer to view their costs by categories appropriate to your product.

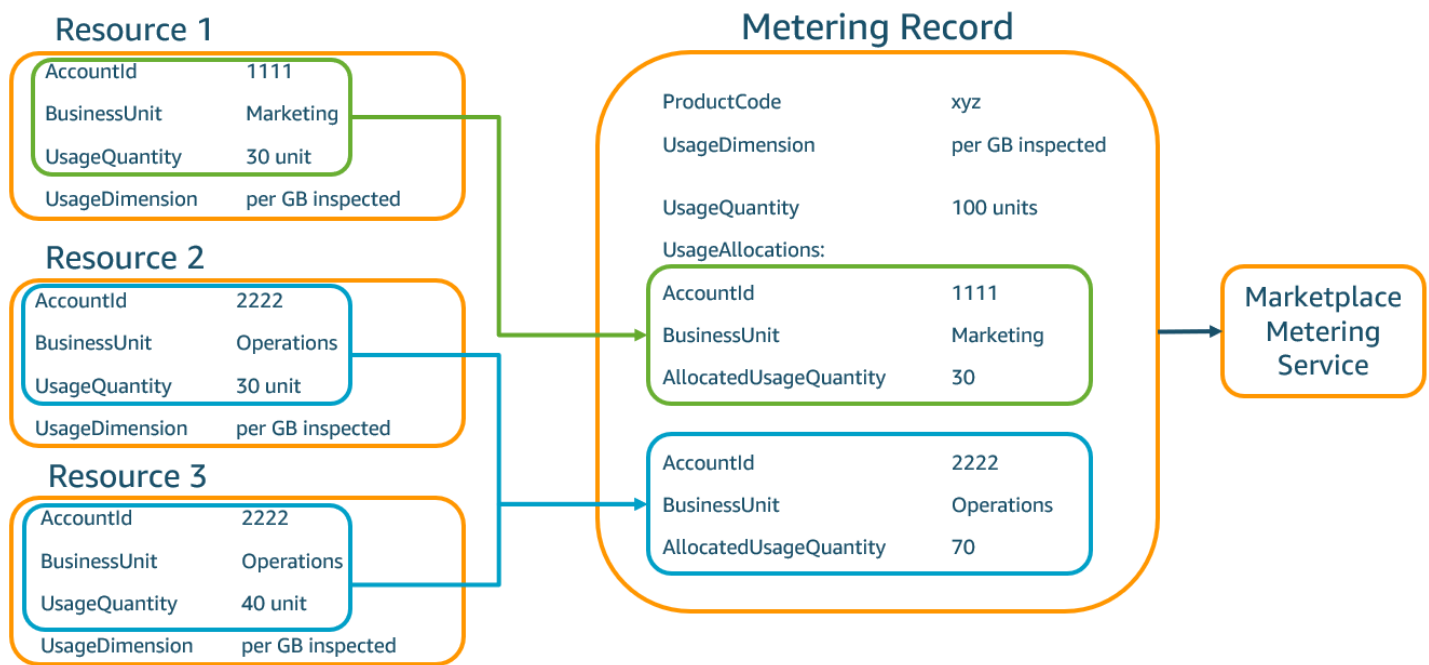
In a common use case, a buyer subscribes to your product with one AWS account. The buyer also has numerous users associated with the same product subscription. You can create usage allocations with tags that have a key of `AccountId`, and then allocate usage to each user. In this case, buyers can activate the `AccountId` tag in their Billing and Cost Management console and analyze individual user usage.

Seller experience

Sellers can aggregate the metering records for resources with the same set of tags instead of aggregating usage for all resources. For example, sellers can construct the metering record that includes different buckets of `UsageAllocations`. Each bucket represents `UsageQuantity` for a set of tags, such as `AccountId` and `BusinessUnit`.

In the following diagram, **Resource 1** has a unique set of `AccountId` and `BusinessUnit` tags, and appears in the **Metering Record** as a single entry.

Resource 2 and **Resource 3** both have the same `AccountId` tag, `2222`, and the same `BusinessUnit` tag, `Operations`. As a result, they're combined into a single `UsageAllocations` entry in the **Metering Record**.



Sellers can also combine resources without tags into a single UsageAllocation with the allocated usage quantity and send it as one of the entries in UsageAllocations.

Limits include:

- Number of tags – 5
- Size of UsageAllocations (cardinality) – 2,500

Validations include:

- Characters allowed for the tag key and value – a-zA-Z0-9+ -=._:\/@
- Maximum tags across UsageAllocation list – 5
- Two UsageAllocations can't have the same tags (that is, the same combination of tag keys and values). If that's the case, they must use the same UsageAllocation.
- The sum of AllocatedUsageQuantity of UsageAllocation must equal the UsageQuantity, which is the aggregate usage.

Buyer experience

The following table shows an example of the buyer experience after a buyer activates the AccountId and BusinessUnit vendor tags.

In this example, the buyer can see allocated usage in their **Cost Usage Report**. The vendor-metered tags use the prefix “aws:marketplace:isv”. Buyers can activate them in the Billing and Cost Management, under **Cost Allocation Tags, AWS-generated cost allocation tags**.

The first and last rows of the **Cost Usage Report** are relevant to what the Seller sends to the Metering Service (as shown in the [Seller experience](#) example).

Cost Usage Report (Simplified)

ProductCode	Buyer	UsageDimension	UsageQuantity	aws:marketplace:isv:AccountId	aws:marketplace:isv:BusinessUnit
xyz	111122223333	Network: per (GB) inspected	70	2222	Operations
xyz	111122223333	Network: per (GB) inspected	30	3333	Finance
xyz	111122223333	Network: per (GB) inspected	20	4444	IT
xyz	111122223333	Network: per (GB) inspected	20	5555	Marketing
xyz	111122223333	Network: per (GB) inspected	30	1111	Marketing

For a code example, see [MeterUsage code example with usage allocation tagging \(Optional\)](#).

Code example

The following code example is provided to help you integrate your container product with the AWS Marketplace APIs required for publishing and maintaining your product.

MeterUsage code example with usage allocation tagging (Optional)

The following code example is relevant for container products with consumption pricing models. The Python example sends a metering record with appropriate usage allocation tags to AWS Marketplace to charge your customers for pay-as-you-go fees.

```
# NOTE: Your application will need to aggregate usage for the
#       customer for the hour and set the quantity as seen below.
#       AWS Marketplace can only accept records for up to an hour in the past.
#
# productCode is supplied after the AWS Marketplace Ops team has
# published the product to limited

# Import AWS Python SDK
import boto3
import time

usageRecord = [
    {
        "AllocatedUsageQuantity": 2,
        "Tags":
            [
                { "Key": "BusinessUnit", "Value": "IT" },
                { "Key": "AccountId", "Value": "123456789" },
            ]
    },
    {
        "AllocatedUsageQuantity": 1,
        "Tags":
            [
                { "Key": "BusinessUnit", "Value": "Finance" },
                { "Key": "AccountId", "Value": "987654321" },
            ]
    }
]
```



```
marketplaceClient = boto3.client("meteringmarketplace")

response = marketplaceClient.meter_usage(
    ProductCode="testProduct",
    Timestamp=int(time.time()),
    UsageDimension="Dimension1",
    UsageQuantity=3,
    DryRun=False,
    UsageAllocations=usageRecord
)
```

For more information about MeterUsage, see [MeterUsage](#) in the *AWS Marketplace Metering Service API Reference*.

Example response

```
{ "MeteringRecordId": "string" }
```

Integrating your container product with the AWS Marketplace Metering Service using the AWS SDK for Java

The following example outlines an implementation that uses the AWS SDK for Java to integrate with the [AWS Marketplace Metering Service](#) MeterUsage operation. For complete details, see [MeterUsage Java examples](#). Many of the following steps apply regardless of the language.

Example: AWS Marketplace Metering Service integration

1. Sign in to the [AWS Marketplace Management Portal](#).
2. From **Assets**, choose **Containers** to start creating a new container product. Creating the product generates the product code for the product to integrate with your container image. For more information about publishing, see [Publishing container products \(legacy\)](#). For information about setting AWS Identity and Access Management (IAM) permissions, see [the section called "AWS Marketplace metering and entitlement API permissions"](#).
3. Download the public [AWS Java SDK](#).

⚠ Important

To call the metering API operations from Amazon Elastic Kubernetes Service (Amazon EKS), you must [use a supported AWS SDK](#) and run on an Amazon EKS cluster running Kubernetes 1.13 or later.

4. Call the `MeterUsage` operation from the task or pod once every hour for each dimension usage. The API operation accepts one metering record for a unique combination of `Dimension`, `Resource`, and `Hour`. The resource is either an Amazon Elastic Container Service (Amazon ECS) task or an Amazon EKS pod.

```
{
  "ProductCode" : "string", // (required)
  "UsageDimension" : "string", // (required)
  "UsageQuantity": int, // (optional) Default is 0. Acceptable value from [0,
2147483647 (INT_MAX)]
  "Timestamp": Date, // (required) Timestamp in UTC. Value can be one hour in the
past.
  "UsageAllocations": List<UsageAllocation> // (optional) UsageAllocations across
1 or more tags.
}
```

📘 Note

It is possible to see transient issues in connecting to the AWS Marketplace Metering Service. AWS Marketplace strongly recommends implementing retries for up to 30 minutes, with exponential back off, to avoid short-term outages or network issues.

5. Rebuild a new version of your container image that includes the `MeterUsage` call, tag the container, and push it to any Docker registry that is compatible with Amazon ECS or Amazon EKS, such as Amazon Elastic Container Registry (Amazon ECR). If you are using Amazon ECR, ensure that the account launching the Amazon ECS task or Amazon EKS pod has permissions on the Amazon ECR repository. Otherwise, the operation fails.
6. Create an [IAM](#) role that grants permission for your container to call `MeterUsage`, as defined in the following code example. You must supply this AWS Identity and Access Management (IAM) role in the [Task Role](#) parameter of the Amazon ECS task or Amazon EKS pod definition.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "aws-marketplace:MeterUsage"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

7. Create an Amazon ECS task or Amazon EKS pod definition that references the container that has integrated with AWS Marketplace and references the IAM role that you created in step 6. If you want to see logging, enable AWS CloudTrail logging in the task definition.
8. Create an Amazon ECS or Amazon EKS cluster to run your task or pod. For more information about creating an Amazon ECS cluster, see [Creating a cluster](#) in the *Amazon Elastic Container Service Developer Guide*. For more information about creating an Amazon EKS cluster (using Kubernetes version 1.1.3.x or later), see [Creating an Amazon EKS Cluster](#).
9. Configure the Amazon ECS or Amazon EKS cluster and launch the Amazon ECS task definition or Amazon EKS pod that you created in step 8, in the us-east-1 AWS Region. It's only during this testing process, before the product is live, that you have to use this Region.
10. When you get a valid response from MeterUsage for each of the dimensions being published for the product, you can begin creating your container product. For questions, contact the [AWS Marketplace Seller Operations](#) team.

MeterUsage Java examples

The following code examples use the AWS SDK for Java and AWS Marketplace Metering Service to call the MeterUsage operation.

The following code example calls the MeterUsage operation without any UsageAllocations.

```
import com.amazonaws.services.marketplacemetering.AWSMarketplaceMetering;
import com.amazonaws.services.marketplacemetering.AWSMarketplaceMeteringClientBuilder;
import com.amazonaws.services.marketplacemetering.model.MeterUsageRequest;
import com.amazonaws.services.marketplacemetering.model.MeterUsageResult;
```

```

import java.util.Date;

public class MeterUsage {
    private static final String PRODUCT_CODE = ".....";
    private final AWSMarketplaceMetering awsMarketplaceMetering;

    public MeterUsage() {
        awsMarketplaceMetering =
AWSMarketplaceMeteringClientBuilder.standard().build();
    }

    /**
     * Submits metering record for a FCP Dimension. The API accepts 1 metering record
     per dimension
     * for a given buyer's resource for a given timestamp hour. Ex. If a buyer is
     running 10 tasks,
     * the API will accepts 1 call to MeterUsage in an hour for a given dimension for
     each running task.
     *
     * @param dimension - FCP dimension name provided during the publishing of the
     product.
     * @param quantity - FCP dimension consumption value for the hour.
     * @param timestamp - Timestamp, in UTC, for which the usage is being reported.
     *
     * Timestamp cant be more than 1 hour in the past.
     *
     * Make sure the timestamp value is not before the start of the
     software usage.
     */
    public void callMeterUsage(String dimension, int quantity, Date timestamp) {
        MeterUsageRequest meterUsageRequest = new MeterUsageRequest()
            .withProductCode(PRODUCT_CODE)
            .withUsageDimension(dimension)
            .withUsageQuantity(quantity)
            .withTimestamp(timestamp);
        MeterUsageResult meterUsageResult =
awsMarketplaceMetering.meterUsage(meterUsageRequest);
    }
}

```

The following code example calls the MeterUsage operation with UsageAllocations.

```

private static String callMeterUsageWithAllocationsByTag(AWSMarketplaceMetering
marketplaceMetering) {
    // Tag Keys for the product

```

```
String tagKey1 = "Key1";
String tagKey2 = "Key2";
String tagKey3 = "Key3";

// 1st Usage Allocation bucket which has two Tags [{Key1, Key1Value1},{Key2,
Key2Value1}]
List<Tag> tagsForUsageAllocation1 = Arrays.asList(new
Tag().withKey(tagKey1).withValue("Key1Value1"),
    new Tag().withKey(tagKey2).withValue("Key2Value1"));
UsageAllocation usageAllocation1 = new UsageAllocation()
    .withTags(tagsForUsageAllocation1)
    .withAllocatedUsageQuantity(20);

// 2nd Usage Allocation bucket which has two Tags [{Key1, Key1Value2},{Key2,
Key2Value1}]
List<Tag> tagsForUsageAllocation2 = Arrays.asList(new
Tag().withKey(tagKey1).withValue("Key1Value2"),
    new Tag().withKey(tagKey2).withValue("Key2Value1"));
UsageAllocation usageAllocation2 = new UsageAllocation()
    .withTags(tagsForUsageAllocation2)
    .withAllocatedUsageQuantity(20);

// 3rd Usage Allocation bucket which has two Tags [{Key1, Key1Value2},{Key2,
Key2Value2},{Key3, Key3Value1}]
List<Tag> tagsForUsageAllocation3 = Arrays.asList(new
Tag().withKey(tagKey1).withValue("Key1Value2"),
    new Tag().withKey(tagKey2).withValue("Key2Value2"),
    new Tag().withKey(tagKey3).withValue("Key3Value1"));
UsageAllocation usageAllocation3 = new UsageAllocation()
    .withTags(tagsForUsageAllocation3)
    .withAllocatedUsageQuantity(15);

// 4th Usage Allocation bucket with no tags
UsageAllocation usageAllocation4 = new UsageAllocation()
    .withAllocatedUsageQuantity(15);

List<UsageAllocation> usageAllocationList = Arrays.asList(usageAllocation1,
    usageAllocation2,
    usageAllocation3,
    usageAllocation4);

MeterUsageRequest meterUsageRequest = new MeterUsageRequest()
    .withProductCode("TestProductCode")
    .withUsageDimension("Dimension1")
```

```
        .withTimestamp(new Date())
        //UsageQuantity value must match with sum of all
AllocatedUsageQuantity
        .withUsageQuantity(70)
        .withUsageAllocations(usageAllocationList);

MeterUsageResult meterUsageResult;
try {
    meterUsageResult = marketplaceMetering.meterUsage(meterUsageRequest);
} catch (Exception e) {
    // Log Error
    throw e;
}

return meterUsageResult.getMeteringRecordId();
}
```

Contract pricing for Container products with AWS License Manager

For container-based products with contract pricing, use AWS License Manager to associate licenses with your product.

AWS License Manager is a license management tool that enables your application to track and update licenses (also known as entitlements) that have been purchased by a customer. This section provides information about how to integrate your product with AWS License Manager. After the integration is complete, you can publish your product listing on AWS Marketplace.

If you're integrating License Manager with an AWS Marketplace for Containers Anywhere product for Amazon EKS Anywhere, Amazon ECS Anywhere, Amazon Elastic Compute Cloud (Amazon EC2), or on-premises infrastructure, follow the instructions in [Integrating an AWS Marketplace for Containers Anywhere product with License Manager](#).

For more information about AWS License Manager, see the [AWS License Manager User Guide](#) and the [AWS License Manager](#) section of the *AWS CLI Command Reference*.

License models

AWS Marketplace integration with AWS License Manager supports two license models:

- [Configurable license model](#)
- [Tiered license model](#)

Configurable license model

The configurable license model (also known as the quantifiable license model) entitles a buyer to a specific quantity of resources after a buyer has procured a license.

You set a pricing dimension and a per unit price. Then, the buyer can choose the quantity of the resources that they want to purchase.

Example of pricing dimension and per unit price

You can set a pricing dimension (such as data backup) and per unit price (such as \$30/unit).

The buyer can choose to purchase 5, 10, or 20 units.

Your product tracks and meters usage to measure the quantity of resources consumed.

With the configuration model, the entitlements are counted in one of two ways:

- [Drawdown licenses](#)
- [Floating licenses](#)

Drawdown license

The license is drawn from the pool of allowed amount of licenses upon use. That entitlement is checked out permanently and can't be returned to the license pool.

Example of processing a limited amount of data

A user is entitled to process 500 GB of data. As they continue to process data, the quantity is drawn from the pool of 500 GB until all 500 GB licenses are consumed.

For drawdown licenses, you can use the `CheckoutLicense` API operation to check out license units (entitlements) that are consumed.

Example of backup to S3 for a number of units/year

You have a storage product that allows backup to Amazon Simple Storage Service (Amazon S3) for up to 1,024 units for data for one year. Your application can be launched by using multiple Amazon EC2 instances. Your application has a mechanism to track and aggregate data. Your software calls the `CheckoutLicense` API operation with the Product ID upon every backup or at fixed intervals to update the consumed quantities.

In this example, your software calls the CheckoutLicense API operation to check out 10 units of data. When the total capacity reaches the backup limit that the customer has purchased, the API call fails.

Request

```
linux-machine ~]$ aws license-manager checkout-license \
--product-sku "2205b290-19e6-4c76-9eea-377d6bf71a47" \
--checkout-type "PERPETUAL" \
--key-fingerprint "aws:294406891311:AWS/Marketplace:issuer-fingerprint" \
--entitlements "Name=DataConsumption, Value=10, Unit=Count" \
--client-token "AKIAIOSFODNN7EXAMPLE"
```

Response

```
{"CheckoutType": "PERPETUAL",
"EntitlementsAllowed": [{
"Name": "IntermediateTier",
"Units": "None"
}],
"Expiration": "2021-04-22T19:02:36",
"IssuedAt": "2021-04-22T18:02:36",
"LicenseArn": "arn:aws:license-manager::294406891311:license:l-16bf01b...",
"LicenseConsumptionToken": "AKIAIOSFODNN7EXAMPLE"
}
```

Floating licenses

The license is returned to the pool of the allowed amount of licenses after use.

For floating licenses, the application checks out entitlements from the entitlements pool using the CheckoutLicense API operation when the resource is being used. The response of the CheckoutLicense API operation includes a license consumption token which is a unique identifier for the checkout. The license consumption token can be used to perform additional actions on the entitlements checked out, such as checking them back into the license or extending the checkout.

To check the entitlement back into the pool, use the CheckInLicense API operation when the resource is no longer in use.

```
aws license-manager check-in-license --license-consumption-token
"f1603b3c1f574b7284db84..."
```


In case of failure to check in the entitlement (in case the application crashed), the entitlement checks back into the pool automatically after 60 minutes. If the resource is in use longer than 60 minutes, it is a best practice to keep the entitlement checked out of the pool by using the `ExtendLicenseConsumption` API operation as long as the resource is being used.

```
aws license-manager extend-license-consumption --license-consumption-token
"f1603b3c1f574b7284..."
```

Example of number of users from a fixed upper limit

A user is entitled to 500 simultaneous users on the application. As users log in and log out, the users are drawn and returned to the pool of 500 users. However, the application can't draw more than 500 users from the pool because 500 simultaneous users is the fixed upper limit.

For floating entitlements, you can use the `CheckInLicense` API operation to return the license units to the entitlement pool.

Example of number of concurrent users for one year

Your product is priced based on number of concurrent users. The customer purchases a license for 10 users for one year. The customer launches the software by providing AWS Identity and Access Management (IAM) permissions. When a user logs in, your application calls the `CheckoutLicense` API operation to reduce the quantity by 1. When the user logs out, the application returns that license to the pool by calling the `CheckInLicense` API operation. If you don't call `CheckInLicense`, the license unit will be automatically checked in after 1 hour.

Note

In the following Request, the key-`fingerprint` isn't a placeholder value but the actual value of the fingerprint with which all licenses will be published.

Request

```
aws license-manager checkout-license\
--product-sku "2205b290-19e6-4c76-9eea-377d6bf71a47" \
--checkout-type "PROVISIONAL" \
--key-fingerprint "aws:294406891311:AWS/Marketplace:issuer-fingerprint" \
--entitlements "Name=ReadOnlyUSers, Value=10, Unit=Count" \
```

```
--client-token "AKIAIOSFODNN7EXAMPLE"
```

Response

```
{
  "CheckoutType": "PROVISIONAL",
  "EntitlementsAllowed": [
    {
      "Name": "ReadOnlyUsers",
      "Count": 10,
      "Units": "Count",
      "Value": "Enabled"
    }
  ],
  "Expiration": "2021-04-22T19:02:36",
  "IssuedAt": "2021-04-22T18:02:36",
  "LicenseArn": "arn:aws:license-manager::294406891311:license:l-16bf01b...",
  "LicenseConsumptionToken": "AKIAIOSFODNN7EXAMPLE"
}
```

Tiered license model

The tiered license model entitles a buyer to a specific level, or tier, of application features after a buyer has procured a license.

You create tiers for your product, such as Basic, Intermediate, and Premium. The buyer then selects one of the predefined tiers.

The application doesn't need to track or meter usage of the application.

With the tiered license model, the entitlements aren't counted but instead signify a tier of service that was procured by the customer.

If you want to offer bundled features together, tiers are preferable.

Example of Basic, Intermediate, and Premium tiers

A customer can sign a contract for one of three possible tiers of the software: Basic, Intermediate, or Premium. Each of these tiers has its own pricing. Your software can identify the tier that the customer has signed up for by invoking the `CheckoutLicense` API operation and specifying all possible tiers in the request.

The response of the request contains the entitlement corresponding to the tier that the customer has procured. Based on this information, the software can provision the appropriate customer experience.

Request

```
linux-machine ~]$ aws license-manager checkout-license\  
--product-sku "2205b290-19e6-4c76-9eea-377d6bf71a47" \  
--checkout-type "PROVISIONAL" \  
--key-fingerprint "aws:294406891311:AWS/Marketplace:issuer-fingerprint" \  
--entitlements "Name=BasicTier, Unit=None" "Name=IntermediateTier, Unit=None" \  
"Name=PremiumTier, Unit=None"
```

Response

```
{  
  "CheckoutType": "PROVISIONAL",  
  "EntitlementsAllowed": [  
    {  
      "Name": "IntermediateTier",  
      "Units": "None"  
    }  
  ],  
  "Expiration": "2021-04-22T19:02:36",  
  "IssuedAt": "2021-04-22T18:02:36",  
  "LicenseArn": "arn:aws:license-manager::294406891311:license:l-16bf01b...",  
  "LicenseConsumptionToken": "AKIAIOSFODNN7EXAMPLE"  
}
```

AWS License Manager integration prerequisites

Before publishing the product, you must do the following:

1. Create a new container product in the AWS Marketplace Management Portal, and make a note of its product code.

For more information, see [Creating a container product](#).

2. Fill out the product load form (PLF) with the necessary price information, and return it to us for processing.

For more information, see [Creating or updating pricing details for container products \(legacy\)](#).

3. Use an IAM role for the task or pod running your application with the IAM permissions necessary to call the `CheckoutLicense`, `ExtendLicenseConsumption`, and `CheckInLicense` API operations.

The required IAM permissions are detailed in the following IAM policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "license-manager:CheckoutLicense",
        "license-manager:GetLicense",
        "license-manager:CheckInLicense",
        "license-manager:ExtendLicenseConsumption",
        "license-manager:ListReceivedLicenses"
      ],
      "Resource": "*"
    }
  ]
}
```

4. Make a test call to the `RegisterUsage` API operation with a record for all of the pricing dimensions you define.

Integrating a container product with License Manager

To integrate your container-based product with License Manager

1. Set IAM permissions to call License Manager. For more information, see [AWS License Manager integration prerequisites](#).
2. Download the AWS SDK.

Note

Don't configure AWS credentials within your software. AWS credentials for the buyer are automatically obtained at runtime when your container is running within an Amazon EC2 instance, Amazon ECS task, or Amazon EKS pod.

3. Add license checks to your product.

Your product can call the `CheckoutLicense` API operation wherever the license check should be performed. To check the license, your product must know:

1. The trusted issuer of the license (AWS Marketplace)
2. The application's Product SKU (Product ID)
3. The entitlement to check for this application

The API calls vary based on what kind of pricing licenses you set up.

4. Publish your product listing on AWS Marketplace.

License Manager API operations

To manage the licenses stored in the customer's License Manager account, your software can use the following API operations:

- `GetLicense` – An API that the software can query. It retrieves the status of a purchased license (i.e. expired or expiring soon) and sends a status notification to the customer.
- `CheckoutLicense` – Discovers licenses that the user has purchased. You can also use the `CheckoutLicense` API operation to update the license quantity when the user has consumed some quantity of licenses. With `CheckoutLicense`, you can keep checking out the quantities of licenses used by the customer. When the customer exhausts all the licenses, this call returns an error. For information about the suggested cadence to run `CheckoutLicense`, see [the section called “License renewals and upgrades”](#).
- `ExtendLicenseConsumption` – In case of floating dimensions, when the software checks out a license, the license will return to the pool automatically after 60 minutes. If you want to extend the time the license remains checked out, use the `ExtendLicenseConsumption` API operation to extend the license for another 60 minutes.
- `CheckInLicense` – In case of floating dimensions, when you want to return the license to the entitlement pool, use the `CheckInLicense` API operation.
- `ListReceivedLicenses` API – Lists licenses purchased by the buyer.

License renewals and upgrades

Customers can renew or upgrade their licenses on the AWS Marketplace Management Portal. After they make an additional purchase, AWS Marketplace generates a new version of the license that reflects the new entitlements. Your software reads the new entitlements by using the same API operations. You don't have to do anything different in terms of License Manager integration to handle renewals and upgrades.

Due to license renewals, upgrades, cancellations, and so on, we recommend that your product calls the `CheckoutLicense` API operation at a regular cadence while the product is in use. By using the `CheckoutLicense` API operation at a regular cadence, the product can detect changes in entitlements such as upgrades and expiry.

We recommend that you perform the `CheckoutLicense` API call every 15 minutes.

Integrating an AWS Marketplace for Containers Anywhere product with License Manager

Follow these instructions to integrate AWS License Manager with an AWS Marketplace for Containers Anywhere product for Amazon EKS Anywhere, Amazon ECS Anywhere, Amazon EC2, or on-premises infrastructure.

For general information about the License Manager integration with AWS Marketplace, including available license models, see [Contract pricing for Container products with AWS License Manager](#). For more information about AWS License Manager, see the [AWS License Manager User Guide](#) and the [AWS License Manager](#) section of the *AWS CLI Command Reference*.

Integrating an AWS Marketplace for Containers Anywhere product with License Manager

Use the following instructions to integrate your AWS Marketplace for Containers Anywhere product with AWS License Manager.

To integrate your AWS Marketplace for Containers Anywhere product with License Manager


1. Open a web browser and sign into the [AWS Marketplace Management Portal](#).
2. Create a product ID for your container product by performing the following steps. You will use this ID in your container image for license checks in a later step.
 - a. From the menu bar, expand **Assets**, and choose **Container**.

- b. Enter a customer-facing name for your product, and choose **Create**. You can change this name later.
- c. Make a note of the **Product ID**. You will use it when you create or update the product pricing details.

 Tip

If you lose your product ID, you can find it in the AWS Marketplace Management Portal by choosing **Container** from the **Assets** menu. The **Containers** page shows a list of your products with their associated product IDs.

3. Download the latest public AWS SDK and then install it in your container application. You can find installation instructions for your preferred AWS SDK at [Tools to Build on AWS](#).

 Note

To call the License Manager API operations from Amazon EKS Anywhere or a Kubernetes cluster that isn't provided by AWS, you must use a supported AWS SDK. To view a list of supported AWS SDKs, see [Using a supported AWS SDK](#).

4. Create an AWS License Manager client with a custom credential provider so that it can provide credentials to the container application deployed on AWS as well as on-premises. For complete source code for a custom credential provider, `LicenseCredentialProvider`, see the following sections:
 - [LicenseManagerCredentialsProvider - Java implementation](#)
 - [LicenseManagerCredentialsProvider - Golang implementation](#)

`LicenseCredentialProvider` extends the AWS SDK's default credential provider chain for on-premises use by adding `LicenseManagerTokenCredentialsProvider`. This provides credentials by using License Manager OIDC issued identity tokens in on-premises environments. You must include the source code for `LicenseCredentialProvider` in your application classpath.

Note

Extending the `DefaultCredentialsProvider` allows the same container application to obtain credentials when running on AWS and when running in an on-premises environment. If the container application already uses a custom credential provider chain instead of the default, it can also be extended by adding `LicenseManagerTokenCredentialsProvider` to the custom chain.

The following code snippet is an example of creating an AWS License Manager client using Java.

```
LicenseManagerClientBuilder clientBuilder =  
    LicenseManagerClient.builder().credentialsProvider(LicenseCredentialsProvider.create());
```

5. Call the `CheckoutLicense` API operation by using the `aws license-manager checkout-license` command from each paid container image in your product offering. This checks that the buyer is entitled to use a license for your application. If the buyer is entitled to the application, `CheckoutLicense` succeeds and returns the requested entitlements and their values. If the buyer isn't entitled to the application, `CheckoutLicense` throws an exception.

The following parameters are required when calling the `CheckoutLicense` API operation:

- `CheckoutType` – The valid values are `PROVISIONAL` or `PERPETUAL`:
 - Use `PERPETUAL` when the quantity of entitlements checked out will be exhausted from the pool.

Example: Buyer is entitled to process 500 GB of data. As they continue to process data, the quantity is drawn down and exhausted from the pool of 500 GB.

- Use `PROVISIONAL` for floating license entitlements where the entitlements are checked out of the pool and returned after use.

Example: User is entitled to 500 simultaneous users on the application. As users log in or log out, the users are drawn or returned to the pool of 500 users. To learn more about floating license entitlements, see [Floating license entitlements with License Manager](#).

- `ClientToken` – A unique, case-sensitive identifier. We recommend using a random UUID for each unique request.

- Entitlements – A list of entitlements to be checked out.
 - For feature entitlements, provide the Name and Unit properties as follows.

```
{
  "Name": "<Entitlement_Name>",
  "Unit": "None"
}
```

- For counted entitlements, provide the Name, Unit, and Count properties as follows.

```
{
  "Name": "<Entitlement_Name>",
  "Unit": "<Entitlement_Unit>",
  "Value": <Desired_Count>
}
```

- KeyFingerprint – The key fingerprint for licenses issued by AWS Marketplace is `aws:294406891311:AWS/Marketplace:issuer-fingerprint`. Using this key fingerprint ensures that the license is issued by AWS Marketplace and not by an unreliable entity.
- ProductSKU – The Product ID generated on AWS Marketplace Management Portal in previous steps.

The following snippet is an example of a call using the CheckoutLicense API operation using the AWS CLI.

```
aws license-manager checkout-license \
--product-sku "2205b290-19e6-4c76-9eea-377d6bf71a47" \
--checkout-type "PROVISIONAL" \
--client-token "79464194dca9429698cc774587a603a1" \
--entitlements "Name=AWS::Marketplace::Usage/Drawdown/DataConsumption, Value=10,
Unit=Gigabytes" \
--key-fingerprint "aws:294406891311:AWS/Marketplace:issuer-fingerprint"
```

Note

To check licenses, container applications require outbound network access to use License Manager. Applications deployed on-premises might experience unreliable or slow outbound network access. These applications should include adequate retries

when calling License Manager. For more information, see [Best practices for integrating with License Manager for on-premises deployments](#).

6. Call the CheckoutLicense API operation at a regular cadence to identify any changes to customers' licenses due to renewals, upgrades, or cancellations made on AWS Marketplace. The cadence depends on the application. We recommend checking licenses once a day to pick up changes automatically without any buyer intervention.

An application deployed on-premises might have unreliable outbound network access to check licenses on a regular cadence. In such cases, the application should use a cached licenses for sufficient resiliency. For more information, see [Best practices for integrating with License Manager for on-premises deployments](#).

7. After you integrate the CheckoutLicense call with your container application, build a new version of your Docker container image with the changes.
8. Update your application's Helm chart to accept a Kubernetes secret as optional input that contains configuration to access licenses using License Manager APIs. The configuration secret will contain an identity token issued by License Manager and an AWS Identity and Access Management role which will be used by the custom credential provider described previously to get AWS credentials for calling License Manager APIs when the container application is deployed on-premises. Also, add the AWS Region as an input with a default value of us-east-1.

Buyers deploying the container application on-premises can create the Kubernetes secret through the AWS Marketplace buyer experience for container products. Provide the Kubernetes secret name as input to the `helm install` command. The configuration secret is configured in the following format.

```
apiVersion: v1
kind: Secret
metadata:
  name: aws-marketplace-license-config
type: Opaque
stringData:
  license_token: <token_value> // License Manager issued JWT token
  iam_role: <role_arn> // AWS Identity and Access Management role to assume with
  license token
```

9. Update the application deployment template in the Helm chart for container images integrated with AWS License Manager to include the following:

- Service account for pod – The service account is required for Helm deployments on Amazon EKS. It's used to get permissions to call License Manager API operations by setting up IAM roles for the service account on the container image. For more information about IAM roles for service accounts, see [IAM roles for service accounts](#).
- License access for on-premises deployments – The license configuration secret is required to provide credentials and appropriate permissions to call License Manager API operations for Helm deployments in on-premises environments. Buyers will generate and provide the license secret to Helm from the AWS Marketplace buyer experience.

The following code snippet is a sample deployment specification with the service account, license configuration, and image pull secret.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: example-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: example-app
  template:
    metadata:
      labels:
        app: example-app
spec:
  // Service account for pod
  serviceAccountName: {{ .Values.serviceAccountName }}
  containers:
    - name: example-app
      image: example-app
      ports:
        - containerPort: 8001
  // Add the following conditional attributes
  {{ - if .Values.awsmpl.licenseConfigSecretName }}
    //Mount the license volume to the container image
    volumeMounts:
      - name: awsmpl-product-license
        mountPath: "/var/run/secrets/product-license"
  //Add following environment variable to container for credential
```

```

provider
  env:
    - name: AWS_WEB_IDENTITY_REFRESH_TOKEN_FILE
      value: "/var/run/secrets/product-license/license_token"
    - name: AWS_ROLE_ARN
      valueFrom:
        secretKeyRef:
          name: {{ .Values.aws.licenseConfigSecretName }}
          key: iam_role
  //Mount the license secret as a volume to the pod
  volumes:
    - name: awsmc-product-license
      secret:
        secretName: {{ .Values.aws.licenseConfigSecretName }}
        optional: true
{{ - end }}

```

Note

The license configuration secret is optional. Buyers only use the value for on-premises deployments. For AWS deployments, the deployment specification must include a service account for the License Manager integrated images.

10. Test the License Manager integration locally and on Amazon EKS by performing the steps in the following sections:
 - a. [Testing License Manager integration locally](#)
 - b. [Testing License Manager integration on Amazon EKS](#)
11. After you successfully verify License Manager integration both on AWS and on-premises, you can create your container product listing by following the steps in [Creating a container product](#).

Testing License Manager integration locally

You can use minikube or any other setup to test License Manager integration on any Kubernetes cluster locally. Make sure that the Kubernetes cluster has outbound internet access to call License Manager API operations.

To test a License Manager integration locally

1. Create a test license in a test seller account with desired entitlements. To set up a test license, see [CreateLicense](#) in the *AWS License Manager API Reference*. Or, use the following script to create a test license and then create a license grant to a test buyer account to consume the license. The following script uses test seller account credentials.

```

read -p 'AWS Account for test buyer: ' TEST_BUYER_ACCOUNT_ID
read -p 'License entitlements: ' ENTITLEMENTS

# TEST_SELLER_ACCOUNT_ID="109876543210"
# ENTITLEMENTS="{\"Name\": \"ByData\", \"MaxCount\": 1000, \"Overage\": true, \"Unit\": \"Gigabits\", \"AllowCheckIn\": true}"

# Create License

NOW=$(date +%Y-%m-%dT00:00:00+00:00)

PRODUCT_NAME="My awesome product"
PRODUCT_SKU="c97b7825-44c4-4f42-b025-12baa4c171e0"

LICENSE_BENEFICIARY=" arn:aws:iam::$TEST_BUYER_ACCOUNT_ID:root "
LICENSE_ISSUER_NAME="test-seller"
LICENSE_NAME="test-seller-license"

CLIENT_TOKEN="b3920968-a94f-4547-af07-3dd232319367"
CONSUMPTION_TTL=180
CONSUMPTION_RENEW_TYPE="None"

HOME_REGION="us-east-1"

LICENSE_ARN=$(aws license-manager create-license --license-name
"$LICENSE_NAME" --product-name "$PRODUCT_NAME" --product-sku
"$PRODUCT_SKU" --issuer Name="$LICENSE_ISSUER_NAME" --home-region
"$HOME_REGION" --validity Begin="$NOW" --entitlements "$ENTITLEMENTS"
--beneficiary "$LICENSE_BENEFICIARY" --consumption-configuration
RenewType="$CONSUMPTION_RENEW_TYPE",ProvisionalConfiguration={MaxTimeToLiveInMinutes=$CONSUMPTION_TTL} --client-token "$CLIENT_TOKEN" | jq -r ".LicenseArn" )

echo "License arn: $LICENSE_ARN"

# Create Grant

```

```

GRANT_TOKEN="e9a14140-4fca-4219-8230-57511a6ea6"
GRANT_NAME="test-grant"

GRANT_ARN=$(aws license-manager create-grant --grant-name "$GRANT_NAME"
  --license-arn "$LICENSE_ARN" --principals "$LICENSE_BENEFICIARY" --home-
  region "$HOME_REGION" --client-token "$GRANT_TOKEN" --allowed-operations
  "CheckoutLicense" "CheckInLicense" "ExtendConsumptionLicense" "CreateToken" | jq -
  r ".GrantArn")

echo "Grant arn: $GRANT_ARN"

```

2. Create a Kubernetes secret with the license token and IAM role using the secret format defined previously. Use the License Manager CreateToken API operation to generate a license token. Then, use the IAM CreateRole API operation to create an IAM role with permissions and a trust policy. See the example in the following script. The following script uses test buyer account credentials.

```

read -p 'AWS Account for test license: ' TEST_ACCOUNT_ID
read -p 'License Arn' LICENSE_ARN
# Create IAM Role
ROLE_NAME="AWSLicenseManagerConsumptionTestRole"
ROLE_DESCRIPTION="Role to test AWS License Manager integration on-prem"
ROLE_POLICY_ARN="arn:aws:iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy"
ROLE_TRUST_POLICY="{\"Version\": \"2012-10-17\", \"Statement\": [{ \"Effect\":
  \"Allow\", \"Principal\": { \"Federated\": \"openid-license-manager.amazonaws.com
  \" }, \"Action\": \"sts:AssumeRoleWithWebIdentity\", \"Condition\":
  { \"ForAnyValue:StringLike\": { \"openid-license-manager.amazonaws.com:amr\":
  \"aws:license-manager:token-issuer-account-id:${TEST_ACCOUNT_ID}\" } } ] }"
ROLE_SESSION_DURATION=3600

ROLE_ARN=$(aws iam create-role --role-name "$ROLE_NAME" --description
  "$ROLE_DESCRIPTION" --assume-role-policy-document "$ROLE_TRUST_POLICY" --max-
  session-duration $ROLE_SESSION_DURATION | jq ".Role" | jq -r ".Arn")

aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn "$ROLE_POLICY_ARN"

echo "Role arn: $ROLE_ARN"

# Create Token
CLIENT_TOKEN="b3920968-a94f-4547-af07-3dd232319367"

```

```
TOKEN=$(aws license-manager create-token --license-arn $LICENSE_ARN --role-arns
$ROLE_ARN --client-token $CLIENT_TOKEN | jq '.Token')

echo "License access token: $TOKEN"c
```

3. Set up any Kubernetes cluster hosted outside AWS. Use it to test that the container applications can connect to the AWS License Manager API from environments other than AWS and that the custom credential provider is well integrated in the application.
4. Deploy the license token and IAM role generated previously into the local Kubernetes cluster.

```
kubectl create secret generic "awsmp-license-access-config" \
--from-literal=license_token=${TOKEN} \
--from-literal=iam_role=${ROLE_ARN}
```

5. Deploy your application through Helm with the secret name as input and verify that the application can call License Manager API operations to perform entitlement checks. For Helm and deployment specification changes, refer to Step 9 in [Integrating an AWS Marketplace for Containers Anywhere product with License Manager](#).

Testing License Manager integration on Amazon EKS

You can also test License Manager integration on Amazon EKS. Test to make sure that the application can call License Manager API operations without the license configuration secret. Also make sure that the service account can be used to set up IAM Roles for Service Accounts (IRSA) and provide relevant credentials to the application.

To test a License Manager integration on Amazon EKS

1. Create a test license in a test seller account with the desired entitlements. See [CreateLicense API reference](#) to set up your test license or use the following script to create one and create a license grant to a test buyer account to consume the license. The following script uses test seller account credentials.

```
read -p 'AWS Account for test buyer: ' TEST_BUYER_ACCOUNT_ID
read -p 'License entitlements: ' ENTITLEMENTS

# TEST_SELLER_ACCOUNT_ID="109876543210"
# ENTITLEMENTS="{\"Name\": \"ByData\", \"MaxCount\": 1000, \"Overage\": true, \"Unit\": \"Gigabits\", \"AllowCheckIn\": true}"
```

```

# Create License

NOW=$(date +"%Y-%m-%dT00:00:00+00:00")

PRODUCT_NAME="My awesome product"
PRODUCT_SKU="c97b7825-44c4-4f42-b025-12baa4c171e0"

LICENSE_BENEFICIARY=" arn:aws:iam::$TEST_BUYER_ACCOUNT_ID:root "
LICENSE_ISSUER_NAME="test-seller"
LICENSE_NAME="test-seller-license"

CLIENT_TOKEN="b3920968-a94f-4547-af07-3dd232319367"
CONSUMPTION_TTL=180
CONSUMPTION_RENEW_TYPE="None"

HOME_REGION="us-east-1"

LICENSE_ARN=$(aws license-manager create-license --license-name
"$LICENSE_NAME" --product-name "$PRODUCT_NAME" --product-sku
"$PRODUCT_SKU" --issuer Name="$LICENSE_ISSUER_NAME" --home-region
"$HOME_REGION" --validity Begin="$NOW" --entitlements "$ENTITLEMENTS"
--beneficiary "$LICENSE_BENEFICIARY" --consumption-configuration
RenewType="$CONSUMPTION_RENEW_TYPE",ProvisionalConfiguration={MaxTimeToLiveInMinutes=$
CONSUMPTION_TTL} --client-token "$CLIENT_TOKEN" | jq -r ".LicenseArn" )

echo "License arn: $LICENSE_ARN"

# Create Grant

GRANT_TOKEN="e9a14140-4fca-4219-8230-57511a6ea6"
GRANT_NAME="test-grant"

GRANT_ARN=$(aws license-manager create-grant --grant-name "$GRANT_NAME"
--license-arn "$LICENSE_ARN" --principals "$LICENSE_BENEFICIARY" --home-
region "$HOME_REGION" --client-token "$GRANT_TOKEN" --allowed-operations
"CheckoutLicense" "CheckInLicense" "ExtendConsumptionLicense" "CreateToken" | jq -
r ".GrantArn")

echo "Grant arn: $GRANT_ARN"

```

2. Create a test Amazon EKS cluster of desired configurations, or run the following commands to use a default configuration.


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

```
eksctl create cluster \  
--name awsmpt-eks-test-example \  
--region us-west-2 \  
--with-oidc \  
--ssh-access \  
--ssh-public-key eks-key-pair
```

3. Create a service account for an existing cluster and associate it with an IAM role. The following command creates an IAM role with the `AWSLicenseManagerConsumptionPolicy`. Then, the command attaches it to the `test_sa` service account of the Amazon EKS cluster where the License Manager integrated images should be deployed. As a result, the service account can get appropriate credentials to call License Manager API operations.

```
eksctl create iamserviceaccount \  
--name test_sa \  
--namespace test_namespace \  
--cluster awsmpt-eks-test-example \  
--attach-policy-arn "arn:aws:iam::aws:policy/service-role/  
AWSLicenseManagerConsumptionPolicy" \  
--approve \  
--override-existing-serviceaccounts
```

4. Deploy the application through Helm in the service account where the IAM role is associated from the previous command. Verify that the application can call License Manager API operations to perform entitlement checks.

Floating license entitlements with License Manager

With floating licenses, as users log into the application, a license is drawn from the pool of available licenses. As users log out, the licenses are added back to the pool of available licenses.

For floating licenses, the application uses the `CheckoutLicense` API operation to check out entitlements from the entitlements pool when the resource is being used. The response of the `CheckoutLicense` API operation includes a license consumption token which is a unique identifier for the checkout. The license consumption token can perform additional actions on the entitlements that are checked out, such as checking them back into the license pool or extending the checkout.

When the resource is no longer in use, the application uses the `CheckInLicense` API operation to check the entitlement back into the pool.

```
aws license-manager check-in-license \  
--license-consumption-token "f1603b3c1f574b7284db84a9e771ee12"
```

If checking a license back into the pool fails, for example, if the application crashes during the operation, the entitlement is checked back into the pool automatically after 60 minutes. Because of this, if the resource is in use longer than 60 minutes, it's a best practice to keep the entitlement checked out of the pool. To do this, use the `ExtendLicenseConsumption` API operation as long as the resource is being used.

```
aws license-manager extend-license-consumption \  
--license-consumption-token "f1603b3c1f574b7284db84a9e771ee12"
```

Best practices for integrating with License Manager for on-premises deployments

Container application deployments in an on-premises environment might encounter unreliable outbound network access. Use the following best practices to add resiliency to avoid service disruption to buyers due to potential issues caused by poor internet connectivity:

- **Adequate retry** – Transient network issues can keep your application from connecting to AWS License Manager. Implement retries for up to 30 minutes, with exponential back off. This can help avoid short-term outages or network issues.
- **Avoid hard limit** – Applications deployed in connected clusters can regularly check licenses to identify any changes due to upgrades or renewals. With unreliable outbound access, the application might not be able to identify those changes. Whenever possible, the application should avoid disruption of service to buyers due to inability to check licenses through License Manager. Applications can fall back on a free-trial or open-source experience when the license expires and they can't check if a license is valid.
- **Notify customers** – When using a cached license, any changes to the license (including renewal or upgrades) are not automatically reflected on the running workload. Notify your customers (that they must allow outbound access to the application again temporarily so the application can update its cached license. For example, notify customers through the application itself or through its documentation. Similarly, when falling back to a lower set of functionalities, notify customers that their entitlements are exhausted or the license is expired. Then, they can choose to either upgrade or renew.

LicenseManagerCredentialsProvider - Java implementation

LicenseCredentialsProvider extends the AWS SDK's default credential provider chain for on-premises use by adding LicenseManagerTokenCredentialsProvider.

LicenseCredentialsProvider

```
package com.amazon.awsmp.license;

import software.amazon.awssdk.auth.credentials.AwsCredentials;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProviderChain;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.auth.credentials.internal.LazyAwsCredentialsProvider;
import software.amazon.awssdk.utils.SdkAutoCloseable;

public class LicenseCredentialsProvider implements AwsCredentialsProvider,
    SdkAutoCloseable {
    private static final LicenseCredentialsProvider CREDENTIALS_PROVIDER = new
    LicenseCredentialsProvider();
    private final LazyAwsCredentialsProvider providerChain;

    private LicenseCredentialsProvider() {
        this.providerChain = createChain();
    }

    public static LicenseCredentialsProvider create() {
        return CREDENTIALS_PROVIDER;
    }

    @Override
    public AwsCredentials resolveCredentials() {
        return this.providerChain.resolveCredentials();
    }

    @Override
    public void close() {
        this.providerChain.close();
    }

    private LazyAwsCredentialsProvider createChain() {
        return LazyAwsCredentialsProvider.create(() -> {
            AwsCredentialsProvider[] credentialsProviders = new
            AwsCredentialsProvider[] {
```

```
        DefaultCredentialsProvider.create(),
        LicenseManagerTokenCredentialsProvider.create()});

    return AwsCredentialsProviderChain.builder().reuseLastProviderEnabled(true)
        .credentialsProviders(credentialsProviders).build();
});
}
}
```

LicenseManagerTokenCredentialsProvider

`LicenseManagerTokenCredentialsProvider` provides credentials by using License Manager OIDC issued identity tokens in on-premises environments. You must include the source code for `LicenseCredentialsProvider` in your application classpath.

```
package com.amazon.awsmp.license;

import software.amazon.awssdk.auth.credentials.AnonymousCredentialsProvider;
import software.amazon.awssdk.auth.credentials.AwsCredentials;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.core.SdkSystemSetting;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicyContext;
import software.amazon.awssdk.core.retry.conditions.OrRetryCondition;
import software.amazon.awssdk.core.retry.conditions.RetryCondition;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.regions.providers.DefaultAwsRegionProviderChain;
import software.amazon.awssdk.services.licensemanager.LicenseManagerClient;
import software.amazon.awssdk.services.licensemanager.model.GetAccessTokenRequest;
import software.amazon.awssdk.services.licensemanager.model.GetAccessTokenResponse;
import software.amazon.awssdk.services.sts.StsClient;
import
    software.amazon.awssdk.services.sts.auth.StsAssumeRoleWithWebIdentityCredentialsProvider;
import software.amazon.awssdk.services.sts.model.AssumeRoleWithWebIdentityRequest;
import software.amazon.awssdk.services.sts.model.IdpCommunicationErrorException;
import software.amazon.awssdk.utils.IoUtils;
import software.amazon.awssdk.utils.SdkAutoCloseable;
import software.amazon.awssdk.utils.StringUtils;
import software.amazon.awssdk.utils.SystemSetting;

import java.io.IOException;
import java.io.InputStream;
import java.io.UncheckedIOException;
```

```
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.function.Supplier;

public class LicenseManagerTokenCredentialsProvider implements AwsCredentialsProvider,
    SdkAutoCloseable {

    private final StsAssumeRoleWithWebIdentityCredentialsProvider credentialsProvider;
    private final RuntimeException loadException;

    private Path licenseAccessTokenFile;
    private String roleArn;
    private String roleSessionName;
    private StsClient stsClient;
    private LicenseManagerClient lmClient;

    public static LicenseManagerTokenCredentialsProvider create() {
        return new Builder().build();
    }

    @Override
    public AwsCredentials resolveCredentials() {
        if (this.loadException != null) {
            throw this.loadException;
        }
        return this.credentialsProvider.resolveCredentials();
    }

    @Override
    public void close() {
        IoUtils.closeQuietly(this.credentialsProvider, null);
        IoUtils.closeQuietly(this.stsClient, null);
        IoUtils.closeIfCloseable(this.lmClient, null);
    }

    private LicenseManagerTokenCredentialsProvider(Builder builder) {
        StsAssumeRoleWithWebIdentityCredentialsProvider credentialsProvider = null;
        RuntimeException loadException = null;

        try {
            this.licenseAccessTokenFile =
                Paths.get(StringUtils.trim(LicenseSystemSetting.AWS_WEB_IDENTITY_REFRESH_TOKEN_FILE.getStringValue()));
        } catch (IOException e) {
            loadException = e;
        }

        this.credentialsProvider = credentialsProvider;
        this.loadException = loadException;
    }
}
```

```
        this.roleArn = SdkSystemSetting.AWS_ROLE_ARN.getStringValueOrThrow();
        this.roleSessionName =
SdkSystemSetting.AWS_ROLE_SESSION_NAME.getStringValue().orElse("aws-sdk-java-" +
System.currentTimeMillis());
        this.stsClient = builder.stsClient != null ? builder.stsClient :
StsClientFactory.create();
        this.lmClient = builder.lmClient != null ? builder.lmClient :
LicenseManagerClientFactory.create();

        AssumeRoleWithWebIdentityRequest request =
AssumeRoleWithWebIdentityRequest.builder()

.roleArn(this.roleArn).roleSessionName(this.roleSessionName).build();

        Supplier<AssumeRoleWithWebIdentityRequest> supplier = new
AssumeRoleRequestSupplier(request,
            this.licenseAccessTokenFile, this.lmClient);

        credentialsProvider =
StsAssumeRoleWithWebIdentityCredentialsProvider.builder()
            .stsClient(this.stsClient).refreshRequest(supplier).build();
    } catch (RuntimeException ex) {
        loadException = ex;
    }

    this.credentialsProvider = credentialsProvider;
    this.loadException = loadException;
}

public static final class Builder {
    private Path licenseAccessTokenFile;
    private String roleArn;
    private String roleSessionName;
    private StsClient stsClient;
    private LicenseManagerClient lmClient;

    public LicenseManagerTokenCredentialsProvider build() {
        return new LicenseManagerTokenCredentialsProvider(this);
    }

    public LicenseManagerTokenCredentialsProvider.Builder
licenseAccessTokenFile(Path licenseAccessTokenFile) {
        this.licenseAccessTokenFile = licenseAccessTokenFile;
        return this;
    }
}
```

```
    }

    public LicenseManagerTokenCredentialsProvider.Builder roleArn(String roleArn) {
        this.roleArn = roleArn;
        return this;
    }

    public LicenseManagerTokenCredentialsProvider.Builder roleSessionName(String
roleSessionName) {
        this.roleSessionName = roleSessionName;
        return this;
    }

    public LicenseManagerTokenCredentialsProvider.Builder stsClient(StsClient
stsClient) {
        this.stsClient = stsClient;
        return this;
    }

    public LicenseManagerTokenCredentialsProvider.Builder
lmClient(LicenseManagerClient lmClient) {
        this.lmClient = lmClient;
        return this;
    }
}

private static final class AssumeRoleRequestSupplier implements Supplier {
    private final LicenseManagerClient lmClient;
    private final AssumeRoleWithWebIdentityRequest request;
    private final Path webIdentityRefreshTokenFile;

    AssumeRoleRequestSupplier(final AssumeRoleWithWebIdentityRequest request,
                             final Path
webIdentityRefreshTokenFile,
                             final LicenseManagerClient lmClient) {

        this.lmClient = lmClient;
        this.request = request;
        this.webIdentityRefreshTokenFile = webIdentityRefreshTokenFile;
    }

    public AssumeRoleWithWebIdentityRequest get() {
        return this.request.toBuilder()
            .webIdentityToken(getIdentityToken())
            .build();
    }
}
```

```
    }

    private String getIdentityToken() {
        return refreshToken(readRefreshToken(this.webIdentityRefreshTokenFile));
    }

    private String readRefreshToken(Path file) {
        try (InputStream webIdentityRefreshTokenStream =
Files.newInputStream(file)) {
            return IoUtils.toUtf8String(webIdentityRefreshTokenStream);
        } catch (IOException e) {
            throw new UncheckedIOException(e);
        }
    }

    private String refreshToken(String licenseRefreshToken) {
        final GetAccessTokenRequest request = GetAccessTokenRequest.builder()
            .token(licenseRefreshToken)
            .build();

        GetAccessTokenResponse response = this.lmClient.getAccessToken(request);
        return response.accessToken();
    }
}

private static final class LicenseManagerClientFactory {
    private static final Duration DEFAULT_API_TIMEOUT = Duration.ofSeconds(30);
    private static final Duration DEFAULT_API_ATTEMPT_TIMEOUT =
Duration.ofSeconds(10);

    public static LicenseManagerClient create() {
        return getLicenseManagerClient();
    }

    private static LicenseManagerClient getLicenseManagerClient() {
        ClientOverrideConfiguration configuration =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(DEFAULT_API_TIMEOUT)
            .apiCallAttemptTimeout(DEFAULT_API_ATTEMPT_TIMEOUT)
            .build();

        LicenseManagerClient client = LicenseManagerClient.builder()
            .region(configureLicenseManagerRegion())
            .credentialsProvider(AnonymousCredentialsProvider.create())
```



```
        .overrideConfiguration(configuration).build();
    return client;
}

private static Region configureLicenseManagerRegion() {
    Region defaultRegion = Region.US_EAST_1;

    Region region;
    try {
        region = (new DefaultAwsRegionProviderChain()).getRegion();
    } catch (RuntimeException ex) {
        region = defaultRegion;
    }
    return region;
}

private static final class StsClientFactory {
    private static final Duration DEFAULT_API_TIMEOUT = Duration.ofSeconds(30);
    private static final Duration DEFAULT_API_ATTEMPT_TIMEOUT =
Duration.ofSeconds(10);

    public static StsClient create() {
        return getStsClient();
    }

    private static StsClient getStsClient() {
        OrRetryCondition retryCondition = OrRetryCondition.create(new
StsRetryCondition(),
            RetryCondition.defaultRetryCondition());

        ClientOverrideConfiguration configuration =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(DEFAULT_API_TIMEOUT)
            .apiCallAttemptTimeout(DEFAULT_API_ATTEMPT_TIMEOUT)
            .retryPolicy(r -> r.retryCondition(retryCondition))
            .build();

        return StsClient.builder()
            .region(configureStsRegion())
            .credentialsProvider(AnonymousCredentialsProvider.create())
            .overrideConfiguration(configuration).build();
    }
}
```

```
private static Region configureStsRegion() {
    Region defaultRegion = Region.US_EAST_1;
    Region stsRegion;
    try {
        stsRegion = (new DefaultAwsRegionProviderChain()).getRegion();
    } catch (RuntimeException ex) {
        stsRegion = defaultRegion;
    }
    return stsRegion;
}

private static final class StsRetryCondition implements RetryCondition {
    public boolean shouldRetry(RetryPolicyContext context) {
        return context.exception() instanceof IdpCommunicationErrorException;
    }
}

private enum LicenseSystemSetting implements SystemSetting {
    AWS_WEB_IDENTITY_REFRESH_TOKEN_FILE("aws.webIdentityRefreshTokenFile");

    private String systemProperty;
    private String defaultValue = null;

    LicenseSystemSetting(String systemProperty) {
        this.systemProperty = systemProperty;
    }

    @Override
    public String property() {
        return this.systemProperty;
    }

    @Override
    public String environmentVariable() {
        return this.name();
    }

    @Override
    public String defaultValue() {
        return this.defaultValue;
    }
}
```

```
}
```

LicenseManagerCredentialsProvider - Golang implementation

LicenseCredentialsProvider

LicenseCredentialsProvider extends the AWS SDK's default credential provider chain for on-premises use by adding LicenseManagerTokenCredentialsProvider.

```
package lib

import (
    "context"
    "fmt"
    "sync"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
)

// LicenseCredentialsProvider is the custom credential provider that can retrieve valid
// temporary aws credentials
type LicenseCredentialsProvider struct {
    fallbackProvider aws.CredentialsProvider
    mux              sync.RWMutex
    licenseCredentials aws.Credentials
    err              error
}

// NewLicenseCredentialsProvider method will create a LicenseCredentialProvider Object
// which contains valid temporary aws credentials
func NewLicenseCredentialsProvider() (*LicenseCredentialsProvider, error) {
    licenseCredentialProvider := &LicenseCredentialsProvider{}
    fallbackProvider, err := createCredentialProvider()
    if err != nil {
        return licenseCredentialProvider, fmt.Errorf("failed to create
LicenseCredentialsProvider, %w", err)
    }
    licenseCredentialProvider.fallbackProvider = fallbackProvider
    return licenseCredentialProvider, nil
}

// Retrieve method will retrieve temporary aws credentials from the credential provider
```

```

func (l *LicenseCredentialsProvider) Retrieve(ctx context.Context) (aws.Credentials,
error) {
    l.mux.RLock()
    defer l.mux.RUnlock()
    l.licenseCredentials, l.err = l.fallBackProvider.Retrieve(ctx)
    return l.licenseCredentials, l.err
}

func createCredentialProvider() (aws.CredentialsProvider, error) {
    // LoadDefaultConfig will examine all "default" credential providers
    ctx := context.TODO()
    cfg, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        return nil, fmt.Errorf("failed to create FallBackProvider, %w", err)
    }

    var useFallbackProvider bool
    if cfg.Credentials != nil {
        if _, err := cfg.Credentials.Retrieve(ctx); err != nil {
            // If the "default" credentials provider cannot retrieve credentials, enable
            fallback to customCredentialsProvider.
            useFallbackProvider = true
        }
    } else {
        useFallbackProvider = true
    }

    if useFallbackProvider {
        customProvider, err := newLicenseManagerTokenCredentialsProvider()
        if err != nil {
            return cfg.Credentials, fmt.Errorf("failed to create fallBackProvider, %w", err)
        }
        // wrap up customProvider with CredentialsCache to enable caching
        cfg.Credentials = aws.NewCredentialsCache(customProvider)
    }
    return cfg.Credentials, nil
}

```

LicenseManagerTokenCredentialsProvider

`LicenseManagerTokenCredentialsProvider` provides credentials by using License Manager OIDC issued identity tokens in on-premises environments. You must include the source code for `LicenseCredentialsProvider` in your application classpath.

```
package lib

import (
    "context"
    "fmt"
    "io/ioutil"
    "os"
    "sync"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/sts"
)

const awsRefreshTokenFilePathEnvVar = "AWS_LICENSE_ACCESS_FILE"

// licenseManagerTokenCredentialsProvider defines and contains
// StsAssumeRoleWithWebIdentityProvider
type licenseManagerTokenCredentialsProvider struct {
    stsCredentialProvider *stsAssumeRoleWithWebIdentityProvider
    mux                   sync.RWMutex
    licenseCredentials    aws.Credentials
    err                   error
}

// Retrieve method will retrieve credentials from credential provider.
// Make this method public to make this provider satisfies CredentialProvider interface
func (a *licenseManagerTokenCredentialsProvider) Retrieve(ctx context.Context)
    (aws.Credentials, error) {
    a.mux.RLock()
    defer a.mux.RUnlock()
    a.licenseCredentials, a.err = a.stsCredentialProvider.Retrieve(ctx)
    return a.licenseCredentials, a.err
}

// newLicenseManagerTokenCredentialsProvider will create and return
// a LicenseManagerTokenCredentialsProvider Object which wraps up
// stsAssumeRoleWithWebIdentityProvider
func newLicenseManagerTokenCredentialsProvider()
    (*licenseManagerTokenCredentialsProvider, error) {
    // 1. Retrieve variables From yaml environment
    envConfig, err := config.NewEnvConfig()
```

```
if err != nil {
    return &licenseManagerTokenCredentialsProvider{}, fmt.Errorf("failed to create
LicenseManagerTokenCredentialsProvider, %w", err)
}
roleArn := envConfig.RoleARN
var roleSessionName string
if envConfig.RoleSessionName == "" {
    roleSessionName = fmt.Sprintf("aws-sdk-go-v2-%v", time.Now().UnixNano())
} else {
    roleSessionName = envConfig.RoleSessionName
}
tokenFilePath := os.Getenv(awsRefreshTokenFilePathEnvVar)
b, err := ioutil.ReadFile(tokenFilePath)
if err != nil {
    return &licenseManagerTokenCredentialsProvider{}, fmt.Errorf("failed to create
LicenseManagerTokenCredentialsProvider, %w", err)
}
refreshToken := aws.String(string(b))

// 2. Create stsClient
cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    return &licenseManagerTokenCredentialsProvider{}, fmt.Errorf("failed to create
LicenseManagerTokenCredentialsProvider, %w", err)
}
stsClient := sts.NewFromConfig(cfg, func(o *sts.Options) {
    o.Region = configureStsClientRegion(cfg.Region)
    o.Credentials = aws.AnonymousCredentials{}
})

// 3. Configure StsAssumeRoleWithWebIdentityProvider
stsCredentialProvider := newStsAssumeRoleWithWebIdentityProvider(stsClient, roleArn,
roleSessionName, refreshToken)

// 4. Build and return
return &licenseManagerTokenCredentialsProvider{
    stsCredentialProvider: stsCredentialProvider,
}, nil
}

func configureStsClientRegion(configRegion string) string {
    defaultRegion := "us-east-1"
    if configRegion == "" {
        return defaultRegion
    }
}
```

```
} else {  
  return configRegion  
}  
}
```

Amazon SNS notifications for container products

To receive notifications, you subscribe to the AWS Marketplace Amazon Simple Notification Service (Amazon SNS) topics provided to you during product creation. The topics provide notifications about changes to customers' subscriptions for your products. For example, you can use this to know when customers accept a private offer.

Note

During the product creation process, you'll receive the actual Amazon Resource Name (ARN) to the SNS topic. For example: `arn:aws:sns:us-east-1:123456789012:aws-mp-subscription-notification-PRODUCTCODE`

The following Amazon SNS topic is available for container products:

- [Amazon SNS topic: `aws-mp-subscription-notification`](#) – This topic notifies you when a buyer subscribes or unsubscribes to a product. This is available for hourly pricing models, including hourly and hourly with long term.

Amazon SNS topic: `aws-mp-subscription-notification`

Each message in the `aws-mp-subscription-notification` topic has the following format.

```
{  
  "action": "<action-name>",  
  "customer-identifier": " X01EXAMPLEX",  
  "product-code": "n0123EXAMPLEXXXXXXXXXXXXX",  
  "offer-identifier": "offer-abcexample123",  
  "isFreeTrialTermPresent": "true"  
}
```

The `<action-name>` will vary depending on the notification. Possible actions are:

- `subscribe-success`
- `subscribe-fail`
- `unsubscribe-pending`
- `unsubscribe-success`

The `offer-identifier` only appears in the notification if the offer is a *private offer*.

Subscribing an Amazon SQS queue to the Amazon SNS topic

We recommend subscribing an Amazon SQS queue to the provided SNS topics. For detailed instructions on creating an SQS queue and subscribing the queue to a topic, see [Subscribing an Amazon SQS queue to an Amazon SNS topic](#) in the *Amazon Simple Notification Service Developer Guide*.

Note

You can only subscribe to AWS Marketplace SNS topics from the AWS account used to sell the products. However, you can forward the messages to a different account. For more information, see [Sending Amazon SNS messages to an Amazon SQS queue in a different account](#) in the *Amazon Simple Notification Service Developer Guide*.

Polling the SQS queue for notifications

After you subscribe your SQS queue to an SNS topic, the messages are stored in SQS. You must define a service that continually polls the queue, looks for messages, and handles them accordingly.

Machine learning products

As a seller, you can use AWS Marketplace to create machine learning (ML) algorithms and models that your buyers can deploy in AWS. There are two types of Amazon SageMaker products listed in AWS Marketplace:

Model package

A pre-trained model for making predictions that does not require any further training by the buyer.

Algorithm

A model that requires the buyer to supply training data before it makes predictions. The training algorithm is included.

These products are available to buyers through the Amazon SageMaker console or AWS Marketplace. Buyers can review product descriptions, documentation, customer reviews, pricing, and support information. When they subscribe to either a model package product or algorithm product, it's added to their product list on the SageMaker console. Buyers can also use AWS SDKs, the AWS Command Line Interface (AWS CLI), or the SageMaker console to create a fully managed REST inference endpoint or perform inference on batches of data.

For support with creating machine learning products with Amazon SageMaker, contact the [AWS Marketplace Seller Operations](#) team.

Getting started with machine learning products

AWS Marketplace supports two machine learning product types, using Amazon SageMaker. Both types, the model package products and the algorithm products, produce a deployable inference model for making predictions.

SageMaker model package

An [Amazon SageMaker model package](#) product contains a pre-trained model. Pre-trained models can be deployed in SageMaker to make inferences or predictions in real time or in batches. This product contains a trained inference component with model artifacts, if any. As a seller, you can train a model using SageMaker or bring your own model.

SageMaker algorithm

Buyers can use a [SageMaker algorithm](#) product to perform complete machine learning workloads. An algorithm product has two logical components: training and inference. In SageMaker, buyers use their own datasets to create a training job with your training component. When the algorithm in your training component completes, it generates the model artifacts of the machine learning model. SageMaker saves the model artifacts in the buyers' Amazon Simple Storage Service (Amazon S3) bucket. In SageMaker, buyers can then deploy your inference component along with those generated model artifacts to perform inference (or prediction) in real time or in batches.

Deploying an inference model

Whether the inference model is created from a model package or an algorithm, there are two methods to deploy them:

- **Endpoint** – This method uses SageMaker to deploy the model and create an API endpoint. The buyer can use this endpoint as part of their backend service to power their applications. When data is sent to the endpoint, SageMaker passes it to the model container and returns the results in an API response. The endpoint and the container continue to run until stopped by the buyer.

Note

In AWS Marketplace, the endpoint method is referred to as *real-time inference*, and in the SageMaker documentation, it is referred to as *hosting services*. For more information, see [Deploy a Model in Amazon SageMaker](#).

- **Batch transform job** – In this method, a buyer stores datasets for inference in Amazon S3. When the batch transform job starts, SageMaker deploys the model, passes data from an S3 bucket to the model's container, and then returns the results to an S3 bucket. When the job completes, SageMaker stops the job. For more information, see [Get Inferences for an Entire Dataset with Batch Transform](#).

Note

Both methods are transparent to the model because SageMaker passes data to the model and returns results to the buyer.

Security and intellectual property

Amazon SageMaker protects both your intellectual property and buyer data for models and algorithms obtained from AWS Marketplace.

Protecting intellectual property

When you create a product, the code is packaged in Docker container images. For more information, see [Prepare your product in SageMaker](#), later in this guide. When you upload a container image, the image and artifacts are encrypted in transit and at rest. The images are also scanned for vulnerabilities before being published.

To help safeguard your intellectual property, SageMaker allows only buyers to access your product through AWS service endpoints. Buyers cannot directly access or pull container images or model artifacts, nor can they access the underlying infrastructure.

No network access

Unlike SageMaker models and algorithms that buyers create, when buyers launch your product from AWS Marketplace, the models and algorithms are deployed without network access. SageMaker deploys images in an environment with no access to the network or AWS service endpoints. For example, a container image can't make outbound API calls to services on the internet, [VPC endpoints](#), or any other AWS services.

Security of customer data

Your product runs in SageMaker within the buyer's AWS account. So, when a buyer uses your product to perform data inference, you as the seller can't access their data.

For algorithm products, model artifacts are outputted by your training image after each training job. Model artifacts are stored in the buyer's account. The model artifacts from the training job are used when the buyer deploys the model with your inference image. To protect any intellectual property that may be contained in the model artifact, encrypt them before outputting them.

Important

This security model prevents your code from accessing the internet during runtime. Therefore, your code can't use resources or libraries from the internet, so package your

dependencies in the Docker container image. This is especially important if you choose to encrypt your outputted artifacts from the training job. The keys to encrypt and decrypt artifacts can't be accessed over the internet at runtime. They must be packaged with your image.

For more information, see [Security in Amazon SageMaker](#).

Machine learning product pricing

You can choose from several available pricing models for your Amazon SageMaker products. Buyers who subscribe to your product run it in SageMaker within their own AWS account. The price for your buyers is a combination of the infrastructure costs for the resources running in their AWS account and the product pricing that you set.

Infrastructure pricing

Buyers are responsible for all the infrastructure costs of SageMaker while using your product. These costs are set by AWS and are available on the [Amazon SageMaker pricing](#) page.

Software pricing

You determine the software prices that AWS Marketplace charges the buyer for using your product. You set the pricing and terms when you are adding your machine learning product to AWS Marketplace.

All infrastructure and software prices per instance type are presented to the buyer on the product listing pages in AWS Marketplace before the buyer subscribes.

Free pricing

You can choose to offer your product for free. In this case, the buyer only pays for infrastructure costs.

Hourly pricing

You can offer your product with a price per hour per instance of your software running in SageMaker. You can charge a different hourly price for each instance type that your software runs

on. While a buyer runs your software, AWS Marketplace tracks usage and then bills the buyer accordingly. Usage is prorated to the minute.

For *model package* products, buyer can run your software in two different ways. They can host an endpoint continuously to perform real-time inference or run a batch transform job on a dataset. You can set different pricing for both of the ways a buyer can run your software.

For *algorithm* products, in addition to determining the prices for performing inference, as mentioned earlier, you also determine an hourly price for training jobs. You can charge a different hourly price for each instance type that your training image supports.

Annual contract with hourly pricing

In the contract option, you can specify a fixed upfront fee and the customer is invoiced for the full amount of the contract at the time of subscription. At the end of the annual contract, any instances that continue to run are billed at the hourly rate that you set.

Inference pricing

When the buyer runs your software by hosting an endpoint to continuously perform real-time inference, you can choose to set a price per inference.

Note

Batch transform processes always use hourly pricing. Training jobs for algorithm products also always use hourly pricing. You can set these prices independently of the inference pricing, and of each other.

By default, with inference pricing, AWS Marketplace charges your buyer for each invocation of your endpoint. However, in some cases, your software processes a batch of inferences in a single invocation (also known as a *mini-batch*). For an endpoint deployment, you can indicate a custom number of inferences that AWS Marketplace should charge the buyer for that single invocation. To do this, include a custom metering header in the HTTP response headers of your invocation, as in the following example. This example shows an invocation that charges the buyer for three inferences.

```
X-Amzn-Inference-Metering: {"Dimension": "inference.count", "ConsumedUnits": 3}
```

Note

For inference pricing, AWS Marketplace only charges buyer for requests where the HTTP response code is 2XX.

Free trial

Optionally, you can create a free trial for your product and define the number of days of the free trial. Free trials can be 5–120 days. During the free trial, buyers can run your software as much as they want and aren't charged for your software. Buyers are charged for the infrastructure costs during the free trial. After the trial ends, they are charged your normal software price, along with the infrastructure costs.

Note

You can only create a free trial for offers that are charged hourly. You can't create a free trial for a product with inference pricing.

When buyers subscribe to a product with a free trial, they receive a welcome email message. The message includes the term of the free trial, a calculated expiration date, and details on unsubscribing. A reminder email message is sent three days before the expiration date.

If you offer a free trial for your product in AWS Marketplace, you agree to the specific [refund policy](#) for free trials.

Note

For information on Private offers for machine learning, see [Private offers](#).

Price change

As a seller, you can change the pricing of your machine learning products by contacting the [AWS Marketplace Seller Operations](#) team. Provide the product ID and new pricing details. New prices are effective after 90 days. Additionally, you must wait 90 days before making a second price change. This limitation also applies to adding new instance types to the existing list of supported instances.

For example,, if you increase the price of your machine learning product on November 1st, 2023, you can add new instance types or make a second price change after January 30th, 2024.

Prepare your product in SageMaker

Before publishing your product in AWS Marketplace, you must prepare it in Amazon SageMaker. There are three steps to preparing your product:

1. [Packaging your code into images](#) – To prepare a model package or algorithm product, you must create the Docker container images for your product.
2. [Uploading your images](#) – After packaging your code in container images and testing them locally, upload the images and scan them for known vulnerabilities. Fix any vulnerabilities before continuing.
3. [Creating your Amazon SageMaker resource](#) – After your images are scanned successfully, they can be used to create a model package or algorithm resource in SageMaker.

Packaging your code into images

Machine learning products in AWS Marketplace use Amazon SageMaker to create and run the machine learning logic you provide for buyers. SageMaker runs Docker container images that contain your logic. SageMaker runs these containers in a secure and scalable infrastructure. For more information, see [Security and intellectual property](#).

Topics

- [Which type of container image do I create?](#)
- [Model package images](#)
- [Algorithm images](#)

Which type of container image do I create?

The two types of container images are an inference image and a training image.

To create a model package product, you need only an inference image. For detailed instructions, see [Model package images](#).

To create an algorithm product, you need both training and inference images. For detailed instructions, see [Algorithm images](#).

To package code properly into a container image, the container must adhere to the SageMaker file structure. The container must expose the correct endpoints to ensure that the service can pass data to and from your container. The following sections explain the details of this process.

Important

For security purposes, when a buyer subscribes to your containerized product, the Docker containers run in an isolated environment without an internet connection. When you create your containers, don't rely on outgoing calls over the internet because they will fail. Calls to AWS services will also fail. For more information, see the [Security and intellectual property](#) section.

Optionally, when creating your inference and training images, use a container from [Available Deep Learning Containers Images](#) as a starting point. The images are already properly packaged with different machine learning frameworks.

Model package images

An Amazon SageMaker model package is a pre-trained model that makes predictions and does not require any further training by the buyer.

A model package includes the following components:

- An inference image stored in [Amazon Elastic Container Registry](#) (Amazon ECR)
- (Optional) Model artifacts, stored separately in [Amazon S3](#)

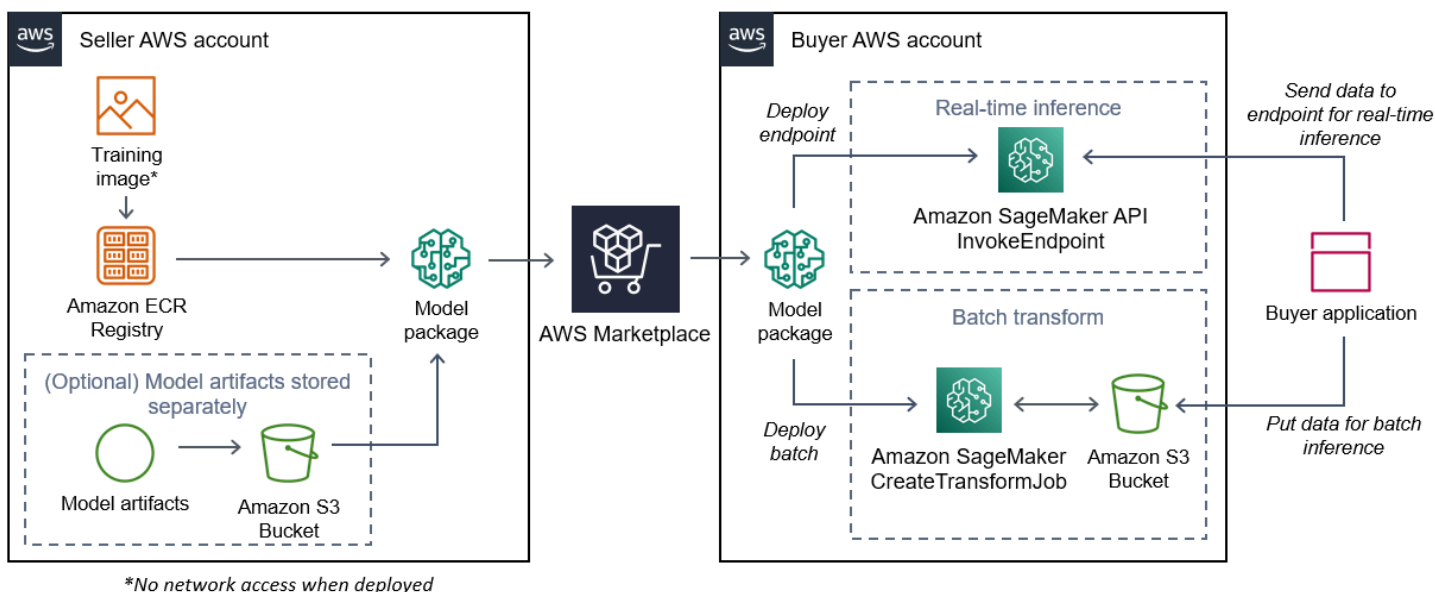
Note

Model artifacts are files your model uses to make predictions and are generally the result of your own training processes. Artifacts can be any file type that is needed by your model but must use tar.gz compression. For model packages, they can either be bundled within your inference image or stored separately in Amazon SageMaker. The model artifacts stored in Amazon S3 are loaded into the inference container at runtime. When publishing your model package, those artifacts are published and stored in AWS Marketplace owned Amazon S3 buckets that are inaccessible by the buyer directly.

Tip

If your inference model is built with a deep learning framework such as Gluon, Keras, MXNet, PyTorch, TensorFlow, TensorFlow-Lite, or ONNX, consider using Amazon SageMaker Neo. Neo can automatically optimize inference models that deploy to a specific family of cloud instance types such as `m1.c4`, `m1.p2`, and others. For more information, see [Optimize model performance using Neo](#) in the *Amazon SageMaker Developer Guide*.

The following diagram shows the workflow for publishing and using model package products.



1. The seller creates an inference image (no network access when deployed) and pushes it to the Amazon ECR Registry.

The model artifacts can either be bundled in the inference image or stored separately in S3.

2. The seller then creates a model package resource in Amazon SageMaker and publishes their ML product on AWS Marketplace.
3. The buyer subscribes to the ML product and deploys the model.

Note

The model can be deployed as an endpoint for real-time inferences or as a batch job to get predictions for an entire dataset at once. For more information, see [Deploy Models for Inference](#).

4. SageMaker runs the inference image. Any seller-provided model artifacts not bundled in the inference image are loaded dynamically at runtime.
5. SageMaker passes the buyer's inference data to the container by using the container's HTTP endpoints and returns the prediction results.

Creating an inference image for model packages

This section provides a walkthrough for packaging your inference code into an inference image for your model package product. The process consists of the following steps:

Steps

- [Step 1: Creating the container image](#)
- [Step 2: Building and testing the image locally](#)

The inference image is a Docker image containing your inference logic. The container at runtime exposes HTTP endpoints to allow SageMaker to pass data to and from your container.

Note

The following is only one example of packaging code for an inference image. For more information, see [Using Docker containers with SageMaker](#) and the [AWS Marketplace SageMaker examples](#) on GitHub.

The following example uses a web service, [Flask](#), for simplicity, and is not considered production-ready.

Step 1: Creating the container image

For the inference image to be compatible with SageMaker, the Docker image must expose HTTP endpoints. While your container is running, SageMaker passes buyer inputs for inference to the container's HTTP endpoint. The inference results are returned in the body of the HTTP response.

The following walkthrough uses the Docker CLI in a development environment using a Linux Ubuntu distribution.

- [Create the web server script](#)
- [Create the script for the container run](#)
- [Create the Dockerfile](#)
- [Package or upload the model artifacts](#)

Create the web server script

This example uses a Python server called [Flask](#), but you can use any web server that works for your framework.

Note

[Flask](#) is used here for simplicity. It is not considered a production-ready web server.

Create a Flask web server script that serves the two HTTP endpoints on TCP port 8080 that SageMaker uses. The following are the two expected endpoints:

- `/ping` – SageMaker makes HTTP GET requests to this endpoint to check if your container is ready. When your container is ready, it responds to HTTP GET requests at this endpoint with an HTTP 200 response code.
- `/invocations` – SageMaker makes HTTP POST requests to this endpoint for inference. The input data for inference is sent in the body of the request. The user-specified content type is passed in the HTTP header. The body of the response is the inference output. For details about timeouts, see [Requirements and best practices for creating machine learning products](#).

`./web_app_serve.py`

```
# Import modules
import json
import re
from flask import Flask
from flask import request
app = Flask(__name__)
```

```
# Create a path for health checks
@app.route("/ping")
def endpoint_ping():
    return ""

# Create a path for inference
@app.route("/invocations", methods=["POST"])
def endpoint_invocations():

    # Read the input
    input_str = request.get_data().decode("utf8")

    # Add your inference code between these comments.
    #
    #
    #
    #
    # Add your inference code above this comment.

    # Return a response with a prediction
    response = {"prediction":"a", "text":input_str}
    return json.dumps(response)
```

In the previous example, there is no actual inference logic. For your actual inference image, add the inference logic into the web app so it processes the input and returns the actual prediction.

Your inference image must contain all of its required dependencies because it will not have internet access, nor will it be able to make calls to any AWS services.

Note

This same code is called for both real-time and batch inferences

Create the script for the container run

Create a script named `serve` that SageMaker runs when it runs the Docker container image. The following script starts the HTTP web server.

```
./serve
```

```
#!/bin/bash

# Run flask server on port 8080 for SageMaker
flask run --host 0.0.0.0 --port 8080
```

Create the Dockerfile

Create a Dockerfile in your build context. This example uses Ubuntu 18.04, but you can start from any base image that works for your framework.

./Dockerfile

```
FROM ubuntu:18.04

# Specify encoding
ENV LC_ALL=C.UTF-8
ENV LANG=C.UTF-8

# Install python-pip
RUN apt-get update \
&& apt-get install -y python3.6 python3-pip \
&& ln -s /usr/bin/python3.6 /usr/bin/python \
&& ln -s /usr/bin/pip3 /usr/bin/pip;

# Install flask server
RUN pip install -U Flask;

# Add a web server script to the image
# Set an environment to tell flask the script to run
COPY /web_app_serve.py /web_app_serve.py
ENV FLASK_APP=/web_app_serve.py

# Add a script that Amazon SageMaker will run
# Set run permissions
# Prepend program directory to $PATH
COPY /serve /opt/program/serve
RUN chmod 755 /opt/program/serve
ENV PATH=/opt/program:${PATH}
```

The Dockerfile adds the two previously created scripts to the image. The directory of the serve script is added to the PATH so it can run when the container runs.

Package or upload the model artifacts

The two ways to provide the model artifacts from training the model to the inference image are as follows:

- Packaged statically with the inference image.
- Loaded dynamically at runtime. Because it's loaded dynamically, you can use the same image for packaging different machine learning models.

If you want to package your model artifacts with the inference image, include the artifacts in the `Dockerfile`.

If you want to load your model artifacts dynamically, store those artifacts separately in a compressed file (`.tar.gz`) in Amazon S3. When creating the model package, specify the location of the compressed file, and SageMaker extracts and copies the contents to the container directory `/opt/ml/model/` when running your container. When publishing your model package, those artifacts are published and stored in AWS Marketplace owned Amazon S3 buckets that are inaccessible by the buyer directly.

Step 2: Building and testing the image locally

In the build context, the following files now exist:

- `./Dockerfile`
- `./web_app_serve.py`
- `./serve`
- Your inference logic and (optional) dependencies

Next build, run, and test the container image.

Build the image

Run the Docker command in the build context to build and tag the image. This example uses the tag `my-inference-image`.

```
sudo docker build --tag my-inference-image ./
```

After running this Docker command to build the image, you should see output as Docker builds the image based on each line in your `Dockerfile`. When it finishes, you should see something similar to the following.

```
Successfully built abcdef123456
Successfully tagged my-inference-image:latest
```

Run locally

After your build has completed, you can test the image locally.

```
sudo docker run \
  --rm \
  --publish 8080:8080/tcp \
  --detach \
  --name my-inference-container \
  my-inference-image \
  serve
```

The following are details about the command:

- `--rm` – Automatically remove the container after it stops.
- `--publish 8080:8080/tcp` – Expose port 8080 to simulate the port that SageMaker sends HTTP requests to.
- `--detach` – Run the container in the background.
- `--name my-inference-container` – Give this running container a name.
- `my-inference-image` – Run the built image.
- `serve` – Run the same script that SageMaker runs when running the container.

After running this command, Docker creates a container from the inference image you built and runs it in the background. The container runs the `serve` script, which launches your web server for testing purposes.

Test the ping HTTP endpoint

When SageMaker runs your container, it periodically pings the endpoint. When the endpoint returns an HTTP response with status code 200, it signals to SageMaker that the container is ready

for inference. You can test this by running the following command, which tests the endpoint and includes the response header.

```
curl --include http://127.0.0.1:8080/ping
```

Example output is as follows.

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 0
Server: MyServer/0.16.0 Python/3.6.8
Date: Mon, 21 Oct 2019 06:58:54 GMT
```

Test the inference HTTP endpoint

When the container indicates it is ready by returning a 200 status code to your ping, SageMaker passes the inference data to the `/invocations` HTTP endpoint via a POST request. Test the inference point by running the following command.

```
curl \
  --request POST \
  --data "hello world" \
  http://127.0.0.1:8080/invocations
```

Example output is as follows.

```
{"prediction": "a", "text": "hello world"}
```

With these two HTTP endpoints working, the inference image is now compatible with SageMaker.

Note

The model of your model package product can be deployed in two ways: real time and batch. In both deployments, SageMaker uses the same HTTP endpoints while running the Docker container.

To stop the container, run the following command.

```
sudo docker container stop my-inference-container
```


When your inference image is ready and tested, you can continue to [Uploading your images](#).

Algorithm images

An Amazon SageMaker algorithm requires that the buyer bring their own data to train before it makes predictions.

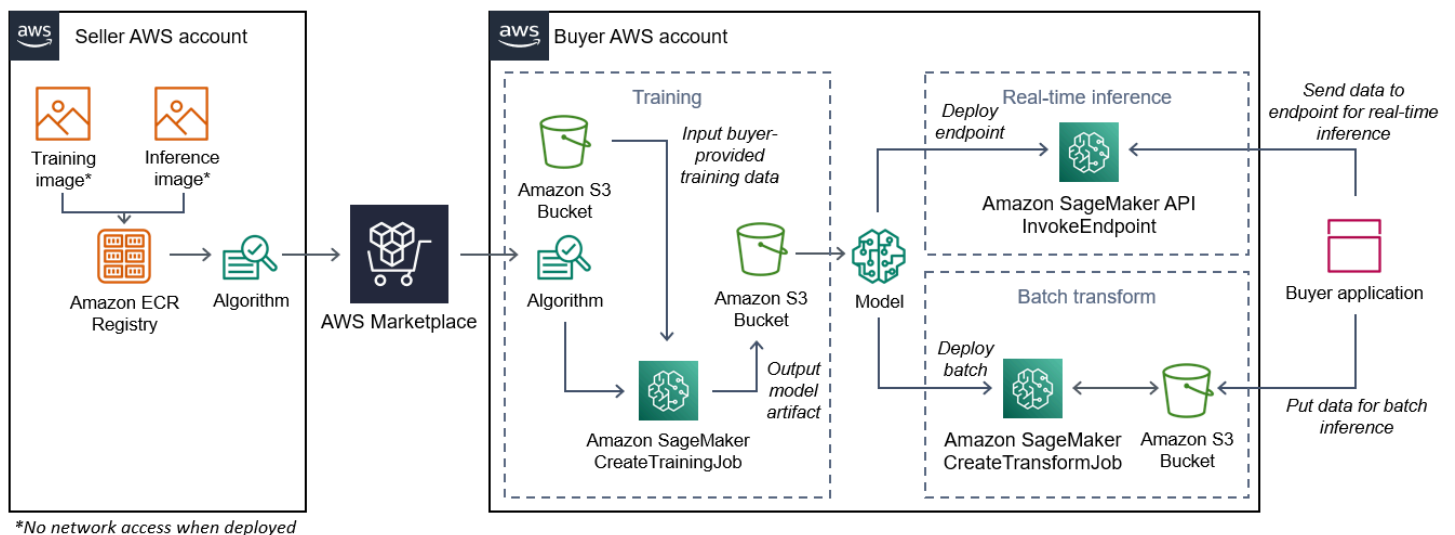
An algorithm includes the following components:

- A training image stored in [Amazon ECR](#)
- An inference image stored in Amazon Elastic Container Registry (Amazon ECR)

Note

For algorithm products, the training container generates model artifacts that are loaded into the inference container on model deployment.

The following diagram shows the workflow for publishing and using algorithm products.



1. The seller creates a training image and an inference image (no network access when deployed) and uploads it to the Amazon ECR Registry.
2. The seller then creates an algorithm resource in Amazon SageMaker and publishes their ML product on AWS Marketplace.
3. The buyer subscribes to the ML product.

4. The buyer creates a training job with a compatible dataset and appropriate hyperparameter values. SageMaker runs the training image and loads the training data and hyperparameters into the training container. When the training job completes, the model artifacts located in `/opt/ml/model/` are compressed and copied to the buyer's [Amazon S3](#) bucket.
5. The buyer creates a model package with the model artifacts from the training stored in Amazon S3 and deploys the model.
6. SageMaker runs the inference image, extracts the compressed model artifacts, and loads the files into the inference container directory path `/opt/ml/model/` where it is consumed by the code that serves the inference.
7. Whether the model deploys as an endpoint or a batch transform job, SageMaker passes the data for inference on behalf of the buyer to the container via the container's HTTP endpoint and returns the prediction results.

Note

For more information, see [Train Models](#).

Creating a training image for algorithms

This section provides a walkthrough for packaging your training code into a training image. A training image is required to create an algorithm product.

A *training image* is a Docker image containing your training algorithm. The container adheres to a specific file structure to allow SageMaker to copy data to and from your container.

Both the training and inference images are required when publishing an algorithm product. After creating your training image, you must create an inference image. The two images can be combined into one image or remain as separate images. Whether to combine the images or separate them is up to you. Typically, inference is simpler than training, and you might want separate images to help with inference performance.

Note

The following is only one example of packaging code for a training image. For more information, see [Use your own algorithms and models with the AWS Marketplace](#) and the [AWS Marketplace SageMaker examples](#) on GitHub.

Steps

- [Step 1: Creating the container image](#)
- [Step 2: Building and testing the image locally](#)

Step 1: Creating the container image

For the training image to be compatible with Amazon SageMaker, it must adhere to a specific file structure to allow SageMaker to copy the training data and configuration inputs to specific paths in your container. When the training completes, the generated model artifacts are stored in a specific directory path in the container where SageMaker copies from.

The following uses Docker CLI installed in a development environment on an Ubuntu distribution of Linux.

- [Prepare your program to read configuration inputs](#)
- [Prepare your program to read data inputs](#)
- [Prepare your program to write training outputs](#)
- [Create the script for the container run](#)
- [Create the Dockerfile](#)

Prepare your program to read configuration inputs

If your training program requires any buyer-provided configuration input, the following is where those are copied to inside your container when ran. If required, your program must read from those specific file paths.

- `/opt/ml/input/config` is the directory that contains information which controls how your program runs.
 - `hyperparameters.json` is a JSON-formatted dictionary of hyperparameter names and values. The values are strings, so you may need to convert them.
 - `resourceConfig.json` is a JSON-formatted file that describes the network layout used for [distributed training](#). If your training image does not support distributed training, you can ignore this file.

Note

For more information about configuration inputs, see [How Amazon SageMaker Provides Training Information](#).

Prepare your program to read data inputs

Training data can be passed to the container in one of the following two modes. Your training program that runs in the container digests the training data in one of those two modes.

File mode

- `/opt/ml/input/data/<channel_name>/` contains the input data for that channel. The channels are created based on the call to the `CreateTrainingJob` operation, but it's generally important that channels match what the algorithm expects. The files for each channel are copied from [Amazon S3](#) to this directory, preserving the tree structure indicated by the Amazon S3 key structure.

Pipe mode

- `/opt/ml/input/data/<channel_name>_<epoch_number>` is the pipe for a given epoch. Epochs start at zero and increase by one each time you read them. There is no limit to the number of epochs that you can run, but you must close each pipe before reading the next epoch.

Prepare your program to write training outputs

The output of the training is written to the following container directories:

- `/opt/ml/model/` is the directory where you write the model or the model artifacts that your training algorithm generates. Your model can be in any format that you want. It can be a single file or a whole directory tree. SageMaker packages any files in this directory into a compressed file (`.tar.gz`). This file is available at the Amazon S3 location returned by the `DescribeTrainingJob` API operation.
- `/opt/ml/output/` is a directory where the algorithm can write a `failure` file that describes why the job failed. The contents of this file are returned in the `FailureReason` field of the `DescribeTrainingJob` result. For jobs that succeed, there is no reason to write this file because it's ignored.

Create the script for the container run

Create a `train` shell script that SageMaker runs when it runs the Docker container image. When the training completes and the model artifacts are written to their respective directories, exit the script.

`./train`

```
#!/bin/bash

# Run your training program here
#
#
#
```

Create the Dockerfile

Create a Dockerfile in your build context. This example uses Ubuntu 18.04 as the base image, but you can start from any base image that works for your framework.

`./Dockerfile`

```
FROM ubuntu:18.04

# Add training dependencies and programs
#
#
#
#
# Add a script that SageMaker will run
# Set run permissions
# Prepend program directory to $PATH
COPY /train /opt/program/train
RUN chmod 755 /opt/program/train
ENV PATH=/opt/program:${PATH}
```

The Dockerfile adds the previously created `train` script to the image. The script's directory is added to the `PATH` so it can run when the container runs.

In the previous example, there is no actual training logic. For your actual training image, add the training dependencies to the `Dockerfile`, and add the logic to read the training inputs to train and generate the model artifacts.

Your training image must contain all of its required dependencies because it will not have internet access.

For more information, see [Use your own algorithms and models with the AWS Marketplace](#) and the [AWS Marketplace SageMaker examples](#) on GitHub.

Step 2: Building and testing the image locally

In the build context, the following files now exist:

- `./Dockerfile`
- `./train`
- Your training dependencies and logic

Next you can build, run, and test this container image.

Build the image

Run the Docker command in the build context to build and tag the image. This example uses the tag `my-training-image`.

```
sudo docker build --tag my-training-image ./
```

After running this Docker command to build the image, you should see output as Docker builds the image based on each line in your `Dockerfile`. When it finishes, you should see something similar to the following.

```
Successfully built abcdef123456  
Successfully tagged my-training-image:latest
```

Run locally

After that has completed, test the image locally as shown in the following example.

```
sudo docker run \  
  --rm \  
  my-training-image
```

```
--volume '<path_to_input>:/opt/ml/input:ro' \  
--volume '<path_to_model>:/opt/ml/model' \  
--volume '<path_to_output>:/opt/ml/output' \  
--name my-training-container \  
my-training-image \  
train
```

The following are command details:

- `--rm` – Automatically remove the container after it stops.
- `--volume '<path_to_input>:/opt/ml/input:ro'` – Make test input directory available to container as read-only.
- `--volume '<path_to_model>:/opt/ml/model'` – Bind mount the path where the model artifacts are stored on the host machine when the training test is complete.
- `--volume '<path_to_output>:/opt/ml/output'` – Bind mount the path where the failure reason in a `failure` file is written to on the host machine.
- `--name my-training-container` – Give this running container a name.
- `my-training-image` – Run the built image.
- `train` – Run the same script SageMaker runs when running the container.

After running this command, Docker creates a container from the training image you built and runs it. The container runs the `train` script, which starts your training program.

After your training program finishes and the container exits, check that the output model artifacts are correct. Additionally, check the log outputs to confirm that they are not producing logs that you do not want, while ensuring enough information is provided about the training job.

This completes packaging your training code for an algorithm product. Because an algorithm product also includes an inference image, continue to the next section, [Creating an inference image for algorithms](#).

Creating an inference image for algorithms

This section provides a walkthrough for packaging your inference code into an inference image for your algorithm product.

The inference image is a Docker image containing your inference logic. The container at runtime exposes HTTP endpoints to allow SageMaker to pass data to and from your container.

Both the training and inference images are required when publishing an algorithm product. If you have not already done so, see the previous section about [Creating a training image for algorithms](#). The two images can be combined into one image or remain as separate images. Whether to combine the images or separate them is up to you. Typically, inference is simpler than training, and you might want separate images to help with inference performance.

Note

The following is only one example of packaging code for an inference image. For more information, see [Use your own algorithms and models with the AWS Marketplace](#) and the [AWS Marketplace SageMaker examples](#) on GitHub.

The following example uses a web service, [Flask](#), for simplicity, and is not considered production-ready.

Steps

- [Step 1: Creating the inference image](#)
- [Step 2: Building and testing the image locally](#)

Step 1: Creating the inference image

For the inference image to be compatible with SageMaker, the Docker image must expose HTTP endpoints. While your container is running, SageMaker passes inputs for inference provided by the buyer to your container's HTTP endpoint. The result of the inference is returned in the body of the HTTP response.

The following uses Docker CLI installed in a development environment on an Ubuntu distribution of Linux.

- [Create the web server script](#)
- [Create the script for the container run](#)
- [Create the Dockerfile](#)
- [Preparing your program to dynamically load model artifacts](#)

Create the web server script

This example uses a Python server called [Flask](#), but you can use any web server that works for your framework.

Note

[Flask](#) is used here for simplicity. It is not considered a production-ready web server.

Create the Flask web server script that serves the two HTTP endpoints on TCP port 8080 that SageMaker uses. The following are the two expected endpoints:

- `/ping` – SageMaker makes HTTP GET requests to this endpoint to check if your container is ready. When your container is ready, it responds to HTTP GET requests at this endpoint with an HTTP 200 response code.
- `/invocations` – SageMaker makes HTTP POST requests to this endpoint for inference. The input data for inference is sent in the body of the request. The user-specified content type is passed in the HTTP header. The body of the response is the inference output.

`./web_app_serve.py`

```
# Import modules
import json
import re
from flask import Flask
from flask import request
app = Flask(__name__)

# Create a path for health checks
@app.route("/ping")
def endpoint_ping():
    return ""

# Create a path for inference
@app.route("/invocations", methods=["POST"])
def endpoint_invocations():

    # Read the input
    input_str = request.get_data().decode("utf8")
```

```
# Add your inference code here.
#
#
#
#
# Add your inference code here.

# Return a response with a prediction
response = {"prediction":"a","text":input_str}
return json.dumps(response)
```

In the previous example, there is no actual inference logic. For your actual inference image, add the inference logic into the web app so it processes the input and returns the prediction.

Your inference image must contain all of its required dependencies because it will not have internet access.

Create the script for the container run

Create a script named `serve` that SageMaker runs when it runs the Docker container image. In this script, start the HTTP web server.

./serve

```
#!/bin/bash

# Run flask server on port 8080 for SageMaker
flask run --host 0.0.0.0 --port 8080
```

Create the Dockerfile

Create a Dockerfile in your build context. This example uses Ubuntu 18.04, but you can start from any base image that works for your framework.

./Dockerfile

```
FROM ubuntu:18.04

# Specify encoding
ENV LC_ALL=C.UTF-8
ENV LANG=C.UTF-8
```

```
# Install python-pip
RUN apt-get update \
&& apt-get install -y python3.6 python3-pip \
&& ln -s /usr/bin/python3.6 /usr/bin/python \
&& ln -s /usr/bin/pip3 /usr/bin/pip;

# Install flask server
RUN pip install -U Flask;

# Add a web server script to the image
# Set an environment to tell flask the script to run
COPY /web_app_serve.py /web_app_serve.py
ENV FLASK_APP=/web_app_serve.py

# Add a script that Amazon SageMaker will run
# Set run permissions
# Prepend program directory to $PATH
COPY /serve /opt/program/serve
RUN chmod 755 /opt/program/serve
ENV PATH=/opt/program:${PATH}
```

The Dockerfile adds the two created previously scripts to the image. The directory of the serve script is added to the PATH so it can run when the container runs.

Preparing your program to dynamically load model artifacts

For algorithm products, the buyer uses their own datasets with your training image to generate unique model artifacts. When the training process completes, your training container outputs model artifacts to the container directory `/opt/ml/model/`. SageMaker compresses the contents in that directory into a `.tar.gz` file and stores it in the buyer's AWS account in Amazon S3.

When the model deploys, SageMaker runs your inference image, extracts the model artifacts from the `.tar.gz` file stored in the buyer's account in Amazon S3, and loads them into the inference container in the `/opt/ml/model/` directory. At runtime, your inference container code uses the model data.

Note

To protect any intellectual property that might be contained in the model artifact files, you can choose to encrypt the files before outputting them. For more information, see [Security and intellectual property](#).

Step 2: Building and testing the image locally

In the build context, the following files now exist:

- ./Dockerfile
- ./web_app_serve.py
- ./serve

Next you can build, run, and test this container image.

Build the image

Run the Docker command to build and tag the image. This example uses the tag `my-inference-image`.

```
sudo docker build --tag my-inference-image ./
```

After running this Docker command to build the image, you should see output as Docker builds the image based on each line in your `Dockerfile`. When it finishes, you should see something similar to the following.

```
Successfully built abcdef123456  
Successfully tagged my-inference-image:latest
```

Run locally

After your build has completed, you can test the image locally.

```
sudo docker run \  
  --rm \  
  --publish 8080:8080/tcp \  
  --volume '<path_to_model>:/opt/ml/model:ro' \  
  --detach \  
  --name my-inference-container \  
  my-inference-image \  
  serve
```

The following are command details:

- `--rm` – Automatically remove the container after it stops.

- `--publish 8080:8080/tcp` – Expose port 8080 to simulate the port SageMaker sends HTTP requests to.
- `--volume '<path_to_model>:/opt/ml/model:ro'` – Bind mount the path to where the test model artifacts are stored on the host machine as read-only to make them available to your inference code in the container.
- `--detach` – Run the container in the background.
- `--name my-inference-container` – Give this running container a name.
- `my-inference-image` – Run the built image.
- `serve` – Run the same script SageMaker runs when running the container.

After running this command, Docker creates a container from the inference image and runs it in the background. The container runs the `serve` script, which starts your web server for testing purposes.

Test the ping HTTP endpoint

When SageMaker runs your container, it periodically pings the endpoint. When the endpoint returns an HTTP response with status code 200, it signals to SageMaker that the container is ready for inference.

Run the following command to test the endpoint and include the response header.

```
curl --include http://127.0.0.1:8080/ping
```

Example output is shown in the following example.

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 0
Server: MyServer/0.16.0 Python/3.6.8
Date: Mon, 21 Oct 2019 06:58:54 GMT
```

Test the inference HTTP endpoint

When the container indicates it is ready by returning a 200 status code, SageMaker passes the inference data to the `/invocations` HTTP endpoint via a POST request.

Run the following command to test the inference endpoint.

```
curl \  
  --request POST \  
  --data "hello world" \  
  http://127.0.0.1:8080/invocations
```

Example output is shown in the following example..

```
{"prediction": "a", "text": "hello world"}
```

With these two HTTP endpoints working, the inference image is now compatible with SageMaker.

Note

The model of your algorithm product can be deployed in two ways: real time and batch. For both deployments, SageMaker uses the same HTTP endpoints while running the Docker container.

To stop the container, run the following command.

```
sudo docker container stop my-inference-container
```

After both your training and inference images for your algorithm product are ready and tested, continue to [Uploading your images](#).

Uploading your images

This section provides a walkthrough for uploading your inference and training images to Amazon Elastic Container Registry. [Amazon ECR](#) is a fully managed Docker registry. This is where Amazon SageMaker pulls images from to create a model package for inference or algorithm for training jobs. This is also where AWS Marketplace retrieves the images to publish your model package and algorithm products.

Which images must I upload?

If you're publishing a model package, upload only an inference image. If you're publishing an algorithm, upload both an inference image and a training image. If the inference and training images are combined, upload the combined image only once.

What IAM permissions are required?

The following steps assume that the local machine has the correct AWS credentials for an AWS Identity and Access Management (IAM) role or user in the seller AWS account. The role or user must have the correct policies in place for both AWS Marketplace and Amazon ECR. For example, you could use the following AWS managed policies:

- `AWSMarketplaceSellerProductsFullAccess` – For access to AWS Marketplace
- `AmazonEC2ContainerRegistryFullAccess` – For access to Amazon ECR

Log your Docker client into AWS

Set a variable for the AWS Region that you want to publish from (see [Supported AWS Regions for publishing](#)). For this example, use the US East (Ohio) Region.

```
region=us-east-2
```

Run the following command to set a variable with your AWS account ID. This example assumes that the current AWS Command Line Interface (AWS CLI) credentials belong to the seller's AWS account.

```
account=$(aws sts get-caller-identity --query Account --output text)
```

To authenticate your Docker CLI client with your AWS account Amazon ECR Docker registry for your Region, run the following command.

```
aws ecr get-login-password \  
--region ${region} \  
| sudo docker login \  
--username AWS \  
--password-stdin \  
${account}.dkr.ecr.${region}.amazonaws.com
```

Create repository and upload image

Set a variable for the tag of the uploaded image and another variable for the name of the uploaded image repository.

```
image=my-inference-image
```

```
repo=my-inference-image
```

Note

In previous sections of this guide where the inference and training images were built, they were tagged as **my-inference-image** and **my-training-image**, respectively. For this example, create and upload the inference image to a repository with the same name.

Run the following command to create the image repository in Amazon ECR.

```
aws ecr --region ${region} create-repository --repository-name "${repo}"
```

The full name of the Amazon ECR repository location is made up of the following parts:
<account-id>.dkr.ecr.<region>.amazonaws.com/<image-repository-name>

To push the image to the repository, you must tag it with the full name of the repository location.

Set a variable for the full name of the image repository location along with the latest tag.

```
fullname="${account}.dkr.ecr.${region}.amazonaws.com/${repo}:latest"
```

Tag the image with the full name.

```
sudo docker tag ${image} ${fullname}
```

Finally, push the inference image to the repository in Amazon ECR.

```
sudo docker push ${fullname}
```

After the upload completes, the image appears in the [repository list of the Amazon ECR console](#) in the Region that you are publishing from. In the previous example, the image was pushed to a repository in the US East (Ohio) Region.

Scan your uploaded image

In the [Amazon ECR console](#), choose the AWS Region that you are publishing from, and open the repository that the image was uploaded to. Select your uploaded image and start a scan to check

for known vulnerabilities. AWS Marketplace checks the Amazon ECR scan results of the container images used in your Amazon SageMaker resource before publishing it. Before you can create your product, you must fix container images that have vulnerabilities with either a Critical or High severity.

After your images are scanned successfully, they can be used to create a model package or algorithm resource.

If you believe that your product had errors in the scan that are false positives, contact the [AWS Marketplace Seller Operations](#) team with information about the error.

Next steps

- See size limits in [Requirements and best practices for creating machine learning products](#)
- Continue to [Creating your Amazon SageMaker resource](#)

Creating your Amazon SageMaker resource

To publish a model package or algorithm product, you must create the respective [model package resource](#) or [algorithm resource](#) in Amazon SageMaker.

When you create your resource for an AWS Marketplace product, it must be certified through a validation step. The validation step requires that you provide data to test your model package or algorithm resource before it can be published.

Note

If you haven't yet created the images for your product and uploaded them to Amazon Elastic Container Registry (Amazon ECR), see [Packaging your code into images](#) and [Uploading your images](#) for information about how to do so.

Creating your model package

The following are requirements for creating a model package for AWS Marketplace:

- An inference image stored in [Amazon ECR](#)
- (Optional) Model artifacts, stored separately in [Amazon S3](#)
- Your test data used for inferences, stored in Amazon Simple Storage Service (Amazon S3)

Note

The following is about creating a model package product. For more information about model packages in SageMaker, see [Create a Model Package Resource](#).

Creating the model package resources

The following procedures step you through creating the model package resources.

Step 1: To create the model package resources

1. Open the [Amazon SageMaker console](#).
2. Ensure that you are in the AWS Region that you want to publish from by looking at the top right of the page. For publishing, see the [Supported AWS Regions for publishing](#) section. The inference image you uploaded to Amazon ECR in previous steps must be in the same Region.
3. In the left navigation menu, choose **Model packages**.
4. Choose **Create model package**.

After you create the package, you need to set the specifications of the inference package.

Step 2: To set inference specifications

1. Provide a **Name** for your model package (for example, *my-model-package*).
2. For **Location of inference image**, enter the URI of your inference image that was uploaded to Amazon ECR. You can retrieve the URI by locating your image in the [Amazon ECR console](#).
3. If your model artifacts from training are bundled with your logic in your inference image, leave the **Location of model data artifacts** empty. Otherwise, specify the full Amazon S3 location of the compressed file (.tar.gz) of your model artifacts.
4. Using the dropdown box, choose the supported instance types of your inference image for both real-time inference (also known as *endpoint*) and batch-transform jobs.
5. Choose **Next**.

Before your model package can be created and published, validation is necessary to ensure that it functions as expected. This requires that you run a batch transform job with test data for inference that you provide. The validation specifications tell SageMaker how to perform the validation.

Step 3: To set validation specifications

1. Set **Publish this model package in AWS Marketplace** to **Yes**. If you set this to **No**, you can't publish this model package later. Choosing **Yes** [certifies](#) your model package for AWS Marketplace and requires the validation step.
2. If this is the first time completing this process, choose **Create a new role** for the **IAM role**. Amazon SageMaker uses this role when it deploys your model package. This includes actions, such as pulling images from Amazon ECR and artifacts from Amazon S3. Review the settings, and choose **Create role**. Creating a role here grants permissions described by the [AmazonSageMakerFullAccess](#) IAM policy to the role that you create.
3. Edit the **JSON** in the validation profile. For details about allowed values, see [TransformJobDefinition](#).
 1. `TransformInput.DataSource.S3Uri`: Set to where your test data for inference is stored.
 2. `TransformInput.ContentType`: Specify your test data content type (for example, `application/json`, `text/plain`, `image/png` , or any other value). SageMaker does not validate the actual input data. This value is passed to your container HTTP endpoint in the `Content-type` header value.
 3. `TransformInput.CompressionType`: Set to `None` if your test data for inference in Amazon S3 is not compressed.
 4. `TransformInput.SplitType`: Set to `None` to pass each object in Amazon S3 as a whole for inference.
 5. `TransformOutput.S3OutputPath`: Set to the location that the inference output is stored.
 6. `TransformOutput.AssembleWith`: Set to `None` to output each inference as separate objects in Amazon S3.
4. Choose **Create model package**.

SageMaker pulls the inference image from Amazon ECR, copies any artifacts to the inference container, and runs a batch transform job using your test data for inference. After the validation succeeds, the status changes to **Completed**.

Note

The validation step does not evaluate the accuracy of the model with your test data. The validation step checks if the container runs and responds as expected.

You have completed creating your model product resources. Continue to [Publishing your product in AWS Marketplace](#).

Creating your algorithm

The following are requirements for creating an algorithm for AWS Marketplace:

- An inference image, stored in Amazon ECR
- A training image, stored in Amazon ECR
- Your test data for training, stored in Amazon S3
- Your test data for inference, stored in Amazon S3

Note

The following walkthrough creates an algorithm product. For more information, see [Create an Algorithm Resource](#).

Creating the algorithm resources

The following procedures step you through creating the resources in your algorithm package.

Step 1: To create the algorithm resources

1. Open the [Amazon SageMaker console](#).
2. Ensure that you are in the AWS Region that you want to publish from by looking at the top right of the page (see [Supported AWS Regions for publishing](#)). The training and inference images you uploaded to Amazon ECR in previous steps must be in this same Region.
3. In the left navigation menu, choose **Algorithms**.
4. Choose **Create algorithm**.

After you have created the algorithm package, you must set the specifications for the training and tuning of your model.

Step 2: To set the training and tuning specifications

1. Enter the **Name** for your algorithm (for example, *my-algorithm*).
2. For **Training image**, paste the full URI location of your training image that was uploaded to Amazon ECR. You can retrieve the URI by locating your image in the [Amazon ECR console](#).
3. Using the dropdown box, choose the **instance types for training** that your training image supports.
4. Under the **Channel specification** section, add a channel for each input dataset that your algorithm supports, up to 20 channels of input sources. For more information, see [Input Data Configuration](#).
5. Choose **Next**.
6. If your algorithm supports hyperparameters and hyperparameter tuning, you must specify the tuning parameters.
7. Choose **Next**.

Note

We highly recommend that your algorithm supports hyperparameter tuning and makes appropriate parameters tunable. This allows data scientists to tune models to get the best results.

After you have set the tuning parameters, if any, you must set the specifications for your inference image.

Step 3: To set inference image specification

1. For **Location of inference image**, paste the URI of the inference image that was uploaded to Amazon ECR. You can retrieve the URI by locating your image in the [Amazon ECR Console](#).
2. Using the dropdown box, choose the supported instance types for your inference image for both real-time inference (also known as *endpoint*) and batch-transform jobs.
3. Choose **Next**.

Before your algorithm can be created and published, validation is necessary to ensure that it functions as expected. This requires that you run both a training job with test data for training and a batch transform job with test data for inference that you provide. The validation specifications tell SageMaker how to perform the validation.

Step 4: To set validation specifications

1. Set **Publish this algorithm in AWS Marketplace** to **Yes**. If you set this to **No**, you can't publish this algorithm later. Choosing **Yes** [certifies](#) your algorithm for AWS Marketplace and requires the validation specification.
2. If this is your first time creating a machine learning package for AWS Marketplace, choose **Create a new role** for the **IAM role**. Amazon SageMaker uses this role when training your algorithm and deploying the subsequent model package. This includes actions such as pulling images from Amazon ECR, storing artifacts in Amazon S3, and copying training data from Amazon S3. Review the settings, and choose **Create role**. Creating a role here grants permissions described by the [AmazonSageMakerFullAccess](#) IAM policy to the role that you create.
3. Edit the **JSON** file in the validation profile for **Training job definition**. For more information about allowed values, see [TrainingJobDefinition](#).
 1. `InputDataConfig`: In this JSON array, add a [Channel object](#) for each channel that you specified in the training-specification step. For each channel, specify where your test data for training is stored.
 2. `OutputDataConfig`: After the training completes, the model artifacts in the training container directory path `/opt/ml/model/` are compressed and copied out to Amazon S3. Specify the Amazon S3 location of where the compressed file (.tar.gz) is stored.
4. Edit the JSON file in the validation profile for **Transform job definition**. For more information about allowed values, see [TransformJobDefinition](#).
 1. `TransformInput.DataSource.S3Uri`: Set to where your test data for inference is stored.
 2. `TransformInput.ContentType`: Specify your test data content type. For example, `application/json`, `text/plain`, `image/png`, or any other value. Amazon SageMaker does not validate the actual input data. This value is passed to your container HTTP endpoint in the `Content-type` header value.
 3. `TransformInput.CompressionType`: Set to `None` if your test data for inference in Amazon S3 is not compressed.

4. `TransformInput.SplitType`: Choose how you want objects in S3 split. For example, `None` passes each object in Amazon S3 as a whole for inference. For more details, see [SplitType](#) in the Amazon SageMaker API Reference.
 5. `TransformOutput.S3OutputPath`: Set to the location where the inference output is stored.
 6. `TransformOutput.AssembleWith`: Set to `None` to output each inference as separate objects in Amazon S3.
5. Choose **Create algorithm package**.

SageMaker pulls the training image from Amazon ECR, runs a test-training job using your data, and stores the model artifacts in Amazon S3. It then pulls the inference image from Amazon ECR, copies the artifacts from Amazon S3 into the inference container, and runs a batch transform job using your test data for inference. After the validation succeeds, the status changes to **Completed**.

Note

The validation step does not evaluate the accuracy of the training or the model with your test data. The validation step checks if the containers run and respond as expected. The validation step only validates batch processing. It is up to you to validate that real-time processing works with your product.

You have completed creating your algorithm product resources. Continue to [Publishing your product in AWS Marketplace](#).

Publishing your product in AWS Marketplace

Before you can publish your model package or algorithm, the following are required:

- An AWS account that is registered as an AWS Marketplace seller. You can do this in the [AWS Marketplace Management Portal](#).
- A completed seller profile under the [Settings](#) page in the AWS Marketplace Management Portal.
- For publishing paid products, you must complete the tax interview and bank forms. This is not required for publishing free products. For more information, see [Seller registration process](#).
- You must have permissions to access the AWS Marketplace Management Portal and Amazon SageMaker. For more information, see [Permissions required](#).

Overview of publishing process

There are four steps in the publishing process:

1. **Submit product** – Create a listing with the description, usage information, and other details of your model package or algorithm product. After you submit your product for publishing, it takes about an hour until the status changes to the next step.
2. **Test product** – Use your AWS account that is registered as an AWS Marketplace seller to preview the listing in the AWS Marketplace, subscribe to it, and test the product. In addition, other allowed AWS accounts can preview and test the product. If any changes are necessary, you can go back and edit the listing details.
3. **Sign off for publishing** – When your product is ready to go live, return to the AWS Marketplace Management Portal, and choose **Sign off and publish**.
4. **Product goes live** – Your product is now live in the AWS Marketplace. You can maintain your product by publishing new versions with updates or product fixes.

Permissions required

To publish an Amazon SageMaker product, the AWS Identity and Access Management user or role you are signed in as requires one or both of the following IAM actions:

- **sagemaker:DescribeModelPackage** – For listing a model package
- **sagemaker:DescribeAlgorithm** – For listing an algorithm

For the AWS Marketplace permissions needed, or for managing your seller account, see [Policies and permissions for AWS Marketplace sellers](#).

Creating your product listing

The following is a walkthrough for creating your product listing in the AWS Marketplace for both model package and algorithm products.

Note

Before creating your listing, ensure that you have the required resources specified in [Requirements and best practices for creating machine learning products](#).

The process has the following steps:

Steps

- [Step 1: Create a new listing](#)
- [Step 2: Provide general product information](#)
- [Step 3: Add your launch option](#)
- [Step 4: Set the pricing and terms](#)
- [Step 5: Submit your product for publishing](#)

Step 1: Create a new listing

To create a new machine learning product listing

1. Sign in to your seller AWS account and navigate to the [AWS Marketplace Management Portal](#).
2. In the top menu, navigate to **Products** and then **Machine learning**.
3. Choose **Create new listing**.

Note

On the **New Product** page, in the **Product summary** section, you can view the current status, privacy setting, product type, creator, and product ID.

Step 2: Provide general product information


To provide general product information

1. In the **General product information** section, for **Product descriptions**, choose **Add**.
 - a. For the **Product visibility** section, choose one of the following options:
 - **Public** – The product will initially be available to a limited set of AWS accounts for testing. After you sign off and publish it, the product is publicly discoverable and available for subscription by all customers.
 - **Private** – The product will only be visible to the AWS accounts that you specify. You will not be able to make this product public in the future.

- b. Enter **Product title**, **Short product description**, **Product overview**, **Product category 1**, and other details. You can change these values later. For product descriptions, see [Requirements and best practices for creating machine learning products](#).
 - c. Choose **Continue** when complete.
2. For **Promotional Resources**, provide a product logo, search keywords, and relevant resource links. You can change these values later.
 - Choose **Continue** when complete.
3. For **Support Information**, choose whether you are offering support for the product.
 - a. If you choose **Yes**, provide support and contact details. You can change these values later.
 - b. Choose **Continue** when complete.
4. For **Region Availability**, choose the specific AWS Regions you want to list your product in.

The default value is **Make available in all current and future supported Regions**.

- Choose **Continue** when complete.

 **Note**

After you submit your draft for publishing, you can't change this selection.

The next step in publishing your product is to provide the launch option, which is the model package or algorithm that you're selling.

Step 3: Add your launch option

To add your launch option

1. In the **Launch option** section, for **Enter ARN**, enter the Amazon Resource Name (ARN) of your model package or algorithm.

You can find the ARN in the Amazon SageMaker console [Model Packages](#) or [Algorithms](#) pages.

Example ARN for a model package

```
arn:aws:sagemaker:<region>:<account-id>:model-package/<model-package-name>
```

Example ARN for an algorithm

arn:aws:sagemaker:<region>:<account-id>:algorithm/<algorithm-name>

2. Choose **Add**.
3. The following steps differ depending on if you publish a model package or algorithm product. With the exception of the buyer-facing version number, you can change the version details later.
 1. For **Step 1: Enter version details and Git repository links**, provide the version number, release notes, and URLs to the sample Jupyter notebook and GitHub repository.
 2. For *algorithm products* only, for **Step 2: Enter details describing the training data inputs**, describe the training data and include an example training data resource along with an overview of the training algorithm.

The algorithm metrics, channel specification, and hyperparameters are automatically displayed on the product detail page based on the values you provided when you created the algorithm resource in SageMaker.

The following examples show how the training data inputs details appear to you as a seller, and how training data inputs details appear to the buyer.

Example Example training data inputs – seller view

Enter details describing the training data inputs

Information to train a model

Describing the training data including an example training data resource along with an overview of training algorithm. Algorithm Metrics, Channel specification and Hyperparameters will be automatically displayed on the product detail page based on details in your algorithm container image See examples [here](#). Note, if you have any details related to trained model input and outputs, please remove from here and enter in Steps 3 and 4.

B I H

Example input(s) for training job:
 [Bank Marketing Dataset from UCI][https://archive.ics.uci.edu/ml/datasets/bank+marketing]
 AutoGluon-Tabular requires no manual data preprocessing as long as your data is a valid CSV table
 Rows in your table represent different examples (data points), columns represent different variables (features).
 The first line of your CSV file should contain names for each column.
 Columns in your CSV file can be strings/text-fields/Numeric.
 Your data must contain the column that you identify as 'label' in your hyperparameter configuration.

572 of 600 characters used

Example Example training data inputs – buyer view

Usage Information

Training

Example input(s) for training job:

[Bank Marketing Dataset from UCI](#)

AutoGluon-Tabular requires no manual data preprocessing as long as your data is a valid CSV table
 Rows in your table represent different examples (data points), columns represent different variables (features).
 The first line of your CSV file should contain names for each column.
 Columns in your CSV file can be strings/text-fields/Numeric.
 Your data must contain the column that you identify as 'label' in your hyperparameter configuration.

[Show less](#)

The following examples show how the custom attributes (invocation parameters) appear to you as a seller, and how custom attributes (invocation parameters) appear to the buyer.

Example Example custom attributes (invocation parameters) – seller view

Custom attributes (invocation parameters) - optional Remove

Names
 Short label for parameter data, can be comma delimited list

threshold

Maximum 100 characters

Description of parameter data
 Brief summary of the parameter field

Threshold of the confidence score of detected objects

53 of 500 characters used

Parameter data type
 Select one

Continuous(float)

Minimum Value - optional **Maximum Value - optional**

0.0 1.0

Maximum 9 characters Maximum 9 characters

Is this parameter variable always required?

Yes

No, it's optional (must enter a default value)

Default Value

0.3

Maximum 200 characters

Example Example custom attributes (invocation parameters) – buyer view

▼ Custom attributes (invocation parameters)

Field name
threshold

Description
Threshold of the confidence score of detected objects

Data type Continuous (Float)	Range Min: 0.0 Max: 1.0	Required No
---------------------------------	-------------------------------	----------------

Default value
0.3

- For **Step 3: Enter input details**, provide the model or algorithm input details and URLs for the sample input files.

The following examples show how the model data inputs details appear to you as a seller, and how model data inputs details appear to the buyer.

Example Example model data inputs – seller view

Model input details

Help customers understand your model capabilities by providing details related to model input (summary, limitations, mime types, and sample data for realtime and batch invocation) and input data descriptions (required for text/csv and application/json mime types). See examples [here](#).

Model input summary
Describe the model input format specification in text.

B I H

This model can analyze images that are supplied as image bytes or stored in an Amazon S3 bucket.

96 of 500 characters used

Limitation for input type - optional
Define any limitations on the input data, such as file size and/or image aspect ratio

The minimum size is 80 pixels for both height and width. The image can be no larger than 1024X1024 pixels, otherwise the detection performance may degrade dramatically. Note, that images with aspect ratio close to 1.0 are best.

227 of 300 characters used

Input mime types
Select all the input data formats permitted

- Select multiple -

image/bmp X image/png X image/jpeg X application/x-image X

Choose to provide raw text or URL for realtime sample input data

URL
 Text

Realtime: Sample input data

Maximum 150 characters

Batch job: Sample input data
Provide URL to a folder that shows an example of a batch input that supports multiple records

Maximum 150 characters

Example Example model data inputs – buyer view

Input

Summary

This model can analyze images that are supplied as image bytes or stored in an Amazon S3 bucket.

Limitations for input type

The minimum size is 80 pixels for both height and width. The image can be no larger than 1024X1024 pixels, otherwise the detection performance may degrade dramatically. Note, that images with aspect ratio close to 1.0 are best.

Input MIME type

image/bmp, image/png, image/jpeg

Sample input data

[view data](#)

- For **Step 4: Enter output details**, provide the model or algorithm output details and sample outputs as text or URLs.

For usage information, see [Requirements and best practices for creating machine learning products](#).

The following examples show how the model data outputs details appear to you as a seller, and how model data outputs details appear to the buyer.

Example Example model data outputs – seller view

Model output details

Help customers understand your model capabilities by providing details related to model output (summary, limitations, mime types, and sample output data for realtime and batch invocation) and output data descriptions (required for text/csv and application/json mime types). See examples [here](#).

Model output summary

Describe the model output format specification in text.

B I H </> ☰ ☷ 🔍 🗨

The model detects instances of common objects such as computer, in an image. The response includes an array of detected object labels (id field) with bounding box pixel coordinates, and an associated level of confidence.

220 of 500 characters used

Limitation for output type - optional

Define any limitations on the output data

Maximum file size is...

0 of 300 characters used

Output mime types

Select one or more output formats

- Select multiple -

application/json ✕

Choose to provide raw text or URL for realtime sample output data

URL

Text

Realtime: Sample output data

Ensure the sample output data corresponds to the input sample data

```
[{"right":603,"bottom":528,"top":177,"score":0.9921523332595825,"id":"person","left":439}, {"right":687,"bottom":539,"top":184,"score":0.9885265231132507,"id":"person","left":577}, {"right":611,"bottom":246,"top":228,"score":0.21156705915927887,"id":"cell phone","left":598}]
```

273 of 1000 characters used

Batch job: Sample output data

Provide a link to a file or folder and ensure example data corresponds to the input sample data

<https://github.com/zhreshold/gluoncv-sagemaker-examples/blob/master/example/input/playground.jpg>

Maximum 150 characters

Example Example model data outputs – buyer view

Output

Summary

The model detects instances of common objects such as computer, in an image. The response includes an array of detected object labels (id field) with bounding box pixel coordinates, and an associated level of confidence.

Output MIME type

application/json

Sample output data

```
[
  {
    "right": 603,
    "bottom": 528,
    "top": 177,
    "score": 0.9921523332595825,
    "id": "person",
    "left": 439
  },
  {
    "right": 687,
    "bottom": 539,
    "top": 184,
    "score": 0.9885265231132507,
    "id": "person",
    "left": 577
  },
  {
    "right": 611,
    "bottom": 246,
    "top": 228,
    "score": 0.21156705915927887,
    "id": "cell phone",
    "left": 598
  }
]
```

5. For **Step 5: Review supported instances and create**, set the recommended instances.
 - If this is a *model package* product, choose the recommended instance type from your supported instances for both the batch transform and real-time deployments.
 - If this is an *algorithm* product, also choose the recommended instance type training jobs.

You can't choose instance types that your model package or algorithm resource doesn't support. The supported instance types were selected when you created those resources in Amazon SageMaker.

4. Choose **Continue** when complete.

Note

Clear usage information that describes the expected inputs and outputs of your product (with examples) is crucial for supporting a positive buyer experience. For more information, see [Requirements and best practices for creating machine learning products](#).

The next step in publishing your product is to set the pricing and terms.

Step 4: Set the pricing and terms

To set the pricing and terms

1. In the **Pricing and terms** section, choose **Add offer**.
2. Set your **Pricing**.

You can provide your software for free, set your paid pricing, or enable a free trial period. For more information, see [Machine learning product pricing](#).

3. Upload a plaintext file to use as your End User License Agreement (EULA).
4. Choose **Save and close**.

You have provided all the information for your product. The next step is to publish it to limited availability so that you can test the product.

Step 5: Submit your product for publishing

To submit your product for publishing

1. On the **New Product** page, in the **Submit for publishing** section, under **Additional test accounts – optional**, enter one or more AWS account IDs for your additional testers.
2. Choose **Submit for publishing**.

This starts the publishing process by creating a preview listing in AWS Marketplace that you (and your optional testers) can subscribe to and use for testing.

You are now ready to test your product. For more information about testing your machine learning product, see [Testing your product](#).

After testing your product, you can redo the steps above if there are any changes that need to be made. When you're ready for your product to be available to buyers, you can [sign off for publishing](#).

Testing your product

After the initial submission of your product, it takes about an hour for your preview listing to be ready. After the status changes to **Test Product**, your seller account and other allow-listed AWS accounts can preview the listing in AWS Marketplace, subscribe to the product, and test it.

To see a preview of your listing

1. In the AWS Marketplace Management Portal, navigate to the **Product Overview** page.
2. Choose **Go to staged product**.
3. If you want to make changes, choose **Edit product** and follow the same steps as [creating your product listing](#).
4. When you're ready for your product to be published publicly for all buyers to see, follow the steps in [Signing off for publishing](#).

To add other AWS accounts to test your product before publishing, contact the [AWS Marketplace Seller Operations team](#) and provide the AWS account IDs. Allow-listed accounts display a **Limited** badge alongside the product version on the product detail page.

Signing off for publishing

This step is to be done after you write your descriptions, pricing, and usage information, and then test your product.

To sign off for publishing

1. Sign in to your seller AWS account and navigate to the [AWS Marketplace Management Portal](#).
2. In the top menu, navigate to **Products** and then **Machine learning**.
3. Navigate to the **Product Overview** of your product.
4. Choose **Sign off and publish**.

Updating your product

You can use the [Machine Learning Listings](#) page in the AWS Marketplace Management Portal to update your model package or algorithm product in the following ways:

- [Add new versions](#) – You can add new model package or algorithm resources as new versions of your existing product.
- [Restrict versions](#) – You can restrict previous versions of your existing product.
- [Remove product](#) – You can remove your entire product.

Adding new versions

To add new versions of your model package or algorithm resources

1. Navigate to the [Machine Learning Listings](#) page in the AWS Marketplace Management Portal.
2. Navigate to the **Product Overview** of your existing product.
3. Choose **Edit product**.
4. Under **Launch option**, choose **Edit**.
5. To add the ARN of your resource, navigate to the **Versions** page, and choose **Add new version**.

For more information about adding a launch option, see [Creating your product listing](#).

Note

Usage information is specific to each product version. Continue to follow the [Requirements and best practices for creating machine learning products](#) when adding usage information to new versions.

When your buyers launch your product from its AWS Marketplace listing, they can choose different versions. When your buyers launch your product from the Amazon SageMaker console, only the latest version is visible.

Restricting versions

To restrict versions of your model package or algorithm resources

1. Navigate to the **Product Overview** of your existing product.
2. Choose **Edit product**.
3. Under **Launch option**, choose **Edit**.
4. On the **Version** page, choose **Restrict version**.
5. Return to the **Product Overview**, and choose **Submit for publishing**.

Note

Buyers that have already subscribed to your product can continue to use restricted versions of your model package or algorithm. However, new buyers will not be able to see those restricted versions as options.

Removing a product

To remove a product

1. Navigate to your list of published products in the [Machine Learning Listings](#) page in the AWS Marketplace Management Portal.
2. Choose the product you want to remove, and in the **Actions** dropdown list, choose **Unpublish listing**.

3. Provide an email address and a reason to remove your listing, in the event that an AWS Marketplace representative contacts you regarding your request.

Note

When you remove a product from AWS Marketplace, new buyers can no longer subscribe to your product. However, existing buyers can continue using your product, which must be supported for a minimum of 90 days. If you plan to have another product replace the unpublished listing, indicate the new listing in the details of your removal request.

Requirements and best practices for creating machine learning products

It is important that your buyers find it easy to test your model package and algorithm products. The following sections describe the requirements for creating machine learning (ML) product listings and best practices for ML products. For a complete summary of requirements and recommendations, see the [Summary of requirements and recommendations for ML product listings](#).

Note

An AWS Marketplace representative might contact you to help you meet these requirements if your published products don't meet them.

Topics

- [Required assets](#)
- [General best practices for ML products](#)
- [Requirements for usage information](#)
- [Requirements for inputs and outputs](#)
- [Requirements for Jupyter notebook](#)
- [Summary of requirements and recommendations for ML product listings](#)

Required assets

Before creating a machine learning product listing, ensure that you have the following required assets:

- **Amazon Resource Name (ARN)** – Provide the ARN of the model package or algorithm resource in the AWS Region that you are publishing from (see [Supported AWS Regions for publishing](#)).
 - An ARN for a model package has this form: `arn:aws:sagemaker:<region>:<account-id>:model-package/<model-package-name>`
 - An ARN for an algorithm has this form: `arn:aws:sagemaker:<region>:<account-id>:algorithm/<algorithm-name>`
- [the section called “Requirements for usage information”](#) – Provide details about inputs, outputs, and code examples.
- [the section called “Requirements for inputs and outputs”](#) – Provide either files or text.
- [the section called “Requirements for Jupyter notebook”](#) – Demonstrate complete product usage.

General best practices for ML products

Provide the following information for your machine learning product:

- For product descriptions, include the following:
 - What your model does
 - Who the target customer is
 - What the most important use case is
 - How your model was trained or the amount of data that was used
 - What the performance metrics are and the validation data used
 - If medical, whether or not your model is for diagnostic use
- By default, machine learning products are configured to have public visibility. However, you can create a product with private visibility. For more information, see [Creating your product listing](#).
- (Optional) For paid products, offer a free trial of 14–30 days for customers to try your product. For more information, see [Machine learning product pricing](#).
- (Optional) For model package products, if you want to enable a real-time product demo on your product listing page, contact the [AWS Marketplace Seller Operations](#) team. The product demo

allows a prospective buyer to try your model directly on the listing page without subscribing to or deploying the model themselves.

Requirements for usage information

Clear usage information that describes the expected inputs and outputs of your product (with examples) is crucial for driving a positive buyer experience.

With each new version of your resource that you add to your product listing, you must provide usage information.

To add usage information for a new product that you are publishing for the first time, sign into the AWS Marketplace Management Portal console. From the **Products** dropdown, choose **Machine learning**. Select your product. In the **Product Overview** under **Launch option**, provide the ARN of your model package or algorithm resource, and choose **Add**.

To edit the existing usage information for a specific version, choose **Edit** under **Launch option** and then **Edit version**.

Requirements for inputs and outputs

A clear explanation of your format, with examples of inputs and outputs, is important to help your buyers to understand and use your product. This understanding helps your buyers to perform any necessary transformations on the input data to get the best inference results.

You will be prompted for the following when adding your Amazon SageMaker resource to your product listing.

Inference inputs and outputs

For inference input, provide the input format for both the real-time endpoint and batch transform job. Include code snippets for any necessary preprocessing of the data. Include supported MIME content types (for example, **image/jpeg**, **image/png**, **image/bmp**), descriptions of values if applicable, and limitations. Include input samples hosted on [GitHub](#).

For inference output, provide the output format for the both real-time endpoint and batch transform job. Include output MIME content type (for example, **application/json**, **image/jpeg**) and description of values if applicable. Include output samples hosted on [GitHub](#).

For samples, provide input files that work with your product. If your model performs multiclass classification, provide at least one sample input file for each class.

Training inputs

In the **Information to train a model** section, provide the input data format and code snippets for any necessary preprocessing of the data. Include supported MIME content types (for example, **image/jpeg**, **image/png**, **image/bmp**), description of values if applicable, and limitations. Ensure to include input samples hosted on [GitHub](#).

Explain both optional and mandatory features that can be provided by the buyer, and specify whether the PIPE input mode is supported. If [distributed training](#) (training with more than 1 CPU/GPU instance) is supported, specify this. For tuning, list the recommend hyperparameters.

Requirements for Jupyter notebook

When adding your SageMaker resource to your product listing, provide a link to a sample Jupyter notebook hosted on [GitHub](#) that demonstrates the complete workflow without asking the buyer to upload or find any data.

Use the AWS SDK for Python (Boto). A well-developed sample notebook makes it easier for buyers to try and use your listing.

For model package products, your sample notebook demonstrates the preparation of input data, creation of an endpoint for real-time inference, and performance of batch-transform jobs. For more information, see [Model Package listing and Sample notebook](#) on GitHub. For sample notebooks, see [generic_sample_notebook](#) and [auto_insurance](#). The latter sample notebook works in all AWS Regions, without entering any parameters and without a buyer needing to locate sample data.

Note

An underdeveloped sample Jupyter notebook that does not show multiple possible inputs and data preprocessing steps might make it difficult for the buyer to fully understand your product's value proposition.

For algorithm products, the sample notebook demonstrates complete training, tuning, model creation, the creation of an endpoint for real-time inference, and the performance of batch-transform jobs (see [Algorithm listing and Sample notebook](#) on GitHub). For sample notebooks,

see [amazon_demo_product](#) and [automl](#) on GitHub. These sample notebooks work in all Regions without entering any parameters and without a buyer needing to locate sample data.

Note

A lack of example training data might prevent your buyer from running the Jupyter notebook successfully. An underdeveloped sample notebook might prevent your buyers from using your product and hinder adoption.

Summary of requirements and recommendations for ML product listings

The following table provides a summary of the requirements and recommendations for a machine learning product listing page.

Details	For model package listings	For algorithm listings
Product descriptions		
Explain in detail what the product does for supported content types (for example, "detects X in images").	Required	Required
Provide compelling and differentiating information about the product (avoid adjectives like "best" or unsubstantiated claims).	Recommended	Recommended
List most important use case(s) for this product.	Required	Required
Describe the data (source and size) it was trained on and list any known limitations.	Required	Not applicable

Details	For model package listings	For algorithm listings
Describe the core framework that the model was built on.	Recommended	Recommended
Summarize model performance metric on validation data (for example, "XX.YY percent accuracy benchmarked using the Z dataset").	Required	Not applicable
Summarize model latency and/or throughput metrics on recommended instance type.	Required	Not applicable
Describe the algorithm category. For example, "This decision forest regression algorithm is based on an ensemble of tree-structured classifiers that are built using the general technique of bootstrap aggregation and a random choice of features."	Not applicable	Required
Usage information		
For inference, provide the input format for both the real-time endpoint and batch transform job. Include supported MIME content types (for example, image/jpeg , image/png , image/bmp), description of values if applicable, and limitations. See Requirements for inputs and outputs .	Required	Required

Details	For model package listings	For algorithm listings
<p>For inference, provide input samples for both the real-time endpoint and batch transform job. Samples must be hosted on GitHub. See Requirements for inputs and outputs.</p>	Required	Required
<p>For inference, provide the output format for both the real-time endpoint and batch transform job. Include output MIME content type (for example, application/json, image/jpeg) and description of values if applicable. See Requirements for inputs and outputs.</p>	Required	Required
<p>For inference, provide output samples for both the real-time endpoint and batch transform job. Samples must be hosted on GitHub. See Requirements for inputs and outputs.</p>	Required	Required
<p>For inference, provide an example of using an endpoint or batch transform job. Include a code example using the AWS Command Line Interface (AWS CLI) commands or using an AWS SDK.</p>	Required	Required

Details	For model package listings	For algorithm listings
<p>For training, provide input format. Include supported MIME content types (for example, image/jpeg, image/png, image/bmp), description of values if applicable, and limitations (for example, minimum rows of data required). See Requirements for inputs and outputs.</p>	Not applicable	Required
<p>For training, provide input samples hosted on GitHub. See Requirements for inputs and outputs.</p>	Not applicable	Required
<p>For training, provide an example of performing training jobs. Describe the supported hyperparameters, their ranges, and their overall impact. Specify if the algorithm supports hyperparameter tuning, distributed training, or GPU instances. Include code example such as AWS CLI commands or using an AWS SDK, for example.</p>	Not applicable	Required
<p>Provide a Jupyter notebook hosted on GitHub demonstrating complete use of your product. See Requirements for Jupyter notebook.</p>	Required	Required

Details	For model package listings	For algorithm listings
Provide technical information related to the usage of the product, including user manuals and sample data.	Recommended	Recommended

Service restrictions and quotas

This section describes restrictions and quotas on your machine learning (ML) products in AWS Marketplace.

Network isolation

For security purposes, when a buyer subscribes to your containerized product, the Docker containers are run in an isolated environment without network access. When you create your containers, don't rely on making outgoing calls over the internet because they will fail. Calls to AWS services will also fail.

Image size

Your Docker image size is governed by the Amazon Elastic Container Registry (Amazon ECR) [service quotas](#). The Docker image size affects the startup time during training jobs, batch-transform jobs, and endpoint creation. For better performance, maintain an optimal Docker image size.

Storage size

When you create an endpoint, Amazon SageMaker attaches an Amazon Elastic Block Store (Amazon EBS) storage volume to each ML compute instance that hosts the endpoint. (An endpoint is also known as *real-time inference* or *Amazon SageMaker hosting service*.) The size of the storage volume depends on the instance type. For more information, see [Host Instance Storage Volumes](#) in the *Amazon SageMaker Developer Guide*.

For batch transform, see [Storage in Batch Transform](#) in the *Amazon SageMaker Developer Guide*.

Instance size

SageMaker provides a selection of instance types that are optimized to fit different ML use cases. Instance types are comprised of varying combinations of CPU, GPU, memory, and networking

capacity. Instance types give you the flexibility to choose the appropriate mix of resources for building, training, and deploying your ML models. For more information, see [Amazon SageMaker ML Instance Types](#).

Payload size for inference

For an endpoint, limit the maximum size of the input data per invocation to 6 MB. This value can't be adjusted.

For batch transform, the maximum size of the input data per invocation is 100 MB. This value can't be adjusted.

Processing time for inference

For an endpoint, the maximum processing time per invocation is 60 seconds. This value can't be adjusted.

For batch transform, the maximum processing time per invocation is 60 minutes. This value can't be adjusted.

Service quotas

For more information about quotas related to training and inference, see [Amazon SageMaker Service Quotas](#).

Asynchronous inference

Model packages and algorithms published in AWS Marketplace can't be deployed to endpoints configured for [Amazon SageMaker Asynchronous Inference](#). Endpoints configured for asynchronous inference requires models to have network connectivity. All AWS Marketplace models operate in network isolation. For more information, see [No network access](#).

Serverless inference

Model packages and algorithms published in AWS Marketplace can't be deployed to endpoints configured for [Amazon SageMaker Serverless Inference](#). Endpoints configured for serverless inference require models to have network connectivity. All AWS Marketplace models operate in network isolation. For more information, see [No network access](#).

Managed spot training

For all algorithms from AWS Marketplace, the value of `MaxWaitTimeInSeconds` is set to 3,600 seconds (60 minutes), even if the checkpoint for [managed spot training](#) is implemented. This value can't be adjusted.

Docker images and AWS accounts

For publishing, images must be stored in Amazon ECR repositories owned by the AWS account of the seller. It isn't possible to publish images that are stored in a repository owned by another AWS account.

Publishing model packages from built-in algorithms or AWS Marketplace

Model packages created from training jobs using an [Amazon SageMaker built-in algorithm](#) or an algorithm from an AWS Marketplace subscription can't be published.

You can still use the model artifacts from the training job, but your own inference image is required for publishing model packages.

Supported AWS Regions for publishing

AWS Marketplace supports publishing model package and algorithm resources from AWS Regions where the following are both true:

- A Region that [Amazon SageMaker supports](#)
- An [available Region](#) that is opted-in by default (for example, [describe-regions](#) returns `"OptInStatus": "opt-in-not-required"`)

All assets required for publishing a model package or algorithm product must be stored in the same Region that you choose to publish from. This includes the following:

- Model package and algorithm resources that are created in Amazon SageMaker
- Inference and training images that are uploaded to Amazon ECR repositories
- Model artifacts (if any) that are stored in Amazon Simple Storage Service (Amazon S3) and dynamically loaded during model deployment for model package resources
- Test data for inference and training validation that are stored in Amazon S3

You can develop and train your product in any Region that is supported by SageMaker. But, before you can publish, you must copy all assets to and re-create resources in a Region that AWS Marketplace supports publishing from.

During the listing process, regardless of the AWS Region that you publish from, you can choose the Regions that you want to publish to and make your product available in.

Troubleshooting

This section provides help for some common errors that you might encounter during the publishing process of your machine learning product. If your issue isn't listed, contact the [AWS Marketplace Seller Operations](#) team.

General: I get a 400 error when I add the Amazon Resource Name (ARN) of my model package or algorithm in the AWS Marketplace Management Portal

If you used the Amazon SageMaker console to create your resource, you must choose **Yes** on the final page of the process for **Publish this model package in AWS Marketplace** or **Yes** for **Publish this algorithm in AWS Marketplace**. You can't choose **No** and later publish it. Selecting **Yes** doesn't publish the model package or algorithm. However, it validates your model package or algorithm resource when it is created, which is necessary for use in AWS Marketplace.

If you're using the AWS SDK to [create a model package](#) or [create an algorithm](#), ensure that the parameter `CertifyForMarketplace` is set to `true`.

After you re-create your certified and validated model package or algorithm resource, add the new ARN in the AWS Marketplace Management Portal.

General: I get a 404 error when I add the ARN of my model package or algorithm in the AWS Marketplace Management Portal

This error can happen for several reasons:

- The ARN might be invalid. Ensure that you are using the correct ARN.
 - For model packages, the ARNs should look similar to `arn:aws:sagemaker:us-east-2:000123456789:model-package/my-model-package-name`.
 - For algorithms, the ARNs should look similar to `arn:aws:sagemaker:us-east-2:000123456789:algorithm/my-algorithm`.

- The model package or algorithm resource wasn't created in the same AWS account as the seller account. Ensure that all resources and assets for publishing are in the seller account that you are publishing from.
- The user or role that you use for publishing doesn't have the correct IAM permissions to access the model package or algorithm resource. Ensure that your user or role has the following permissions:
 - For model packages, the action `sagemaker:DescribeModelPackage` on the model package resource must be allowed.
 - For algorithms, the action `sagemaker:DescribeAlgorithm` on the algorithm resource must be allowed.

General: I get a 500 error when I specify the pricing for my algorithm product in the AWS Marketplace Management Portal

This error can happen when you attempt to publish an algorithm resource with only a training image and without an accompanying inference image. Algorithm resources that are published on AWS Marketplace must have both components. For more information, see [Prepare your product in SageMaker](#).

Amazon SageMaker: I get a “Client error: Access denied for registry” failure message when I create a model package or algorithm resource

This error can happen when the image that is being used to create the model package or algorithm is stored in an [Amazon ECR](#) repository that belongs to another AWS account. Model package or algorithm validation does not support cross-account images. Copy the image to an Amazon ECR repository owned by the AWS account that you are using to publish. Then, proceed with creating the resource using the new image location.

Amazon SageMaker: I get “Not Started” and “Client error: No scan scheduled...” failure messages when I create a model package or algorithm resource

This error can happen when SageMaker fails to start a scan of your Docker container image stored in Amazon ECR. If this happens, open the [Amazon ECR console](#), find the repository where your image was uploaded to, choose the image, and then choose **Scan**.

Reporting

AWS Marketplace produces reports for your Amazon SageMaker products that include data about buyers, financials, usage, and taxes. All reports are available in the AWS Marketplace Management Portal on the [Reports page](#). For more information, see [Seller Reports](#).

The following subsections summarize how financials for your machine learning products that use SageMaker are reported.

Daily business report

The daily business report provides the instance type, hours of usage, revenue from software charges, and other details for each buyer and product. Buyers are identified by a unique and anonymous Customer Reference ID. For more information, see [Daily business report](#).

Monthly revenue report

The monthly revenue report provides you with the monthly revenue that has been billed to your buyers for using your software. For more information, see [Monthly billed revenue report](#).

Disbursement report

The monthly disbursement report provides a breakdown of all funds collected on your behalf during the settlement period for your software charges. The total settlement amount reflected in the report should match the amount deposited to your bank account. For more information, see [Disbursement report](#).

Other reports and analysis

For other available reports, see [Seller reports](#).

You can also create custom reports using the available [Seller delivery data feeds service](#) from AWS Marketplace.

SaaS-based products

With software as a service (SaaS) products, you deploy software hosted on AWS infrastructure and grant buyers access to the software in your AWS environment. You are responsible for managing customer access, account creation, resource provisioning, and account management within your software.

For information about integrating your APIs with Amazon API Gateway, see [Sell your API Gateway APIs through AWS Marketplace](#) in the *Amazon API Gateway Developer Guide*.

For assistance with your SaaS products, contact the [AWS Marketplace Seller Operations](#) team.

Topics

- [Getting started with SaaS products](#)
- [Plan your SaaS product](#)
- [SaaS product guidelines](#)
- [SaaS product pricing](#)
- [SaaS free trials](#)
- [SaaS customer onboarding](#)
- [Amazon SNS notifications for SaaS products](#)
- [Accessing the AWS Marketplace Metering and Entitlement Service APIs](#)
- [Reporting](#)
- [Code examples for SaaS product integration](#)
- [Using AWS PrivateLink with AWS Marketplace](#)

Getting started with SaaS products

This chapter outlines how software as a service (SaaS) products work for sellers who create and maintain them. This section describes how to get your SaaS product in AWS Marketplace. Also described is how to integrate your SaaS product with the appropriate AWS Marketplace API operations, based on the SaaS product's billing model.

Prerequisites

Before you get started, you must complete the following prerequisites:

1. Access and use the [AWS Marketplace Management Portal](#). This is the tool that you use to register as a seller and manage the products that you sell in AWS Marketplace. For more information, see [AWS Marketplace Management Portal](#).
2. Register as a seller, and submit your tax and banking information. For more information, see [Seller registration process](#).
3. Plan how you'll create and integrate your SaaS product in AWS Marketplace. For more information, see [Plan your SaaS product](#).

Topics

- [SaaS product lifecycle](#)
- [Creating a SaaS product](#)
- [Create an initial SaaS product page](#)
- [SaaS product settings](#)
- [Integrate your SaaS subscription product](#)
- [Integrate your SaaS contract product](#)
- [Integrate your SaaS contract with pay-as-you-go product](#)
- [Deploy a serverless SaaS integration solution](#)

SaaS product lifecycle

When you create a SaaS product in AWS Marketplace, it's initially published with limited visibility so that only your account can access it. When you're ready, you can publish it to the AWS Marketplace catalog to allow buyers to subscribe and purchase your product.

On the SaaS product page, you can view the list of your products. Depending on its stage in the product lifecycle, the product will have one of the following statuses:

- **Staging** – An incomplete product for which you're still adding information. At the first **Save and exit** from the self-service experience, the successful change request creates an unpublished product with information from the completed steps that you submitted. From this status, you can continue adding information to the product or change already submitted details through change requests.
- **Limited** – A product is complete after it is submitted to the system and passes all validation in the system. Then the product is released to a **Limited** status. At this point, the product has a detail page that is only accessible to your account and whoever you have allowlisted. You

can test your product through the detail page. For more information or help, contact the [AWS Marketplace Seller Operations](#) team.

- **Public** – When you're ready to publish the product so that buyers can view and subscribe to the product, you use the **Update visibility** change request. This request initiates a workflow for the AWS Marketplace Seller Operations team to review and audit your product against AWS policies. After the product is approved and the change request is processed, the product is moved from a status of **Limited** to **Public**. For information about AWS guidelines, see [SaaS product guidelines](#).
- **Restricted** – If you want to stop new users from subscribing to your product, you can restrict the product by using the **Update visibility** change request. A **Restricted** status means that existing users can continue to use the product. However, the product will no longer be visible to the public or be available to new users.

You can update your product at the **Staging**, **Limited**, and **Public** statuses. For more information, see [Updating product information](#).

Creating a SaaS product

To sell software as a service (SaaS) products in AWS Marketplace, you must complete the following tasks:

1. Create the SaaS product in AWS Marketplace.
2. Integrate the SaaS [subscription](#), [contract](#), or [contract with pay-as-you-go](#) product with AWS Marketplace.
3. Test the [subscription](#), [contract](#), or [contract with pay-as-you-go](#) product's integration with AWS Marketplace.
4. Submit your product for launch.

Topics

- [Create a SaaS product using self-service](#)
- [Create a SaaS product \(legacy\)](#)

Create a SaaS product using self-service

To create a SaaS product in AWS Marketplace using self-service

1. **Decide to list a SaaS product**

Have a SaaS product that you would like to sell in AWS Marketplace. Review and understand how to [Plan your SaaS product](#).

2. Determine pricing and offer type

There are three offer types for SaaS products: subscriptions, contracts, and contracts with pay-as-you-go. Your choice of offer type affects how you integrate your SaaS product with AWS Marketplace. For more information, see [Plan your pricing](#).

3. Collect assets

Collect the assets that you will need to use to submit your product. Assets for your product include:

- Product logo URL – A publicly accessible Amazon S3 URL that contains a clear image of the logo for the product that you're providing.
- End User License Agreement (EULA) URL – Your product must have a EULA that's available as a PDF file. You must provide a link to an Amazon S3 bucket where customers can review the EULA on your product's AWS Marketplace page.
- Product registration URL – This is the URL where buyers are redirected after successfully subscribing to your product in AWS Marketplace.
- Metadata about your product – You provide the metadata in the product creation wizard of the AWS Marketplace Management Portal.
- Support information for your product – This information includes email addresses and URLs for your product's support channels.

4. Submit your product for integration

[Create an initial SaaS product page](#) from your seller account using AWS Marketplace Management Portal. AWS Marketplace will publish your product as a limited product, which means that it's only available to use for integration and testing. Your product code and Amazon Simple Notification Service (SNS) topics will be available to you on the product overview page.

Note

Your product must remain at a reduced price so you and the AWS Marketplace Seller Operations team can test your product without incurring a large cost. We'll ask you for the product's actual price when you request public visibility for your product.

5. Integrate with AWS Marketplace

Your product must support customers onboarding and using your product, including validating their subscription before giving them access, and, in some cases, metering for their usage. How you integrate with AWS Marketplace depends on the offer type you're using for your product. For more information about integration, based on offer type, see the following topics:

- [Subscription integration](#)
- [Contract integration](#)
- [Contract with pay-as-you-go integration](#)

The final step of integrating your product with AWS Marketplace is to test it to ensure that the integration works properly.

6. Submit your product for launch

After you verify your integration and you're ready for the product to be live, choose **Update visibility**. The AWS Marketplace Seller Operations team will review your product and update the price before the visibility can be updated to Public.

Note

Product verification and publication is a manual process, which is handled by the AWS Marketplace Seller Operations team. If there are no errors, it can take 7–10 business days to publish your initial product listing version. For more details about timing, see [Timing and expectations](#).

Create a SaaS product (legacy)

To create a SaaS product in AWS Marketplace (legacy)

1. Decide to list a SaaS product

Have a SaaS product that you would like to sell in AWS Marketplace. Review and understand how to [Plan your SaaS product](#).

2. Determine pricing model

There are three pricing models for SaaS products: subscriptions, contracts, and contracts with pay-as-you-go . Your choice of offer type affects how you integrate your SaaS product with AWS Marketplace. For more information, see [Plan your pricing](#).

3. Collect assets


Collect the assets that you will need to use to submit your product. Assets for your product include:

- Product logo URL – A publicly accessible URL that contains a clear image of the logo for the product you're providing.
- End User License Agreement (EULA) URL – Your product must have a EULA, and you must provide a link to it for customers to read and review on your product's AWS Marketplace page.
- Product registration URL – This URL is where customers are sent after subscribing to your product in AWS Marketplace.
- Metadata about your product – You provide the metadata in the product creation wizard of the AWS Marketplace Management Portal.
- Support information for your product – This includes email addresses and URLs for your product's support channels.

4. Submit your product for integration

[Create an initial SaaS product page](#) from your seller account using AWS Marketplace Management Portal. AWS Marketplace will publish your product as a limited product, which means that it's only available to your accounts to use for integration and testing. The AWS Marketplace Seller Operations team will send you an email message with your product code, Amazon Simple Notification Service (Amazon SNS) topics, and product page URL. With that information, you will have an environment to use for creating and testing your integration

with AWS Marketplace in your product. Use the email message that you received from the AWS Marketplace Seller Operations team for correspondence regarding the product.

 **Note**

Your product must remain at a reduced price so you and the AWS Marketplace Seller Operations team can test your product without incurring a large cost. We'll ask you for the product's actual price when you request public visibility for your product.

5. Integrate with AWS Marketplace

Your product must support customers onboarding and using your product, including validating their subscription before giving them access, and, in some cases, metering for their usage. How you integrate with AWS Marketplace depends on the offer type you're using for your product. For more information about integration, based on offer type, see the following topics:

- [Subscription integration](#)
- [Contract integration](#)
- [Contract with pay-as-you-go integration](#)

The final step of integrating your product with AWS Marketplace is to test it to ensure that the integration works properly.

6. Submit your product for launch

After you have verified your integration, and you're ready for the product to be live, submit it to the AWS Marketplace Seller Operations team (using the email case created earlier) for end-to-end testing and launch.

7. Launch

After end-to-end testing is complete, you must review the product page with the original prices. Approve the page by responding to the email case that you received when you created your product (see [Creating a SaaS product](#)). After your approval, the AWS Marketplace Seller Operations team will make the product page live on AWS Marketplace. At this point, customers can start discovering and subscribing to your product.

Create an initial SaaS product page

Use your software as a service (SaaS) application metadata, and create an initial SaaS product in the AWS Marketplace catalog, using the AWS Marketplace Management Portal.

To create an initial SaaS product page

1. Sign in to the [AWS Marketplace Management Portal](#).
2. For **Products**, choose **SaaS**.

Note

If you're creating a SaaS listing in one of the AWS GovCloud (US) Regions, use the [???](#) process.

3. Choose **Create SaaS product**, and then choose **SaaS product**.
4. Generate a SaaS product ID and code. You can also add optional tags to support tag-based authorization.

Note

For information about tag-based authorization, see [Controlling access to AWS resources using tags](#) in the *AWS Identity and Access Management User Guide*.


5. Use the self-service experience to create the AWS Marketplace listing. Add product information, product deployment details, and public offer details. Optionally, you can also add accounts to the allowlist to test the product.

Note

If you need to end your session before finishing the steps, choose the **Save and exit** option to save your current selections to the staging area. This option creates a request to validate the information that you provided. While your request is being validated, you can't edit the product. If your request is successful, you can continue creating your product by choosing **Resume product creation**.


If your request isn't successful, it's because of a validation error, which is visible on the product request log. Select the request to view the error, and choose **Copy to new**

under **Actions** to correct the error and resubmit the request. To update previous steps, open the product detail page and submit a change request.

 **Note**

Your price will default to \$0.01 per dimension during testing. This price allows you to test your product in the **Limited** state without incurring a large bill. You'll provide your actual price when making your product public.

6. Choose **Submit**. Then, AWS Marketplace validates the information. If the validation succeeds, AWS Marketplace releases the product in a **Limited** status. After the validation succeeds, you can preview, integrate, and test your product.

 **Note**

While the validation is in progress, you can't edit the product. When your product is initially published, it's only accessible for the AWS account used to create the product and the AWS Marketplace Seller Operations team's test account. If you view the product from the **SaaS products** page, you can choose **View on AWS Marketplace** to view the product details as they will appear in AWS Marketplace for buyers. This detail listing isn't visible to other AWS Marketplace users.

SaaS product settings

After you [create a software as a service \(SaaS\) product](#) in AWS Marketplace, you can modify many of the product's settings when necessary. For more information, see the following topics.

Topics

- [Manage change requests](#)
- [Update product information](#)
- [Update the allowlist of AWS account IDs](#)
- [Update product visibility](#)
- [Update pricing terms](#)
- [Add pricing dimensions](#)

- [Update pricing dimensions](#)
- [Restrict pricing dimensions](#)
- [Determine how buyers will access your product](#)
- [Update availability by country](#)
- [Update the refund policy of a product](#)
- [Update the end user license agreement \(EULA\)](#)

Manage change requests

In a [self-service listing](#), you use a *change request* to make changes to your product. Your current requests can be found on the AWS Marketplace Management Portal on the [Requests tab](#). You can make new requests through the **Request changes** dropdown list that is located under the navigation bar.

To create a change request for a SaaS product

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and sign in to your seller account.
2. From the **Products** tab, select **SaaS** from the dropdown list.
3. After the request is submitted, it begins processing. The change request goes through the following statuses: **Under review**, **Preparing changes**, and **Applying changes**.
4. When the request's processing is completed, its status changes to one of the following values:
 - **Succeeded** – This status indicates that your requested change was processed and changes are reflected in the system.
 - **Failed** – This status indicates that something went wrong with the request and the changes were not processed. If the status is **Failed**, you can select the request to find **Error Codes** that provide recommendations on how to correct the issue. You can troubleshoot the errors and create a new request for the change. To make the process faster, you can use a **Copy to new request** function which copies the details of the **Failed** request. You can make needed changes and resubmit the request.

Change requests that start with an update will load the current details of the project. Then, you can make updates, which overwrite the existing details. Add and restrict request pairs are specifically for updates that are provisioned after each request succeeds (after you choose **Save**

and exit and **Submit** actions in the self-service experience). This means existing subscribers can continue to use the product until their subscription or contract ends. However, no new subscribers can be added to a product that is in a **Restricted** status.

Update product information

After you create your product, you might want to change the information associated with it in AWS Marketplace.

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and sign in to your seller account.
2. From the [SaaS Products](#) page, on the **SaaS products** tab, select the product that you want to modify.
3. From the **Request changes** dropdown list, choose **Update product information**.
4. Update any of the following fields that you want to change:
 - **Product title**
 - **SKU**
 - **Short description**
 - **Long description**
 - **Product logo URL**
 - **Highlights**
 - **Product categories**
 - **Keywords**
 - **Product video URL**
 - **Resources**
 - **Support information**

Note

For details about the logo format, see [Company and product logo requirements](#).

5. To update the product information, choose **Submit**.
6. Verify that the request appears on the **Requests** tab with the **Under review** status. You might need to refresh the page to see your new request.

Update the allowlist of AWS account IDs

You can change the list of AWS account IDs that can view your product in a limited state.

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and sign in to your seller account.
2. From the [SaaS products](#) page, on the **SaaS products** tab, select the product that you want to modify.
3. From the **Request changes** dropdown list, select **Update allowlist**. A list shows the AWS account IDs that are currently allowlisted.
4. In the **Allowlisted AWS accounts** field, enter the AWS account IDs and separate them using a comma.
5. To update the allowlist of AWS account IDs, choose **Submit**.

Update product visibility

To change which buyers can view your Quick Launch experience in AWS Marketplace, you can use **Update visibility**.

1. Open the [AWS Marketplace Management Portal](#), and then sign in to your seller account.
2. From the [SaaS products](#) page, select the product that you want to modify.
3. From the **Request changes** dropdown, choose **Update visibility**.

Note

You can request that the product be moved from a **Limited** status to a **Public** status by using this change request. However, the change request must go through an AWS Marketplace Seller Operations team approval process to be moved to **Public**.

4. When you publish to public, you'll provide the actual price for your product. This price will be applied after your listing is approved for public visibility.
5. To submit your request for review, choose **Submit**.
6. Verify that the **Requests** tab shows the **Request status** as **Under review**. When the request completes, the status becomes **Succeeded**.

Update pricing terms

If you want to change the pricing per dimension on your SaaS product, then you can use **Update pricing terms**.

Note

A pricing increase for any dimension will result in the pricing update option being unavailable for at least the next 90 days. If updating both a price decrease and an increase, update the price decrease first.

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and sign in to your seller account.
2. From the [SaaS Products](#) page, on the **SaaS products** tab, select the product that you want to modify.
3. From the **Request changes** dropdown list, select **Update public offers**, and then select **Update pricing terms**.
4. Current pricing is pre-filled in the fields. You can delete the current price, and then add your new price.
5. To submit your request for review, choose **Submit**.
6. Verify that the **Requests** tab shows the **Request status** as **Under review**. When the request completes, the status will update to **Succeeded** or **Failed**.

Add pricing dimensions


You can add a dimension that you want to use to charge your product. A dimension is the foundational unit of measure that your buyer is charged for when using your product.

Note

To update the name or description of an existing pricing dimension, see [the section called "Update pricing dimensions"](#).


1. Open the [AWS Marketplace Management Portal](#), and then sign in to your seller account.

2. From the [SaaS Products](#) tab, select the product that you want to modify.
3. From the **Request changes** dropdown, choose **Update pricing dimensions** and then **Add pricing dimensions**.
4. Provide a dimension API identifier, display name, and description to add a new dimension to your product, and then choose **Next**.

 **Note**

The API identifier and name must be unique across all dimensions. You can't change the API identifier and unit after the dimension is created.

5. Define the prices for each dimension you've added, and then choose **Next** to review your changes.

 **Note**

You can only add dimensions for the pricing model selected for your product (for example, contract, usage, or contract with consumption). For limited products, the prices for the newly added dimensions are set to \$0.01. You can update the prices when the product is ready for public visibility.

6. Choose **Submit** to submit your request for review.
7. In the **Requests** tab, verify that the request status is **Under review**. When the request is complete, the status will change to **Succeeded**.

Update pricing dimensions

You can update a dimension that you want to use to charge your product. A dimension is the foundational unit of measure that your buyer is charged for when using your product.

1. Open the [AWS Marketplace Management Portal](#), and then sign in to your seller account.
2. From the [SaaS Products](#) tab, select the product that you want to modify.
3. From the **Request changes** dropdown, choose **Update pricing dimensions** and then **Update dimension information**.
4. Find the dimension you want to update, and then choose the name or description.
5. Provide the new name or description, and then choose the **checkmark** to confirm your update.

Note

The dimension name must be unique.

6. Choose **Submit** to submit your request for review.
7. In the **Requests** tab, verify that the request status is **Under review**. When the request is complete, the status will change to **Succeeded**.

Restrict pricing dimensions

You can restrict a dimension that is currently listed in the product. This request removes the selected dimension from the product.

1. Open the [AWS Marketplace Management Portal](#), and then sign in to your seller account.
2. From the [SaaS Products](#) tab, select the product that you want to modify.
3. From the **Request changes** dropdown, choose **Update pricing dimensions** and then **Restrict pricing dimensions**.
4. For limited and public products, you'll be prompted to contact the AWS Marketplace Seller Operations team using the [Contact Us](#) button. Using the form, provide details for the dimensions you want to remove from your product listing.

Determine how buyers will access your product

You can choose one of the following options for how customers can access your product:

- [the section called "Update the SaaS URL fulfillment option"](#) – Customers use a URL for the site that they are redirected to after subscribing to your product in AWS Marketplace.
- [the section called "Configure Quick Launch"](#) – Customers use a simplified process to configure and launch your product. You can complete this configuration for existing products with either Limited or Public visibility.

Update the SaaS URL fulfillment option

To update the URL that is used to fulfill your SaaS product, use the **Update fulfillment options** tab.

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and sign in to your seller account.
2. From the [SaaS Products](#) page, on the **SaaS products** tab, select the product that you want to modify.
3. From the **Request changes** dropdown list, select **Update public offers**, and then select **Edit default fulfillment URL**.
4. In the **Fulfillment URL** field, enter the new URL for the SaaS product fulfillment option.
5. To submit your request for review, choose **Submit**.
6. Verify that the **Requests** tab shows the **Request status** as **Under review**. When the request completes, the status will update to **Succeeded** or **Failed**.

Configure Quick Launch

SaaS products listed in AWS Marketplace often require AWS resources to be deployed in the subscribing buyer's account (for example, IAM roles). Quick Launch allows you to provide your buyers with guided, step-by-step instructions and resource deployment using AWS CloudFormation templates. Buyers use the CloudFormation templates to configure and launch products.

Tip

To learn more about the Quick Launch configuration process, see the [Enable SaaS Quick Launch](#) lab.

To configure a Quick Launch experience that customers can use to launch your SaaS product, use the **Fulfillment options** tab.


1. From the [SaaS Products](#) page, on the **SaaS products** tab, select the product that you want to modify.

Note

To configure the Quick Launch experience, the product must have either Limited or Public visibility.


2. On the product detail page, choose the **Fulfillment options** tab.

3. For **Quick Launch**, choose the **Activate and configure** button.
4. For **Account login details**, provide a URL for your site where the buyer can log in or create an account. This URL opens a new tab in the buyer experience. Buyers then log in or create an account and return to AWS Marketplace to launch the template.
5. For **AWS CloudFormation template**, choose the **Add AWS CloudFormation template** button and provide the following information:
 1. Title – Provide the name of your CloudFormation deployment.
 2. Description – Provide a description of the template.
 3. Stack name – Provide a name for the stack. This name is the stack name for the buyer in CloudFormation.
 4. CloudFormation template URL – Provide the Amazon Simple Storage Service (Amazon S3) URL for the template. AWS will review this template, and AWS will provide the final template URL.

 **Note**

To simplify the launch process for your customers, we suggest minimizing the number of templates that are associated with your configuration process. Ideally, you want one template that deploys the resources needed to use the product. For questions related to your CloudFormation template, contact your AWS Marketplace business development partner or the [AWS Marketplace Seller Operations](#) team.

5. Required IAM permissions – Provide the permissions that are required to deploy the CloudFormation template. If you want to share deployment parameters, which are stored as secrets in [AWS Secrets Manager](#) for the buyer, your policy must include the following actions:
 - `secretsManager:ListSecrets`
 - `secretsManager:DescribeSecret`
 - `secretsManager:ReplicateSecretToRegions`
 - `secretsManager:GetSecretValue`

 **Note**

If your product requires seller-provided CloudFormation deployment parameters (for example, API keys and [external IDs](#)), use the `PutDeploymentParameter`

operation to share the parameter with your customers. For more information, see [PutDeploymentParameter](#) in the *AWS Marketplace Deployment Service API Reference*.

- (Optional) For **Manual configuration instructions**, provide instructions for buyers who want to configure your product manually. Consider including links to your product's onboarding guide and documentation.
- For **Launch details**, provide the URL where buyers will access the product after the CloudFormation stack has been deployed.
- (Optional) For **Allowlisted accounts for Quick Launch**, provide a comma-separated list of AWS accounts that can view the Quick Launch experience with Limited visibility.
- Choose the **Submit** button. The Quick Launch experience will have Limited visibility, meaning it will only be visible to your account and allowlisted accounts. With Limited visibility, you can test your configuration using the **Configure and launch** page after subscribing to your product and choosing the **Set up your account** button.
- When you're ready, you can publish the Quick Launch experience in the AWS Marketplace catalog. Use the **Update Quick Launch visibility** button on the **Fulfillment options** tab on the product detail page.

When you change the visibility to Public, the AWS Marketplace Seller Operations team will review the configuration, conduct buyer testing, and publish the experience.

Note

If you need support as you enable the Quick Launch experience, contact the [AWS Marketplace Seller Operations](#) team.

Update availability by country

You can define the countries in which your product can be offered.

- Open the [AWS Marketplace Management Portal](#), and then sign in to your seller account.
- From the [SaaS Products](#) tab, select the product that you want to modify.
- From the **Request changes** dropdown, select **Update public offer** and then choose **Update availability by country**.
- Choose one of the following options:

- All countries – Available in all supported countries.
 - All countries with exclusions – Available in all supported countries except in selected countries.
 - Allowlisted countries only – Specific list of countries where the product is available.
5. Choose **Submit** to submit your request for review.
 6. In the **Requests** tab, verify that the request status is **Under review**. When the request is complete, the status will change to **Succeeded**.

Update the refund policy of a product

You can update the refund policy for your product by using **Update refund policy**.

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and sign in to your seller account.
2. From the **SaaS Products** page, on the **SaaS products** tab, select the product that you want to modify.
3. From the **Request changes** dropdown list, select **Update public offer**, and then select **Update refund policy**.
4. The current refund policy details are provided in the text box. Review and modify the details as you want. Submitting the request overwrites the current refund policy.
5. To submit your request for review, choose **Submit**.
6. Verify that the **Requests** tab shows the **Request status** as **Under review**. When the request completes, the status will update to **Succeeded** or **Failed**.

Update the end user license agreement (EULA)

You can update your EULA for new users subscribing to your product.

1. Open the [AWS Marketplace Management Portal](#), and then sign in to your seller account.
2. From the **SaaS Products** tab, select the product that you want to modify.
3. From the **Request changes** dropdown, choose **Update public offer** and then **Update EULA**.
4. You can choose the [Standard Contract for AWS Marketplace \(SCMP\)](#) or submit a custom EULA. For a custom EULA, you must provide an Amazon Simple Storage Service (Amazon S3) URL for the contract.

Note

Your Amazon S3 bucket must be publicly accessible.

5. Choose **Submit** to submit your request for review.
6. In the **Requests** tab, verify that the request status is **Under review**. When the request is complete, the status will change to **Succeeded**.

Integrate your SaaS subscription product

Integrating your product with AWS Marketplace is one step in [Creating a SaaS product](#). To integrate your software as a service (SaaS) subscription product with AWS Marketplace, you must write code and demonstrate that it can respond successfully to several customer scenarios. The following sections describe these scenarios, how to respond to them, and provide an overview of testing your integration.

Note

Before you begin, make sure you've chosen the right pricing model for your software-as-a-service (SaaS) product in AWS Marketplace. For more information, see [Plan your SaaS product](#).

Topics

- [Scenario: Your service validates new customers](#)
- [Scenario: Meter usage](#)
- [Scenario: Monitor changes to user subscriptions](#)
- [Scenario: Verify customer subscription](#)
- [Testing your SaaS subscription product integration](#)

Scenario: Your service validates new customers

When a customer subscribes to your product, they are redirected to your registration URL which is an HTTP POST request with a temporary `x-amzn-marketplace-token` token. Respond to this request in the following ways:

1. Exchange the token for a `CustomerIdentifier`, `CustomerAWSAccountId`, and `ProductCode` by calling the [ResolveCustomer](#) API operation in the AWS Marketplace Metering Service.
2. Persist the `CustomerIdentifier`, `CustomerAWSAccountID`, and `ProductCode` in your system for future calls. You must store whether the customer has a valid subscription, along with whatever information you need about the customer.
3. As a response to the request, you must show your user's first use experience (as applicable for your service).

Scenario: Meter usage

When the customer starts to use your service, you must send metering records hourly. For details on how to meter, see [Metering for usage](#).

We recommend that you use AWS CloudTrail to monitor activity to ensure that billing information is being sent to AWS. Keep the following in mind when sending metering records:

- Metering requests are de-duplicated on the hour.
- Records sent every hour are cumulative.
- We strongly recommend as a best practice that, even if there were no records in the last hour, you send metering records every hour, with usage of 0.

Scenario: Monitor changes to user subscriptions

Set up an Amazon Simple Queue Service (Amazon SQS) queue, and subscribe to your product's Amazon SNS topic. Your SNS topic information was included in the email message that you received from the AWS Marketplace Seller Operations team when you created your product. For more information, see [Creating a SaaS product](#). By subscribing to your SNS topic, you receive notifications about changes to customer subscriptions, including providing or revoking access for specific customers.

Note

An Amazon SNS topic Amazon Resource Name (ARN) looks like `arn:aws:sns:us-east-1:<account id>:aws-mp-subscription-notification-<product code>`.

The notifications that you must respond to are:

- `subscribe-success` – The customer is subscribed, and you can successfully meter against their customer ID.
- `unsubscribe-pending` – The customer is in the process of unsubscribing. You should send any last metering records.
- `unsubscribe-success` – The customer has unsubscribed. Metering records for the customer will no longer be accepted. Follow your practices for shutting down customer resources, adhering to your retention policies.
- `subscribe-fail` – The customer subscription failed. You should not meter against their customer ID or create resources on behalf of the customer.

Scenario: Verify customer subscription

Before creating resources on the customer's behalf, verify that the customer should have access to your product. Store the latest status of the customer from the notifications you receive via Amazon SQS to know if the customer has access.

Testing your SaaS subscription product integration

After you've integrated your SaaS subscription product with AWS Marketplace, you must conduct in-depth testing to ensure that the integration is successful. The following procedure outlines the steps to verify your product integration.


Note

Use your own accounts to subscribe to your product and test that the integration is successful. Prices can be temporarily reduced so that you can test the purchase flow without incurring high charges in those accounts. For more information about temporarily reducing the prices or allowing additional test accounts to access your product, contact the [AWS Marketplace Seller Operations](#) team.

After your product is launched, the service must continue to respond to these scenarios for new customers.

1. Use an allowed account to test the customer experience by subscribing to your product.

2. After you've subscribed with the allowed account, ensure that the account is redirected to the registration URL, and that the redirect is a POST request that includes a temporary token. Make sure that your application persists the customer ID for future calls. This tests part of [Scenario: Your service validates new customers](#).
3. After verifying the test account in the previous step, onboard the account into your application. For example, you can have the test customer fill out a form to create a new user. Or, provide them with other next steps to get access to your SaaS application. This tests part of [Scenario: Your service validates new customers](#).
4. After the test customer is onboarded, make requests that will send metering records to AWS for billing purposes by using the `BatchMeterUsage` API operation in the AWS Marketplace Metering Service. This tests [Scenario: Meter usage](#).
5. Test for subscription changes. Possible scenarios include unsubscribes, successful subscriptions, and failed subscriptions. This tests [Scenario: Monitor changes to user subscriptions](#).
6. Verify a successful subscription. After you receive an Amazon SNS notification for your test account with a successful subscription message, metering can begin. Records that are sent to the AWS Marketplace Metering Service before you receive the Amazon SNS notification aren't metered. This tests [Scenario: Verify customer subscription](#).

 **Note**

To prevent billing issues, we strongly recommend programmatically waiting for this notification before launching resources on behalf of your customers.


7. After you have completed all of the integration requirements and tested the solution, notify the AWS Marketplace Seller Operations team. They will run a series of final tests on the solution by verifying that you have successfully sent metered records with the `BatchMeterUsage` API operation.

After your integration and testing is complete, you can perform a final review and list your product on the public AWS Marketplace. For more information, see [Creating a SaaS product](#).

Integrate your SaaS contract product

Integrating your product with AWS Marketplace is one step in [Creating a SaaS product](#). To integrate your software as a service (SaaS) contract product with AWS Marketplace, you must write code and demonstrate that it can respond successfully to several customer scenarios. The following

sections describe these scenarios, how to respond to them, and provide an overview of testing your integration.

 **Note**

Before you begin, make sure you've chosen the right pricing model for your software as a service (SaaS) product in AWS Marketplace. For more information, see [Plan your SaaS product](#).

Topics

- [Scenario: Your service validates new customers](#)
- [Scenario: Your service handles customer requests](#)
- [Scenario: Monitor changes to user subscriptions](#)
- [Testing your SaaS contract product integration](#)

Scenario: Your service validates new customers

When a customer subscribes to your product, they are redirected to your registration URL, which is an HTTP POST request with a temporary `x-amzn-marketplace-token` token. Respond to this request in the following ways:

1. Exchange the token for a `CustomerIdentifier`, `CustomerAWSAccountId`, and `ProductCode` by calling the [ResolveCustomer](#) API operation in the AWS Marketplace Metering Service.
2. Verify the subscription and quantity (if applicable) the customer has access to by calling the [GetEntitlements](#) API operation in the AWS Marketplace Entitlement Service.
3. Persist the `CustomerIdentifier`, `CustomerAWSAccountId`, and `ProductCode` in your system for future calls. Store whether the customer has a valid subscription, along with whatever information you need about the customer.
4. As a response to the request, you must show your user's first use experience (as applicable for your service).

Scenario: Your service handles customer requests

When a customer makes a request to your service, you must respond to the following scenarios with appropriate actions or messaging:

- They don't have a customer ID in your system. This means that they have not yet subscribed. You should tell the user how to subscribe.
- They have a customer ID, and the `GetEntitlements` API operation returns an appropriate entitlement. In this scenario, you should fulfill the request.
- They do have a customer ID, but the `GetEntitlements` API operation returns no entitlement, or not enough quantity to fulfill the request. In this scenario, you must determine how to handle access and manage their experience.

Scenario: Monitor changes to user subscriptions

Set up an Amazon Simple Queue Service (Amazon SQS) queue, and subscribe to your product's Amazon SNS topic. Your SNS topic information was included in the email message that you received from the AWS Marketplace Operations Team when you created your product. For more information, see [Creating a SaaS product](#). By subscribing to your SNS topic, you receive notifications about changes to customer entitlements, including providing or revoking access for specific customers.

Note

An SNS topic Amazon Resource Name (ARN) looks like `arn:aws:sns:us-east-1:<account id>:aws-mp-entitlement-notification-<product code>`.

The only notification that you must respond to is:

- `entitlement-updated` – The customer entitlement has changed, and you must call the `GetEntitlements` API operation to see the new status. Update your customer store, and, if applicable (for example, the customer's contract has lapsed), follow your practices for shutting down customer resources, adhering to your retention policies.

Note

For additional information, see [Checking entitlements](#).

Testing your SaaS contract product integration

After you've integrated your SaaS contract product with AWS Marketplace, you must conduct in-depth testing to ensure that the integration is successful. The following procedure outlines the steps to verify your product integration.

Note

Use your own accounts to subscribe to your product and test that the integration is successful. Prices can be temporarily reduced so that you can test the purchase flow without incurring high charges in those accounts. For more information about temporarily reducing the prices or allowing additional test accounts to access your product, [contact us](#). After your product is launched, the service must continue to respond to these scenarios for new customers.

1. Use an allowed account to test the customer experience by getting a contract for your product.
2. After the account has the contract, ensure that the account is redirected to the registration URL, and that the redirect is a POST request that includes a temporary token. Make sure that your application persists the customer ID for future calls and correctly handles the entitlement the customer has. This tests part of [Scenario: Your service validates new customers](#).
3. After verifying the test account in the previous step, onboard the account into your application. For example, you can have the test customer fill out a form to create a new user. Or, provide them with other next steps to get access to your SaaS application. This tests part of [Scenario: Your service validates new customers](#).
4. If no entitlement is returned from the GetEntitlements API operation, either during onboarding or in your ongoing verification passes, your application must correctly manage access and the experience for users who are not entitled. This tests [Scenario: Your service handles customer requests](#).
5. Test for subscription changes. Verify that your application correctly handles unsubscribes, successful subscription, and failed subscription scenarios. This tests [Scenario: Monitor changes to user subscriptions](#).

6. After you have completed all the integration requirements and tested the solution, notify the AWS Marketplace Operations team. They will then test the solution by verifying that you have successfully called the `GetEntitlements` API operation and sufficiently onboarded new customers.

After your integration and testing is complete, you can perform a final review and list your product on the public AWS Marketplace. For more information, see [Creating a SaaS product](#). You can also cancel your test subscription by completing a Refund Request Form. For more information on cancelling a subscription, see the [the section called "AWS Marketplace product refund process"](#).

Integrate your SaaS contract with pay-as-you-go product

Integrating your product with AWS Marketplace is one step in [Creating a SaaS product](#). To integrate your software as a service (SaaS) contract product with AWS Marketplace, you must write code and demonstrate that it can respond successfully to several customer scenarios. The following sections describe these scenarios, how to respond to them, and provide an overview of testing your integration.

Note

Before you begin, make sure you've chosen the right pricing model for your software as a service (SaaS) product in AWS Marketplace. For more information, see [Plan your SaaS product](#).

Topics

- [Scenario: Your service validates new customers](#)
- [Scenario: Your service handles customer requests](#)
- [Scenario: Meter usage](#)
- [Scenario: Monitor changes to user entitlements](#)
- [Testing your SaaS contract product integration](#)

Scenario: Your service validates new customers

When a customer subscribes to your product, they are redirected to your registration URL, which is an HTTP POST request with a temporary `x-amzn-marketplace-token` token. Respond to this request in the following ways:

1. Exchange the token for a `CustomerId`, `CustomerAWSAccountId`, and `ProductCode` by calling the [ResolveCustomer](#) API operation in the AWS Marketplace Metering Service.
2. Verify the subscription and quantity (if applicable) the customer has access to by calling the [GetEntitlements](#) action in the AWS Marketplace Entitlement Service.
3. Persist the `CustomerId`, `CustomerAWSAccountId`, and `ProductCode` in your system for future calls. Store whether the customer has a valid subscription, along with whatever information you need about the customer.
4. As a response to the request, you must show your user's first use experience (as applicable for your service).

Scenario: Your service handles customer requests

When a customer makes a request to your service, you must respond to the following scenarios with appropriate actions or messaging:

- They don't have a customer ID in your system. This means that they have not yet subscribed. You should give them messaging describing how to subscribe.
- They have a customer ID, and the `GetEntitlements` API operation returns an appropriate entitlement. In this scenario, you should fulfill the request.
- They do have a customer ID, but the `GetEntitlements` API operation returns no entitlement, or not enough quantity to fulfill the request. In this scenario, you must determine how to handle access and manage their experience.

Scenario: Meter usage

When the customer starts to use your service, you must send metering records hourly. For details on how to meter, see [Metering for usage](#).

We recommend that you use AWS CloudTrail to monitor activity to ensure that billing information is being sent to AWS. Keep the following in mind when sending metering records:

- Metering requests are de-duplicated on the hour.
- Records sent every hour are cumulative.
- We strongly recommend as a best practice that, even if there were no records in the last hour, you send metering records every hour, with usage of 0.

Scenario: Monitor changes to user entitlements

Set up an Amazon Simple Queue Service (Amazon SQS) queue, and subscribe to your product's Amazon SNS topics—there are two SNS topics, one for entitlement changes and one for subscription changes. Your topic information was included in the email message that you received from the AWS Marketplace Seller Operations team when you created your product. For more information, see [Creating a SaaS product](#). By subscribing to your SNS topics, you receive notifications about changes to customer subscriptions, including providing or revoking access for specific customers.

Note

An SNS topic Amazon Resource Name (ARN) for a subscription change looks like `arn:aws:sns:us-east-1:<account id>:aws-mp-subscription-notification-<product code>`. An SNS topic ARN for entitlement changes looks like `arn:aws:sns:us-east-1:<account id>:aws-mp-entitlement-notification-<product code>`.

The notifications that you must respond to are as follows:

- `entitlement-updated` (in the entitlement SNS topic)– The customer entitlement has changed, and you must call the `GetEntitlements` API operation to see the new status. Update your customer store, and, if applicable (for example, the customer's contract has lapsed), follow your practices for shutting down customer resources, adhering to your retention policies.
- `subscribe-success` (in the subscription SNS topic) – The customer is subscribed, and you can successfully meter against their customer ID.
- `unsubscribe-pending` (in the subscription SNS topic) – The customer is in the process of unsubscribing. You should send any last metering records.

- `unsubscribe-success` (in the subscription SNS topic) – The customer has unsubscribed. Metering records for the customer will no longer be accepted. Follow your practices for shutting down customer resources, adhering to your retention policies.
- `subscribe-fail` (in the subscription SNS topic) – The customer subscription failed. You should not meter against their customer ID or enable resources on behalf of the customer.

Note

For additional information, see [Checking entitlements](#).

Testing your SaaS contract product integration

After you've integrated your contract with pay-as-you-go product with AWS Marketplace, you must conduct in-depth testing to ensure that the integration is successful. The following procedure outlines the steps to verify your product integration.

Note

Use your own accounts to subscribe to your product and test that the integration is successful. Prices can be temporarily reduced so that you can test the purchase flow without incurring high charges in those accounts. For more information about temporarily reducing the prices or allowing additional test accounts to access your product, contact the [AWS Marketplace Seller Operations](#) team.

After your product is launched, the service must continue to respond to these scenarios for new customers.

1. Use an allowed account to test the customer experience by getting a contract for your product.
2. After the account has the contract, ensure that the account is redirected to the registration URL, and that the redirect is a POST request that includes a temporary token. Make sure that your application persists the customer ID for future calls and correctly handles the entitlement the customer has. This tests part of [Scenario: Your service validates new customers](#).
3. After verifying the test account in the previous step, onboard the account into your application. For example, you can have the test customer fill out a form to create a new user. Or, provide them with other next steps to get access to your SaaS application. This tests part of [Scenario: Your service validates new customers](#).

4. If no entitlement is returned from the `GetEntitlements` API operation, either during onboarding or in your ongoing verification passes, your application must correctly manage access and the experience for users who are not entitled. This tests [Scenario: Your service handles customer requests](#).
5. After the test customer is onboarded, make requests that will send metering records to AWS for billing purposes by using the `BatchMeterUsage` API operation in the AWS Marketplace Metering Service. This tests [Scenario: Meter usage](#).
6. Test for subscription changes. Verify that your application correctly handles unsubscribes, successful subscription, and failed subscription scenarios. This tests [Scenario: Monitor changes to user entitlements](#).
7. After you have completed all the integration requirements and tested the solution, notify the AWS Marketplace Seller Operations team. They will then test the solution by verifying that you have successfully called the `GetEntitlements` API operation and sufficiently onboarded new customers. They will also verify that you have successfully sent metered records with the `BatchMeterUsage` API operation.

After your integration and testing is complete, you can perform a final review and list your product on the public AWS Marketplace. For more information, see [Creating a SaaS product](#).

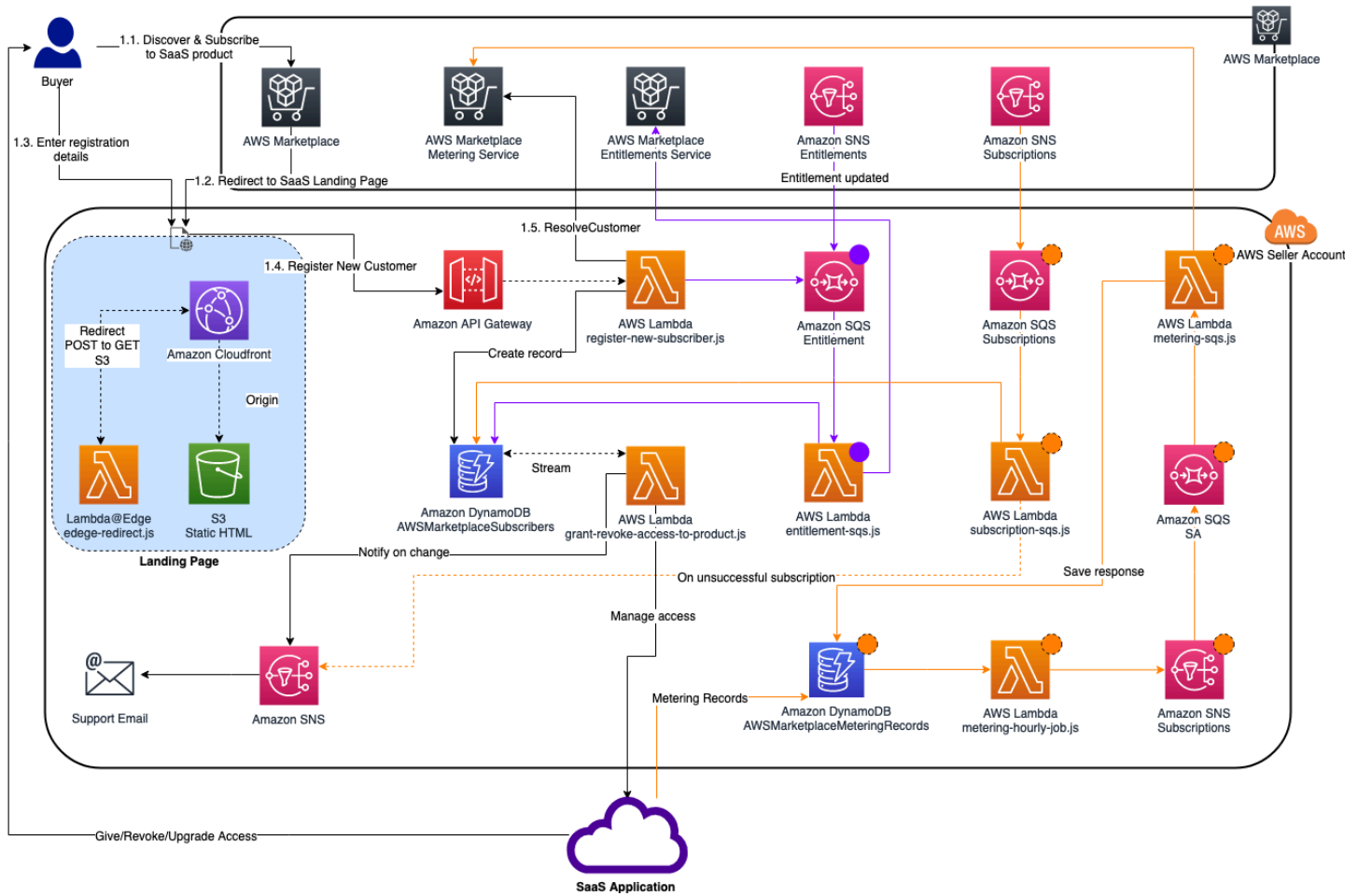
Deploy a serverless SaaS integration solution

The AWS Marketplace serverless SaaS integration deployment fulfills the core capabilities required to successfully integrate a vendor's SaaS solution with its corresponding listing on AWS Marketplace. These capabilities include accepting new customer registrations, granting and revoking customer access, updating customer entitlements, and reporting metered usage.

The video featured here explains how to deploy the AWS Quick Start for AWS Marketplace serverless SaaS integration. With this automated reference deployment, you can easily integrate new software as a service (SaaS) solutions on your AWS Marketplace seller account, accelerate the listing process, and significantly reduce go-to-market time.

[AWS Quick Start for AWS Marketplace Serverless SaaS Integration](#)

The following figure shows how the AWS Marketplace serverless SaaS integration on AWS environment sets up the following workflow of events.



For more information about how to deploy a serverless SaaS integration on the AWS Cloud, reference the [AWS Marketplace Serverless SaaS Integration Quick Start Reference Deployment Guide](#). This *Quick Start Reference Guide* is for registered AWS Marketplace sellers who want a lightweight serverless solution for completing the required integration on new SaaS listings.

Plan your SaaS product

Before you add your software as a service (SaaS) product to AWS Marketplace, you must first do some planning. This step is critical to the success of your product. A lack of planning can result in billing issues or you might have to re-create your product in AWS Marketplace.

⚠ Important

Most of your product's settings can't be changed after you've configured them. If you need to change them after the product is created in AWS Marketplace, you probably need to create a new product with the correct settings.

Plan your pricing

There are three pricing models for SaaS products on AWS Marketplace. Choosing the right pricing model for your product is the most important decision you'll make as you plan your product. Choosing the wrong pricing model can set you back by weeks. The pricing model determines the payment options for your customers and the billing integration code that you need to write, test, and deploy. For information about the different types of pricing models, see [SaaS product pricing](#).

Note

All SaaS pricing models support free trials. For more information, see [SaaS free trials](#).

Plan your billing integration

One of the benefits of having a SaaS product on AWS Marketplace is consolidating billing. In order to take advantage of this benefit, you must integrate with the AWS Marketplace Metering Service or the AWS Marketplace Entitlement Service, depending on your chosen pricing model. These two services help you ensure that your billing and usage reporting is accurate.

After you plan your integration, you must test the integration with your product before it goes live. For more information about integration and testing, see [Accessing the AWS Marketplace Metering and Entitlement Service APIs](#).

Plan your Amazon SNS integration

There are two Amazon Simple Notification Service (Amazon SNS) topics that you can subscribe to for your SaaS product. For more information, see [SaaS notifications](#). These messages can help you programmatically handle changes to subscriptions and contracts initiated by AWS or by your customers. Amazon SNS notifications can be programmatic triggers enabling customers to register for a new account on your product registration website. They can also deny customers with expired subscriptions from accessing your product. You have options for how your customers receive notifications depending on how you program the handling of these notifications.

Plan how customers will access your product

This section describes how to make your product accessible to buyers.

Plan your SaaS product registration website

Customers who buy your SaaS product need access to it. You must plan and implement how you want your customers to access the product. SaaS products support the following access options:

- Quick Launch
- AWS PrivateLink
- Your own product website

To validate AWS Marketplace customers using your registration website, see [SaaS customer onboarding](#).

Using Quick Launch for customers to access your product

Use the Quick Launch deployment option to reduce the time and resources that are required for buyers to configure, deploy, and launch your products. Quick Launch reduces the number of sites that buyers must visit during the process. For more information, see [Configure Quick Launch](#).

Using AWS PrivateLink for customers to access your SaaS product

You can use [Using AWS PrivateLink with AWS Marketplace](#) to configure your service as an Amazon Virtual Private Cloud (Amazon VPC) endpoint service. Your customers can create a VPC endpoint and access your software across the AWS Cloud virtual network. Alternatively, you can provide access to your software product through a website you own and maintain, with customers creating a connection across the internet.

Using your own website

Your SaaS product is hosted in your environment and it must be accessed over the internet through a public endpoint that you manage and maintain, like a website. Typically, you have a website that customers use to register for your product, sign in to use the product, and access support for your product.

SaaS product guidelines

AWS Marketplace maintains these guidelines for all SaaS products and offerings on AWS Marketplace to promote a safe, secure, and trustworthy platform for our customers.

All products and their related metadata are reviewed when submitted to ensure that they meet or exceed current AWS Marketplace guidelines. These guidelines are reviewed and adjusted to meet

our evolving security requirements. In addition, AWS Marketplace continuously reviews products to verify that they meet any changes to these guidelines. If products fall out of compliance, we might require that you update your product and in some cases your product might temporarily be unavailable to new subscribers until issues are resolved.

Product setup guidelines

All SaaS products must adhere to the following product setup guidelines:

- At least one pricing dimension must have a price greater than \$0.00.
- All pricing dimensions must relate to actual software and cannot include any other products or services unrelated to the software.
- SaaS products offered exclusively in the AWS GovCloud (US) Regions must include GovCloud somewhere in the product title.

Customer information requirements

All SaaS products must adhere to the following customer information requirements:

- SaaS products must be billed entirely through the listed dimensions on AWS Marketplace.
- You cannot collect customer payment information for your SaaS product at any time, including credit card and bank account information.

Product usage guidelines

All SaaS products must adhere to the following product usage guidelines:

- After subscribing to the product in AWS Marketplace, customers should be able to create an account within your SaaS application and gain access to a web console. If the customer cannot gain access to the application immediately, you must provide a message with specific instructions on when they will gain access. When an account has been created, the customer must be sent a notification confirming that their account has been created along with clear next steps.
- If a customer already has an account in the SaaS application, they must have the ability to log in from the fulfillment landing page.
- Customers must be able to see the status of their subscription within the SaaS application, including any relevant contract or subscription usage information.

- Customers must be able to easily get help with issues such as: using the application, troubleshooting, and requesting refunds (if applicable). Support contact options must be specified on the fulfillment landing page.
- Product software and metadata must not contain language that redirects users to other cloud platforms, additional products, upsell services, or free trial offers that aren't available on AWS Marketplace.

For information about free trials for SaaS products, see [SaaS free trials](#).

- If your product is an add-on to another product or another ISV's product, your product description must indicate that it extends the functionality of the other product and that without it, your product has very limited utility. For example, *This product extends the functionality of <product name> and without it, this product has very limited utility. Please note that <product name> might require its own license for full functionality with this listing.*

Architecture guidelines

All SaaS products must adhere to the following architecture guidelines:

- A portion of your application must be hosted in an AWS account that you own.
- All application components should be hosted in infrastructure you manage. Applications that require additional resources in the customer's infrastructure must follow these guidelines:
 - Provision resources in a secure way, such as using the AWS Security Token Service (AWS STS) or AWS Identity and Access Management (IAM).
 - Provide additional documentation including a description of all provisioned AWS services, IAM policy statements, and how an IAM role or user is deployed and used in the customer's account.
 - Include a notification in the product description that explains that if the customer incurs additional AWS infrastructure charges separate from their AWS Marketplace transaction, they're responsible for paying the additional infrastructure charges.
 - If your product deploys an agent, you must provide instructions to the customer that describe how to deploy it in their AWS account.
 - Applications that require resources running in the customer's infrastructure will have an additional review by AWS Marketplace, which can take 2-4 weeks.

- Successfully call the AWS Marketplace APIs from the AWS account that registered as a provider and submitted the SaaS publishing request. The SaaS pricing model determines which APIs should be called:
 - SaaS contracts – [GetEntitlements](#) in the AWS Marketplace Entitlement Service.
 - SaaS contracts with consumption – [GetEntitlements](#) in the AWS Marketplace Entitlement Service and [BatchMeterUsage](#) in the AWS Marketplace Metering Service.
 - SaaS subscriptions – [BatchMeterUsage](#) in the AWS Marketplace Metering Service.
- SaaS products offered exclusively in the AWS GovCloud (US) Regions must outline the architectural boundaries between other AWS Regions and the AWS GovCloud (US) Regions, use cases for the product, and the workloads not recommended for the product.

SaaS product pricing

After a buyer purchases your software as a service (SaaS) product on AWS Marketplace, AWS Marketplace provides you with their billing identifier. You use the billing identifier to call the AWS Marketplace Entitlement Service and the AWS Marketplace Metering Service. Then, customers access the product in your AWS environment or through a virtual private cloud (VPC) endpoint connection that you create.

Note

All SaaS pricing models support free trials. For more information, see [SaaS free trials](#).

SaaS pricing models

Pricing model	Description
SaaS subscriptions	A pay-as-you-go model where we bill buyers for their hourly usage of your SaaS product. For more information, see Pricing for SaaS subscriptions .
SaaS contracts	Buyers are either billed in advance for the use of your software, or you can offer them a flexible payment schedule. Customers can also pay for additional usage above their contract.

Pricing model	Description
SaaS contracts with pay-as-you-go	<p>For more information, see Pricing for SaaS contracts.</p> <p>Buyers are either billed in advance for the use of your software, or you can offer them a flexible payment schedule. Buyers are also billed an additional metered rate for usage on top of the contract price. For more information, see Pricing for SaaS contracts.</p>

To make your SaaS product available on AWS Marketplace, decide whether you want to offer the [SaaS subscriptions pricing model](#) or the [SaaS contracts pricing model](#).

Pricing for SaaS subscriptions

For software as a service (SaaS) subscriptions, AWS Marketplace bills your customers based on the metering records that you send to us. All charges must be measured and reported every hour from the software deployed in the customer's account. All usage is then calculated monthly and billed monthly using the same mechanism as AMI based AWS Marketplace offerings. Our ability to bill customers for usage of your product is dependent upon receiving metering records from you. You are responsible for ensuring that your product's metering records are successfully transmitted and received.

Before you can publish a SaaS product with subscription pricing, you must do the following:

1. Create a new SaaS product in the AWS Marketplace Management Portal, choose **New SaaS Subscription**.
2. Complete the fields in the **General** tab with the necessary information. Make a note of the product code.
3. On the **Pricing** tab, under **Set Pricing**, select the **Category** that describes your product's pricing most accurately. The pricing category appears to customers on the AWS Marketplace website. You can choose from **Bandwidth** (GBps, MBps), **Data** (GB, MB, TB), **Hosts** (hours), **Requests**, **Tiers** (hours), or **Users** (hours). If none of the predefined categories fit your needs, you can choose the more generic **Units** category.

Next, define your Pricing Dimensions. Each Pricing Dimension represents a feature or service that you can set a per-unit price for. Examples of dimensions include users, hosts scanned, and GB of logs ingested. You can define up to 24 dimensions. For each dimension you define, you must add the following information:

- **Dimension API Name** – The API name used when sending metering records to the [AWS Marketplace Metering Service](#). This name indicates which dimension your customer used. This name is visible in billing reports. The name doesn't need to be reader-friendly because you're the only one with access to your reports. After you set the name, you can't change it.
- **Dimension Description** – The customer-facing statement that describes the dimension for the product. The description can be no more than 70 characters and should be user-friendly. Examples of descriptions are Administrators per hour, and Per Mbps bandwidth provisioned. After the product is published, you can change this description.
- **Dimension Rate** – The software charge per FCP unit for this product, in USD. This field supports three decimal places.

When a SaaS subscription ends

A customer can unsubscribe from your SaaS subscription product through the AWS Management Console. Key points of the SaaS subscription ending process include the following:

1. Your SaaS product is sent an `unsubscribe-pending` notification through the Amazon SNS topic for that customer.
2. You have one hour to meter any remaining usage for the customer.
3. After this hour, you receive an `unsubscribe-success` notification. At this point, you can no longer send metering records for this customer.

It's up to you to decide how you want to disable functionality in your SaaS product for unsubscribed customers. For example, your product might complete the customer's existing work but prevent them from creating work. You might want to display a message to the customer that their usage has been disabled. Customers can resubscribe to your product through AWS Marketplace.

When a SaaS subscription is cancelled

Key points of the SaaS subscription cancellation process include the following:

1. A customer can cancel their subscription to your SaaS subscription product the **Your Marketplace Software** page of the AWS Marketplace website.

Your SaaS product is sent notification through the Amazon SNS topic for that customer.

2. You have one hour to meter any remaining usage for the customer.
3. You notify the customer from your product that the cancellation is in progress. If a customer indicates that they want to cancel through your product, direct the customer to AWS Marketplace. To guarantee that there will be no future charges, customers should confirm the cancellation with AWS Marketplace.

Pricing for SaaS contracts

For software as a service (SaaS) contracts, the customer initiates a purchase of your software and enters into an agreement with you. Under the agreement, the customer is entitled to a specified quantity of use of your SaaS product. AWS Marketplace communicates these entitlements to your SaaS application. This is done through the AWS Marketplace Entitlement Service. When using the SaaS Contract pricing model, your application never sends metering records. Instead, it verifies entitlement by calling the AWS Marketplace Entitlement Service. You define the usage categories, dimensions, and the length of the contract.

AWS Marketplace bills your customers upfront or by the payment schedule that you define, based on the contract between you and your customer. After that point, they're entitled to use those resources. For additional usage above their contract, your software needs to report the usage and AWS Marketplace bills your customers based on the metering records received by us through the AWS Marketplace Metering Service.

Before you can publish a SaaS product with contract pricing, you must do the following:

1. Create a new SaaS product in the AWS Marketplace Management Portal, and choose **New SaaS Contract**.
2. Complete the fields in the **General** tab with the necessary information. Make a note of the product code.
3. On the **Pricing** tab:
 - a. For **Set Pricing**, choose the **Contract Duration** you want offer customers. You can enter different prices for each contract duration. You can select one or more of the following options: **Monthly**, **1 year**, **2 Years**, and **3 Years**. If you are creating a private offer, you can choose a custom duration in months (up to 60 months).

- b. For **Choose the contract type you want to offer**, choose how you want customers to be able to purchase your product from the following options:
 - **Buyer can choose one or more options offered** – Customers can select a quantity for each pricing dimension you offer.
 - **Buyer can choose one tier from multiple tiers offered** – Customers choose a tier from options that include different sets of features, services, and usage amounts.
 - c. Choose the usage unit category that describes your product's pricing most accurately. The pricing category appears to customers on the AWS Marketplace website. You can choose from **Bandwidth** (GBps, MBps), **Data** (GB, MB, TB), **Hosts** (hours), **Requests**, **Tiers** (hours), or **Users** (hours). If none of the predefined categories fit your needs, you can choose the more generic **Units** category.
4. After you choose a category, define your Pricing Dimensions. Each Pricing Dimension represents a feature or service that you can set a per unit price for. Examples of dimensions are users, hosts scanned, and GB of logs ingested. For each dimension you define, you add a name, a description, a price, and an API name. The name, price, and description are displayed to customers. You use the API name for tracking and reporting with AWS Marketplace as follows:
- Calling the [AWS Marketplace Entitlement Service](#) to retrieve the dimensions your customers have purchased.
 - Calling the [AWS Marketplace Metering Service](#) to indicate which dimensions customers used.

For each pricing dimension in your contract, you can choose to let customers pay as they go for additional usage of that dimension above their contract. You can also add additional dimensions without contract prices that customers only consume by paying as they go.

When using the wizard to create the contracts for your SaaS product, you must define the following fields for your pricing dimensions:

- **Dimension API Name** – The name used when calling the Entitlements API. This name is visible in billing reports and reports that aren't external-facing. The maximum length for the API name is 15 characters. After you set the name, it can't be changed.
- **Dimension Display Name:** – The customer-facing name of a dimension. This name should help the customer understand the dimension for the product. The name should be user-friendly, and its maximum length is 24 characters. This value can be changed.
- **Dimension Description:** – The customer-facing description of a dimension that provides additional information about the dimension for the product. The maximum length for the description is 70 characters.

- **Dimension - Monthly Price** – The software charge per unit for the 1-month option for this dimension. This field supports three decimal places.
- **Dimension - 1 Year Price** – The software charge per unit for the 12-month option for this dimension. This field supports three decimal places. It's not a monthly charge. The price must reflect the 12-month one-time charge price.
- **Dimension - 2 Years Price** – The software charge per unit for the 24-month option for this dimension. This field supports three decimal places.
- **Dimension - 3 Years Price** – The software charge per unit for the 36-month option for this dimension. This field supports three decimal places.

Example: Data storage application

	Monthly price	12-month price	24-month price	Pay-as-you-go price for additional usage
Unencrypted data (GB)	\$1.50/GB	\$16.00/GB	\$30.00/GB	\$0.1/GB per hour
Encrypted data (GB)	\$1.55/GB	\$16.60/GB	\$31.20/GB	\$0.11/GB per hour

Example: Log monitoring product

	Monthly price	12-month price	Pay-as-you-go price for additional usage
Basic (10 hosts monitored, 5 containers monitored)	\$100	\$1000	
Standard (20 hosts monitored, 10 containers monitored)	\$200	\$2000	

	Monthly price	12-month price	Pay-as-you-go price for additional usage
containers monitored)			
Pro (40 hosts monitored, 20 containers monitored)	\$400	\$4000	
Additional hosts monitored per hour			\$0.1
Additional containers monitored per hour			\$0.2

Note

The prices can be for the following durations: 1 month, 12 months, 24 months, or 36 months. You can choose to offer one or more of these options for your product. The durations must be the same across each dimension. For example, assume that you have `ReadOnlyUsers` and `AdminUsers` dimensions. If you offer a yearly price for `ReadOnlyUsers`, you must offer a yearly price for `AdminUsers`, too.

SaaS contract upgrades

Customers can upgrade a contract to one of a higher value except for longer durations. For example, they can upgrade to higher quantities or higher-value entitlements. Customers are given a prorated credit for their existing contract. Customers can't decrease the size of their existing contract. They can only decrease the size at renewal, or cancel their renewal.

Entitlements are verified by your SaaS product, which makes calls to the AWS Marketplace Entitlement Service.

Automatic renewals

When a customer purchases your product through AWS Marketplace using SaaS contracts, they can agree to automatic renewal of the contract terms. The customer continues to pay for the entitlements every month or for 1, 2, or 3 years. The customer always has the option to modify the renewal settings. They can cancel the renewal or renew the contract for different quantities and durations.

When a SaaS contract ends

A SaaS contract product has a contract expiry. When a contract ends, the following events occur:

1. Your SaaS product receives an entitlement-updated notification indicating the buyer's entitlement has changed. The AWS Marketplace Entitlement Service returns an empty response.
2. You have 1 hour to meter any remaining usage for the customer. After this time has elapsed, you can no longer send metering records for this customer.

When a SaaS contract is canceled

Key points of the SaaS contract cancellation process include the following:

1. Customers can request a cancellation and refund for SaaS contract products through AWS Support.

Customers must request refunds within 48 hours through AWS Support.

The full or prorated refund is typically granted in 3–5 business days.

2. Your SaaS product is sent notification through the Amazon SNS topic for that customer.
3. You have one hour to send a final metering record for the customer for any additional usage charges.
4. You notify the customer from your product that the cancellation is in progress. If a customer indicates that they want to cancel through your product, direct the customer to AWS Marketplace. To guarantee that there will be no future charges, customers should confirm the cancellation with AWS Marketplace.

SaaS free trials

Sellers can create software as a service (SaaS) free trial offers in the AWS Marketplace Management Portal (AMMP). Customers can evaluate software products before making large purchase decisions by using the SaaS free trial option. After a customer subscribes to your product, your product performs entitlement checks the same way it does for paid customers.

Each AWS account can only use a free trial for a SaaS product once. The free usage amount granted during a free trial is not shared across linked accounts in an AWS organization. Different linked accounts within a single main payer account can create their own individual free trials.

Note

If you use Seller Data Delivery Service (SDDS), you'll receive an [Agreement details trial report](#) in your Amazon Simple Storage Service (Amazon S3) bucket. The report includes agreement details such as the subscriber name and ID, offer ID, and agreement start and end dates. As a seller, you'll also receive [Amazon Simple Notification Service \(Amazon SNS\) notifications](#) when new subscriptions are created. Amazon SNS notifications include an `isFreeTrialTermPresent` flag to identify free trial agreements.

Creating a SaaS free trial offer

Sellers can create SaaS free trial offers in the AWS Marketplace Management Portal (AMMP).

To create a SaaS free trial offer

1. Sign in to the [AWS Marketplace Management Portal](#).
2. On the AWS Marketplace Management Portal, choose either:
 - **Create or manage offers**
 - The **Offers** tab
3. On the **Offers** page, choose the **Public free trials** tab to review all SaaS free trials.
4. Choose **Create free trial offer**. Sellers can create one SaaS free trial offer per each public SaaS product.
5. For **Offer fundamentals**, select your **Product** and then choose **Next**.
6. In **Free trial settings**:

- a. Enter the number of days for your **Free trial length (days)**.

The duration of free trials range from 7–90 days.

- b. View the **Product dimensions** from your existing public offer.

You can't change the product dimensions for SaaS subscription free trials.

You can set the quantity limits per each dimension for SaaS contract free trials, and **Remove** or **Add dimensions**.

7. View the **Service agreement**.

For the EULA version, you can select either **Standard contract for AWS Marketplace** or **Custom EULA**, and then choose **Review offer**.

8. Verify and review all information for the offer, and then choose **Create offer**.

Canceling a SaaS free trial offer

Sellers can cancel free trial offers at any time from the AWS Marketplace Management Portal.

To cancel a SaaS free trial offer

1. Sign in to the [AWS Marketplace Management Portal](#).
2. On the AWS Marketplace Management Portal, choose either:
 - **Create or manage offers**
 - The **Offers** tab
3. On the **Offers** page, select the offer.
4. Choose **View offer**.
5. Choose **Cancel offer**.

After an offer is canceled, active agreements for this offer are active until expiration. New agreements for a canceled offer can't be created.

SaaS customer onboarding

With software as a service (SaaS) subscriptions and SaaS contracts, your customers subscribe to your products through AWS Marketplace but access the product in your AWS environment. After subscribing to the product, your customer is directed to a website you create and manage as a part of your SaaS product to register their account and configure the product.

When creating your SaaS product listing, you provide a URL to your registration landing page. We use that URL to redirect customers to your registration landing page after they subscribe. On your software's registration landing page, you collect whatever information is required to create an account for the customer. We recommend collecting your customer's email addresses if you plan to contact them through email for usage notifications.

The registration landing page must be able to identify and accept the `x-amzn-marketplace-token` token in the form data from AWS Marketplace with the customer's identifier for billing. It should then pass that token value to the AWS Marketplace Metering Service to resolve for the unique customer identifier, customer AWS account Id, and corresponding product code. For a code example, see [ResolveCustomer code example](#).

Note


The registration token resolves to a specific subscribed customer and each generated token has an expiration window of 4 hours. As long as the caller is calling the API with the same token, it will keep returning the same response values until the token expires.

Configuring your SaaS product to accept new buyers

You're responsible for correctly configuring your SaaS software to accept new customers and meter them appropriately. The following process outlines one recommended way of identifying, implementing, and metering a new customer's access to your software:


1. When a customer visits your product page on the AWS Marketplace website, they choose to subscribe to your product.
2. The customer's AWS account is subscribed to your product. This means subscription and metering records sent from your product become part of the customer's AWS bill.
3. A registration token is generated for the customer that contains their customer identifier and your product code.

4. The customer is redirected to your software's registration landing page. This page must be able to accept the token with the customer's identifier.
5. The customer's browser sends a POST request to your software's registration landing page URL. The request contains one POST parameter, `x-amzn-marketplace-token`, containing the customer's registration token. From the perspective of your registration website, the customer has submitted a form with this parameter. The registration token is an opaque string. If the offer type is a free trial, a second parameter, `x-amzn-marketplace-offer-type` with the value `free-trial`, will be added to the request.
6. To redeem this registration token for a customer identifier, customer AWS account Id, and product code, your website must call [ResolveCustomer](#) on the AWS Marketplace Metering Service. The customer identifier isn't the customer's AWS account ID, but it's universal between products and should be saved to an internal source as part of your customer records. The product code is a unique string for your SaaS product that AWS provides to you. Each AWS product has one unique product code, which is assigned to you during registration.

 **Note**

To see an example of a `ResolveCustomer` call, see [ResolveCustomer code example](#).

7. The customer is instructed to either create an account in your product or sign in to an existing account.

 **Note**

If setting up or linking to an existing customer account in your product requires a manual process by your team, you can use a contact-us form to collect the customer's contact information. After collecting their contact information and resolving their AWS Marketplace unique customer identifier (as obtained in step 6), display a notification message for the customer. In the notification, state that their account is being set up and ask that they wait for you to contact them. Provide the customer with the expected turnaround time and your contact information. Also send an email message to the customer with the same details.

8. The customer is now signed in to your website using credentials specific to that SaaS product. In your accounts database, you can have an entry for each customer. Your accounts database must have a column for the AWS customer identifier, which you populate with the customer identifier that you obtained in step 6. Verify that no other accounts in your system share

this customer identifier. For customers who subscribe to multiple products through AWS Marketplace, the customer identifier will remain the same, with each subscription having a unique product code.

9. During your seller registration process, you subscribe to Amazon SNS topics that notify you when customers subscribe or unsubscribe to your product. These are Amazon SNS notifications in JSON format that inform you of customer actions:
 - Entitlement notification – For products with pricing models that include a contract, you are notified when buyers create a new contract, upgrade it, renew it, or it expires. Your accounts database must have an extra column for the subscription state. For more information, see [Amazon SNS topic: aws-mp-entitlement-notification](#).
 - Subscription notification – For products with any pricing model, including contracts and subscriptions, you are notified when a buyer subscribes or unsubscribes to a product. For more information, see [Amazon SNS topic: aws-mp-subscription-notification](#).

We recommend that you use Amazon Simple Queue Service (Amazon SQS) to capture these messages. After you receive a subscription notification with `subscribe-success`, the customer account is ready for metering. Records that you send before this notification aren't metered. For information about how to do this, see [Step 2: Give permission to the Amazon SNS topic to send messages to the Amazon SQS queue](#) in the *Amazon Simple Notification Service Developer Guide*.

Note

Do not activate a product subscription unless you receive a `subscribe-success` notification.

10. Use the customer identifier stored in your database to meter for usage through the AWS Marketplace Metering Service or check for entitlements through the AWS Marketplace Entitlement Service.

Security and ordering

As a seller, it's your responsibility to trust only customer identifiers that are immediately returned from AWS or those that your system has signed. We recommend that you resolve the registration token immediately because it may expire after approximately 1 hour. After you resolve the

registration token, store the customer identifier as a signed attribute on the customer's browser session until the registration is complete.

Amazon SNS notifications for SaaS products

To receive notifications, you subscribe to the AWS Marketplace Amazon Simple Notification Service (Amazon SNS) topics provided to you during product creation. The topics provide notifications about changes to customers' subscriptions and contract entitlements for your products. This enables you to know when to provide and revoke access for specific customers.

Note

During the product creation process, you'll receive the actual Amazon Resource Name (ARN) to the SNS topic. For example: `arn:aws:sns:us-east-1:123456789012:aws-mp-subscription-notification-PRODUCTCODE`

The following Amazon SNS topics are available to software as a service (SaaS) products:

- [Amazon SNS topic: `aws-mp-entitlement-notification`](#) – This topic notifies you when buyers create a new contract, upgrade it, renew it, or it expires. This is only available for products with pricing models that include a contract (also known as **SaaS Contracts** and **SaaS Contracts with Consumption (Overages)**).
- [Amazon SNS topic: `aws-mp-subscription-notification`](#) – This topic notifies you when a buyer subscribes to or unsubscribes from a product and includes the `offer-identifier` for private offers and a free trials flag for SaaS free trials. This is available for all pricing models, including contracts and subscriptions (also known as **SaaS Subscriptions**, **SaaS Contracts**, and **SaaS Contracts with Consumption (Overages)**).

To learn more about the scenarios in which you respond to these notifications, see the following topics:

- [Integrate your SaaS subscription product](#)
- [Integrate your SaaS contract product](#)
- [Integrate your SaaS contract with pay-as-you-go product](#)

Amazon SNS topic: aws-mp-entitlement-notification

Each message in the aws-mp-entitlement-notification topic has the following format.

```
{
  "action": "<action-name>",
  "customer-identifier": " X01EXAMPLEX",
  "product-code": "n0123EXAMPLEXXXXXXXXXXXXX",
}
```

The *<action-name>* will always be entitlement-updated.

Note

- For entitlement messages, regardless of the action (new, upgrade, renewal, or expired), the message is the same. A subsequent call to GetEntitlement is required to discover the content of the update.
- For **SaaS Contract with Consumption (Overages)**, sellers are provided with the [aws-mp-subscription-notification SNS topic](#). This is an extra notification that a seller receives when they add on overage pricing. When a seller acquires new customers, instead of only getting entitlement-updated (which may refer to any kind of action), the seller receives a subscribe message indicating that this is a new customer.
- For future dated agreements (FDAs), this topic is initiated on the agreement start date (and not agreement sign date). It's also initiated when subsequent changes occur in the entitlement, such as cancellation, replacement, renewal, or expiration of the agreement.

Products with contract pricing (including contracts with pay-as-you-go) must respond to these messages. For more information about how to respond, see [Scenario: Monitor changes to user subscriptions](#).

Amazon SNS topic: aws-mp-subscription-notification

Each message in the aws-mp-subscription-notification topic has the following format.

```
{
  "action": "<action-name>",
  "customer-identifier": " X01EXAMPLEX",
  "product-code": "n0123EXAMPLEXXXXXXXXXXXXX",
}
```

```
"offer-identifier": "offer-abcexample123",  
"isFreeTrialTermPresent": "true"  
}
```

The `offer-identifier` only appears in the notification if the offer is a *private offer*.

The `isFreeTrialTermPresent` property indicates if the buyer's subscription is a free trial. The JSON value of this property is not a *boolean* datatype. Instead, the value is converted to a *string* datatype. For more information, see [SaaS free trials](#).

The `<action-name>` will vary depending on the notification. Possible actions are:

- `subscribe-success` – The `subscribe-success` message signals when the seller can begin sending metering records.
- `subscribe-fail` – If the `subscribe-fail` message is generated, payment might have failed even though the buyer has already transitioned from the AWS Marketplace to the seller's SaaS landing page. The seller should wait for the `subscribe-success` message before allowing consumption of the product.
- `unsubscribe-pending` – When a buyer unsubscribes, an `unsubscribe-pending` message is sent first. This indicates that the seller has a limited time (about one hour) to get final metering records sent before the buyer is cancelled completely.
- `unsubscribe-success` – The `unsubscribe-success` message signals the completion of cancellation, after which no further metering records will be accepted.

Note

- If a buyer unsubscribes and then immediately successfully re-subscribes before the final `unsubscribe-success` message is sent, the final `unsubscribe-success` message will not be sent and a `subscribe-success` message will be sent instead.
- For future dated agreements (FDAs), the `subscribe success` action is initiated on the agreement start date (and not agreement sign date).

Products with subscription pricing (including contracts with pay-as-you-go) must respond to these messages. For more information about how to respond, see the following topics:

- [Integrate your SaaS subscription product](#)

- [Integrate your SaaS contract with pay-as-you-go product](#)

Subscribing an SQS queue to the SNS topic

We recommend subscribing an Amazon SQS queue to the provided SNS topics. For detailed instructions on creating an SQS queue and subscribing the queue to a topic, see [Subscribing an Amazon SQS queue to an Amazon SNS topic](#) in the *Amazon Simple Notification Service Developer Guide*.

Note

You can only subscribe to AWS Marketplace SNS topics from the AWS account used to sell the products. However, you can forward the messages to a different account. For more information, see [Sending Amazon SNS messages to an Amazon SQS queue in a different account](#) in the *Amazon Simple Notification Service Developer Guide*.

Polling the SQS queue for notifications

After you subscribe your SQS queue to an SNS topic, the messages are stored in SQS. You must define a service that continually polls the queue, looks for messages, and handles them accordingly.

Accessing the AWS Marketplace Metering and Entitlement Service APIs

This section outlines the process of integrating with the AWS Marketplace Metering Service or AWS Marketplace Entitlement Service, used to ensure your billing and reporting of customer usage of your software as a service (SaaS) products is accurate. It's assumed that you've submitted a SaaS subscriptions product or a SaaS contracts product that has been published to a limited state. In a limited state, you can use your test accounts to verify proper configuration and function but your product is not available publicly.

Note

If your SaaS product is integrated with another AWS managed service that handles metering in a different way (such as Amazon SageMaker Ground Truth, or AWS WAF), then

you do not need to integrate with AWS Marketplace metering service. Metering for your product should only happen in one system to avoid double billing your customer.

Topics

- [Metering for usage](#)
- [Checking entitlements](#)
- [SaaS product integration checklist](#)

For information about setting up the AWS CLI, along with credentials, see [Configuring the AWS CLI](#) in the *AWS Command Line Interface User Guide*. If you're new to the AWS Python SDK, see the [Boto 3 Quickstart](#).

Metering for usage

For software as a service (SaaS) subscriptions, you meter for all usage, and then customers are billed by AWS based on the metering records that you provide. For SaaS contracts, you only meter for usage beyond a customer's contract entitlements. When your application meters usage for a customer, your application is providing AWS with a quantity of usage accrued. Your application meters for the pricing dimensions that you defined when you created your product, such as gigabytes transferred or hosts scanned in a given hour. For example, if you charge based on the amount of data sent into your application, you can measure the amount of data and send a corresponding metering record once an hour. AWS calculates a customer's bill using the metering data along with the prices that you provided when you created your product.

Note

Optionally, you can split the usage across properties that you track. These properties are exposed to the buyer as tags. These tags allow the buyer to view their costs split into usage by the tag values. For example, if you charge by the user, and users have a Department property, you could create a usage allocations with tags that have a key of Department, and one allocation per value. This doesn't change the price, dimensions, or the total usage that you report, but allows your customer to view their costs by categories appropriate to your product. For more information, see [Vendor-metered tagging \(Optional\)](#).

Meter on an hourly basis

We recommend that you report usage to AWS on an hourly basis for all your customers in batches of up to 25 at a time. This gives customers as much granular visibility into their usage and costs as possible. If you aggregate usage in time periods greater than an hour (for example, one day), note the following considerations.

- AWS can only bill customers for usage of your product upon receiving metering records from you. You're responsible for ensuring that your product's metering records are successfully transmitted and received. You can use AWS CloudTrail to verify the record or records that you send are accurate. You can also use the information to perform audits over time. For more information, see [Logging AWS Marketplace Metering API calls with AWS CloudTrail](#).
- If this is a SaaS with the pricing model "Subscription" (not pricing models "Contract" or "Contract with Consumption"), then the buyer can unsubscribe at any time. When the buyer initiates this unsubscribe action, the seller will receive an unsubscribe-pending [notification](#) and have 1 hour to send in all unreported usage before the final unsubscribe-success notification. Anything after the second notification will not be honored. The other two pricing models have a set duration based on the time of subscription and the buyer cannot unsubscribe during it. They can only turn off autorenewal. The same notification is sent at the end of that duration if not autorenewing.
- If you don't send metering records hourly and there is an application or network outage, your records will be further behind. This may result in unreported usage if the application or network outage is restored after the subscription expires.
- Even if there is no usage to report, you can continue sending metering records every hour and record a quantity of 0 if there is no usage to report for that hour. Note that after you report a buyer's usage of a dimension, 0 or more, you cannot amend the record. Thus it is best practice to report usage for the previous hour.
- During publishing, the AWS Marketplace Operations team will test that the SaaS application sends the metering record successfully before allowing the product to be published. Typically, the team will perform a mock sign up of the SaaS and confirm that a metering record is received.

Note

If your SaaS product is integrated with another AWS managed service that handles metering in a different way (such as Amazon SageMaker Ground Truth, or AWS WAF), then you do not need to integrate with AWS Marketplace metering service. Metering for your

product should only happen in one system to avoid double billing your customer. Note that AWS Marketplace isn't publishing new AWS WAF products at this time.

Configure your product to meter usage

You use the `BatchMeterUsage` operation in the AWS Marketplace Metering Service to deliver metering records to AWS. Keep the following in mind:

- We require sellers to use batching by using the `BatchMeterUsage` operation.
- We deduplicate metering requests on the hour.
 - Requests are deduplicated per product/customer/hour/dimension.
 - You can always retry any request, but if you meter for a different quantity, the original quantity is billed.
 - If you send multiple requests for the same customer/dimension/hour, the records are not aggregated.
- Sellers can send metering records with a timestamp up to 6 hours in the past if the customer is subscribed to your product. If the customer unsubscribes, sellers have to send the metering records within 1 hour of the customer unsubscribing.
- `BatchMeterUsage` payloads must not exceed 1MB. Choose the number of usage records to send in a `BatchMeterUsage` request so that you don't exceed the size of the payload.
- The AWS Marketplace Metering Service is available in the AWS Regions listed in [AWS Marketplace endpoints and quotas](#) in the *AWS General Reference*. By default, the US East (N. Virginia) Region is enabled for SaaS metering products when you request your product. If you intend to use other Regions, contact the [AWS Marketplace Seller Operations](#) team. For more information, see [BatchMeterUsage](#).

For code examples, see [Code examples for SaaS product integration](#).

Example: Host scanning

Your product analyzes computing hardware for known security vulnerabilities. Customers manually initiate or schedule these scans of their Amazon Elastic Compute Cloud (Amazon EC2) instances. As your product performs these scans, it tallies the number of unique hosts scanned every hour. In this example, your product uses the **Hosts** category. You can declare multiple dimensions for the types of hosts scanned. For example, you can charge different prices for small, medium, and large hosts.

Example: Log analysis

Your SaaS product digests logs that are generated by customer products, reporting trends, and anomalies. As customers upload logs to your product, you measure the quantity of data received in megabytes, gigabytes, or terabytes. On the tenth minute of every hour, a cron job reads this usage for each customer for the previous hour. The job builds a batch report and uses the `BatchMeterUsage` operation to send it to AWS. In this example, your product uses the **Data** category. Your product can also meter for the amount of log data stored for any given hour. In this case, your product can meter along two dimensions: data received in the hour and total data stored in the hour. You can continue to meter for data stored until the customer deletes this data or it expires.

Vendor-metered tagging (Optional)

Vendor-metered tagging helps Independent Software Vendors (ISVs) give the buyer more granular insight into their software usage and can help them perform cost allocation.

There are many ways to tag a buyer's software usage. One way is to first ask your buyers what they want to see in their cost allocation. Then you can split the usage across properties that you track for the buyer's account. Examples of properties include `Account ID`, `Business Unit`, `Cost Centers`, and other relevant metadata for your product. These properties are exposed to the buyer as tags. Using tags, buyers can view their costs split into usage by the tag values in their AWS Billing Console (<https://console.aws.amazon.com/billing/>). Vendor-metered tagging doesn't change the price, dimensions, or the total usage that you report. It allows your customer to view their costs by categories appropriate to your product.

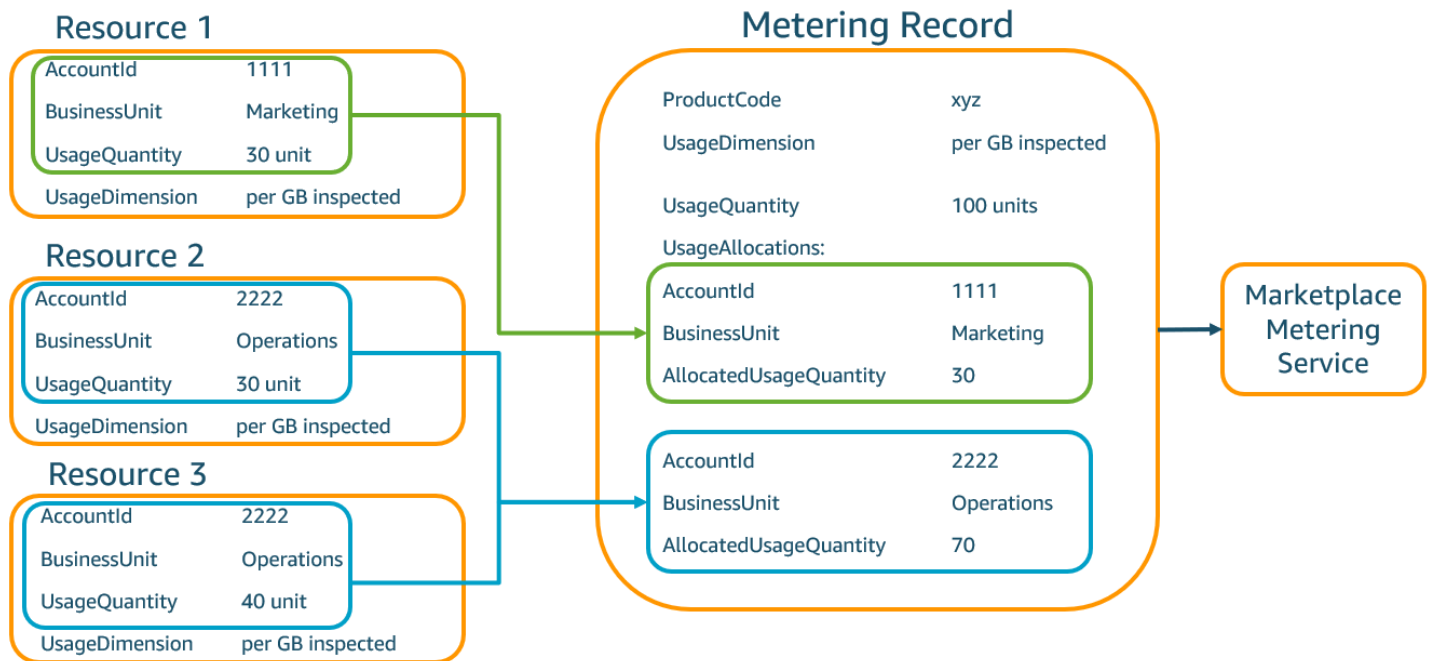
In a common use case, a buyer subscribes to your product with one AWS account. The buyer also has numerous users associated with the same product subscription. You can create usage allocations with tags that have a key of `Account ID`, and then allocate usage to each user. In this case, buyers can activate the `Account ID` tag in their Billing and Cost Management console and analyze individual user usage.

Seller experience

Sellers can aggregate the metering records for resources with the same set of tags instead of aggregating usage for all resources. For example, sellers can construct the metering record that includes different buckets of `UsageAllocations`. Each bucket represents `UsageQuantity` for a set of tags, such as `AccountId` and `BusinessUnit`.

In the following diagram, **Resource 1** has a unique set of AccountId and BusinessUnit tags, and appears in the **Metering Record** as a single entry.

Resource 2 and **Resource 3** both have the same AccountId tag, 2222, and the same BusinessUnit tag, Operations. As a result, they're combined into a single UsageAllocations entry in the **Metering Record**.



Sellers can also combine resources without tags into a single UsageAllocation and send it as one of the entries in UsageAllocations.

Limits include:

- Number of tags – 5
- Size of UsageAllocations (cardinality) – 2,500
- Maximum request size – 1 MB

Validations include:

- Characters allowed for the tag key and value – a-zA-Z0-9+ -=:\/@
- Maximum tags across UsageAllocation list – 5
- Two UsageAllocations can't have the same tags (that is, the same combination of tag keys and values). If that's the case, they must use the same UsageAllocation.

- The sum of `AllocatedUsageQuantity` of `UsageAllocation` must equal the `UsageQuantity`, which is the aggregate usage.
- The maximum payload size can't be more than 1 MB. This includes input attribute keys (for example, `UsageRecords`, `AllocatedUsageQuantity`, tags).

Note

To make sure that you aren't breaching the payload limit, create a sample request object with a maximum size based on the business requirement, convert the object into a JSON string, and obtain the size in bytes. Make sure that a single API call won't breach the 1 MB limit. For example, if a request with 1 `UsageRecord` has a maximum size of 200 KB, don't send more than 5 `UsageRecords` as part of the request (200KB * 5 = 1MB).

Buyer experience

The following table shows an example of the buyer experience after a buyer activates the `AccountId` and `BusinessUnit` vendor tags.

In this example, the buyer can see allocated usage in their **Cost Usage Report**. The vendor-metered tags use the prefix `aws:marketplace:isv`. Buyers can activate them in the Billing and Cost Management, under **Cost Allocation Tags, AWS-generated cost allocation tags**.

The first and last rows of the **Cost Usage Report** are relevant to what the Seller sends to the Metering Service (as shown in the [Seller experience](#) example).

Cost Usage Report (Simplified)

ProductCode	Buyer	UsageDimension	UsageQuantity	aws:marketplace:isv:AccountId	aws:marketplace:isv:BusinessUnit
xyz	111122223333	Network: per (GB) inspected	70	2222	Operations

ProductCode	Buyer	UsageDimension	UsageQuantity	aws:marketplace:isv:AccountId	aws:marketplace:isv:BusinessUnit
xyz	111122223333	Network: per (GB) inspected	30	3333	Finance
xyz	111122223333	Network: per (GB) inspected	20	4444	IT
xyz	111122223333	Network: per (GB) inspected	20	5555	Marketing
xyz	111122223333	Network: per (GB) inspected	30	1111	Marketing

For a code example, see [BatchMeterUsage with usage allocation tagging code example \(Optional\)](#).

Checking entitlements

If your product is a SaaS contracts product, your product calls the AWS Marketplace Entitlement Service to retrieve the customer's entitlement using the [GetEntitlements](#). Your product should verify subsequent usage on that account against the AWS Marketplace Entitlement Service. For example, if the customer provisions 10 users on the account, your product should check the AWS Marketplace Entitlement Service for entitlement to that capacity.

To verify a customer's entitlement to your product, use the `GetEntitlements` operation in the AWS Marketplace Entitlement Service. The AWS Marketplace Entitlement Service is available only in the US East (N. Virginia) Region, accessible through `entitlement.marketplace.us-east-1.amazonaws.com`.

GetEntitlements accepts a customer identifier and dimension as filters. ProductCode is a required parameter. The operation returns a paginated list of entitlements. The result has an ExpirationDate field that shows the minimum period of time that the entitlement is valid for. If the customer has set up automatic renewal, the date in the ExpirationDate field is the renewal date.

For code samples, see [Code examples for SaaS product integration](#).

Retrieving entitlement on user actions

The following examples can help you better understand the process for retrieving entitlement on user actions.

Example: User-based product

You offer a product that allows a number of accounts to exist for a given customer. The customer can visit a dashboard to provision new users (for example, to assign credentials). When the customer provisions a new user, your product calls GetEntitlements to verify that the capacity exists. If it does not, you can call the AWS Marketplace Metering Service to bill for additional users.

Example: Data storage product

You offer a product that enables customers to store a certain amount of data in encrypted or unencrypted form. The customer can view a dashboard that displays the amount of data existing and allocated in your product. Your dashboard retrieves the allocation amount through GetEntitlements.

SaaS product integration checklist

Before your SaaS product goes live, use this checklist to verify that you have completed the required configuration.

Category	Requirements
Access	Submitted a seller registration form with the desired AWS account for AWS Marketplace usage.

Category	Requirements
Access	Completed the seller registration, including terms and conditions, bank account, and W8 or W9 tax form.
Access	Configured cross-account roles for the registered AWS Marketplace account.
Product	Completed the product request form in the AWS Marketplace Management Portal.
Product	Provided AWS account IDs for testing in the Notes tab of the Create product wizard in the AMMP.
Product	Provided a URL of the EULA in .txt format in the Products tab.
Product	Received your product code and Amazon SNS topic information from AWS Marketplace.
Product	Subscribed to the Amazon SNS topic and created an Amazon SQS queue to subscribe to the Amazon SNS topic.
Billing Solution	Validated you can send metering records to the <code>BatchMeterUsage</code> operation each hour for each customer for SaaS subscriptions products. Can send metering records for additional usage by each customer for SaaS contracts products.
Billing Solution	Validated you can verify customer entitlements from the AWS Marketplace Entitlement Service for SaaS contracts products.
Billing Solution	Validated that the costs appear as expected on bills generated for test accounts.

Category	Requirements
Billing Solution	Tested for situations such as invalid customer IDs and canceled subscriptions.
Product	Submitted the product request back to AWS Marketplace for publishing.
Registration	Implemented an HTTPS registration page that can accept HTTP POST requests.
Registration	Validated you can accept new customer registrations.
Registration	Validated you are <i>not</i> storing the registration token in a cookie.
Registration	Validated you are using <code>ResolveCustomer</code> to obtain the <code>ProductCode</code> and <code>CustomerIdentifier</code> from the AWS token.
Registration	Validated you can resolve the registration token received from AWS with no delays.
Registration	Tested that you aren't blocked from registering with email services addresses such as Gmail.
Registration	Tested that you can accept incomplete registrations and multiple registration attempts.
Subscription	Test that you can handle <code>unsubscribe-pending</code> and <code>unsubscribe-success</code> messages.
Subscription	Validated that you send final metering records within an hour of receiving an <code>unsubscribe-pending</code> message.

Category	Requirements
Security	Validated the AWS root account doesn't have API keys, has a strong password, and is associated with a hardware multi-factor authentication (MFA) device. All administrative access is through identities created with AWS Identity and Access Management (IAM). No shared accounts.
Security	Validated that IAM roles are used for all programmatic Amazon Elastic Compute Cloud (Amazon EC2) access. Credentials aren't hard-coded into scripts, headers, or source code.
Security	Validated you maintain comprehensive logging and log consolidation.
Security	Verified you have well-defined public and private subnet boundaries that isolate application services and access to database and file systems. Distinct data class definitions that demarcate sensitive data and segregate public and private data.
Security	Verified you have private data encryption in transit and at rest with scheduled key rotation.
Security	Validated you have security incident tools and access in place and routinely scheduled incident response exercises that accommodate timely investigation and recovery.
Reliability	Verified the system adapts to changes in demand, scaling up and down as required, and employs load balancing to ensure high performance. The system also provides edge-based caching as required.

Category	Requirements
Reliability	Validated recovery time and point objective s are specified, and disaster recovery is scheduled at regular intervals. Component failure is self-healing via automated triggers and notifications.

Reporting

AWS Marketplace produces reports for your SaaS products that include data on subscribers, financials, usage, and taxes. For more information, see [the section called “Seller reports”](#). The following table summarizes how financials for SaaS products are reported.

Report	SaaS content
Daily business report	Upfront contract charges appear in the Fees section. Metered usage charges appear in the Usage section.
Monthly revenue report	Upfront contract charges appear in the Annual subscriptions section. Metered usage charges appear in the Billing and revenue data section.
Sales compensation report	Upfront contract charges and monthly additional usage charges appear as separate line items.
Customer subscriber report	New SaaS contracts appear in the Annual subscriptions section. New SaaS subscriptions appear in the Hourly/monthly subscriptions section.

Code examples for SaaS product integration

The following code examples can help you integrate your software as a service (SaaS) product with the AWS Marketplace APIs that are required for publishing and maintaining your product.

Topics

- [ResolveCustomer code example](#)
- [GetEntitlement code example](#)
- [BatchMeterUsage code example](#)
- [BatchMeterUsage with usage allocation tagging code example \(Optional\)](#)

ResolveCustomer code example

The following code example is relevant for all pricing models. The Python example exchanges a `x-amzn-marketplace-token` token for a `CustomerIdentifier`, `ProductCode`, and `CustomerAWSAccountId`. The `CustomerAWSAccountId` is the AWS account Id associated with the subscription. This code runs in an application on your registration website, when you are redirected there from the AWS Marketplace Management Portal. The redirect is a POST request that includes the token.

For more information about `ResolveCustomer`, see [ResolveCustomer](#) in the *AWS Marketplace Metering Service API Reference*.

```
# Import AWS Python SDK and urllib.parse
import boto3
import urllib.parse as urlparse

# Resolving Customer Registration Token
formFields = urlparse.parse_qs(postBody)
regToken = formFields['x-amzn-marketplace-token'][0]

# If regToken present in POST request, exchange for customerID
if (regToken):
    marketplaceClient = boto3.client('meteringmarketplace')
    customerData = marketplaceClient.resolve_customer(RegistrationToken=regToken)
    productCode = customerData['ProductCode']
    customerID = customerData['CustomerIdentifier']
    customerAWSAccountId = customerData['CustomerAWSAccountId']
```

```
# TODO: Store customer information
# TODO: Validate no other accounts share the same customerID
```

Example response

```
{
  'CustomerIdentifier': 'string',
  'CustomerAWSAccountId': 'string',
  'ProductCode': 'string'
}
```

GetEntitlement code example

The following code example is relevant for SaaS products with the contract and SaaS contract with consumption pricing model. The Python example verifies that a customer has an active entitlement.

For more information about GetEntitlement, see [GetEntitlement](#) in the *AWS Marketplace Entitlement Service API Reference*.

```
# Import AWS Python SDK
import boto3

marketplaceClient = boto3.client('marketplace-entitlement', region_name='us-east-1')

# Filter entitlements for a specific customerID
#
# productCode is supplied after the AWS Marketplace Ops team has published
# the product to limited
#
# customerID is obtained from the ResolveCustomer response
entitlement = marketplaceClient.get_entitlements({
    'ProductCode': 'productCode',
    'Filter' : {
        'CUSTOMER_IDENTIFIER': [
            'customerID',
        ]
    },
    'NextToken' : 'string',
    'MaxResults': 123
})
```

```
# TODO: Verify the dimension a customer is subscribed to and the quantity,  
# if applicable
```

Example response

The returned value corresponds to the dimensions created when you created the product in the AWS Marketplace Management Portal.

```
{  
  "Entitlements": [  
    {  
      "CustomerIdentifier": "string",  
      "Dimension": "string",  
      "ExpirationDate": number,  
      "ProductCode": "string",  
      "Value": {  
        "BooleanValue": boolean,  
        "DoubleValue": number,  
        "IntegerValue": number,  
        "StringValue": "string"  
      }  
    }  
  ],  
  "NextToken": "string"  
}
```

BatchMeterUsage code example

The following code example is relevant for SaaS subscription and contract with consumption pricing models, but not for SaaS contract products without consumption. The Python example sends a metering record to AWS Marketplace to charge your customers for pay-as-you-go fees.

```
# NOTE: Your application will need to aggregate usage for the  
#       customer for the hour and set the quantity as seen below.  
#       AWS Marketplace can only accept records for up to an hour in the past.  
#  
# productCode is supplied after the AWS Marketplace Ops team has  
# published the product to limited  
#  
# customerID is obtained from the ResolveCustomer response
```

```
# Import AWS Python SDK
import boto3

usageRecord = [
    {
        'Timestamp': datetime(2015, 1, 1),
        'CustomerIdentifier': 'customerID',
        'Dimension': 'string',
        'Quantity': 123
    }
]

marketplaceClient = boto3.client('meteringmarketplace')

response = marketplaceClient.batch_meter_usage(usageRecord, productCode)
```

For more information about BatchMeterUsage, see [BatchMeterUsage](#) in the *AWS Marketplace Metering Service API Reference*.

Example response

```
{
  'Results': [
    {
      'UsageRecord': {
        'Timestamp': datetime(2015, 1, 1),
        'CustomerIdentifier': 'string',
        'Dimension': 'string',
        'Quantity': 123
      },
      'MeteringRecordId': 'string',
      'Status': 'Success' | 'CustomerNotSubscribed' | 'DuplicateRecord'
    },
  ],
  'UnprocessedRecords': [
    {
      'Timestamp': datetime(2015, 1, 1),
      'CustomerIdentifier': 'string',
      'Dimension': 'string',
      'Quantity': 123
    }
  ]
}
```

```
}
```

BatchMeterUsage with usage allocation tagging code example (Optional)

The following code example is relevant for a SaaS subscription and contract with consumption pricing models, but not for SaaS contract products without consumption. The Python example sends a metering record with appropriate usage allocation tags to AWS Marketplace to charge your customers for pay-as-you-go fees.

```
# NOTE: Your application will need to aggregate usage for the
#       customer for the hour and set the quantity as seen below.
#       AWS Marketplace can only accept records for up to an hour in the past.
#
# productCode is supplied after the AWS Marketplace Ops team has
# published the product to limited
#
# customerID is obtained from the ResolveCustomer response

# Import AWS Python SDK
import boto3
import time

usageRecords = [
    {
        "Timestamp": int(time.time()),
        "CustomerIdentifier": "customerID",
        "Dimension": "Dimension1",
        "Quantity": 3,
        "UsageAllocations": [
            {
                "AllocatedUsageQuantity": 2,
                "Tags":
                    [
                        { "Key": "BusinessUnit", "Value": "IT" },
                        { "Key": "AccountId", "Value": "123456789" },
                    ]
            },
            {
                "AllocatedUsageQuantity": 1,
                "Tags":
```

```

        [
            { "Key": "BusinessUnit", "Value": "Finance" },
            { "Key": "AccountId", "Value": "987654321" },
        ]
    },
]
}

marketplaceClient = boto3.client('meteringmarketplace')

response = marketplaceClient.batch_meter_usage(UsageRecords=usageRecords,
        ProductCode="testProduct")

```

For more information about BatchMeterUsage, see [BatchMeterUsage](#) in the *AWS Marketplace Metering Service API Reference*.

Example response

```

{
  "Results": [
    {
      "Timestamp": "1634691015",
      "CustomerIdentifier": "customerID",
      "Dimension": "Dimension1",
      "Quantity": 3,
      "UsageAllocations": [
        {
          "AllocatedUsageQuantity": 2,
          "Tags": [
            { "Key": "BusinessUnit", "Value": "IT" },
            { "Key": "AccountId", "Value": "123456789" },
          ]
        },
        {
          "AllocatedUsageQuantity": 1,
          "Tags": [
            { "Key": "BusinessUnit", "Value": "Finance" },
            { "Key": "AccountId", "Value": "987654321" },
          ]
        }
      ]
    }
  ]
}

```

```
    ]
  },
]
},
"MeteringRecordId": "8fjef98ejf",
"Status": "Success"
},
],
"UnprocessedRecords": [
  {
    "Timestamp": "1634691015",
    "CustomerIdentifier": "customerID",
    "Dimension": "Dimension1",
    "Quantity": 3,
    "UsageAllocations": []
  }
]
}
```

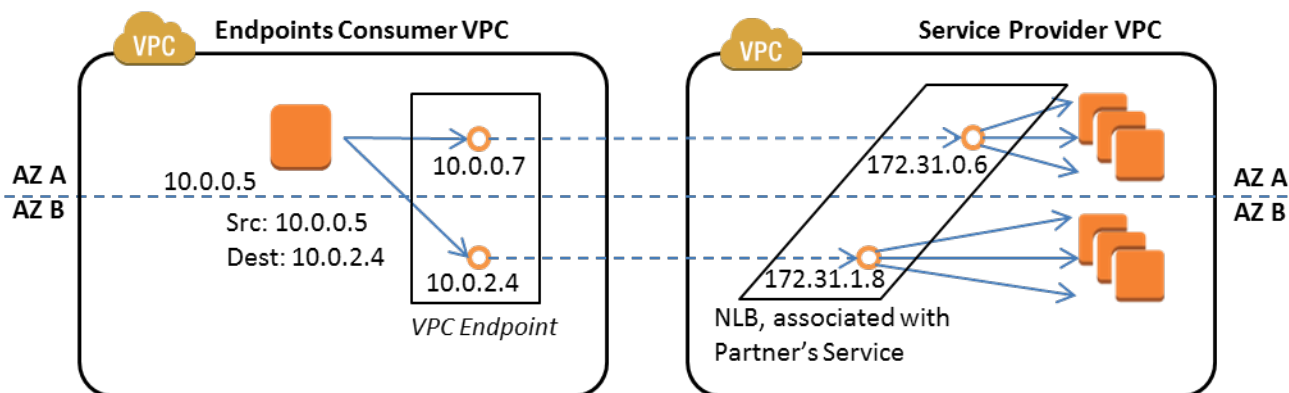
Using AWS PrivateLink with AWS Marketplace

AWS Marketplace supports AWS PrivateLink, a technology that allows you to use the Amazon network to provide buyers with access to products you sell through AWS Marketplace. This document outlines the process for configuring and delivering your products through an Amazon Virtual Private Cloud (VPC) endpoint using AWS PrivateLink technology.

In this document, we assume that you have working knowledge of several AWS services and the AWS Marketplace environment.

Introduction

As an AWS Marketplace seller, you can provide buyers access to your service through an Amazon VPC endpoint. This approach provides buyers with access to your service across the Amazon network using [AWS PrivateLink](#) technology. If you use AWS Marketplace to create and deliver this offering, buyers can discover your service in AWS Marketplace. Your buyers can also find your product in the list of available services for creating a VPC endpoint.



A [VPC endpoint](#) is a virtual device that enables AWS customers to create a private connection between their VPC and another AWS service without requiring access over the internet, through a NAT device, a VPN connection, or AWS Direct Connect. You can create an endpoint service through AWS Marketplace that makes it possible for buyers to use this technology to connect to your service. This connection method is more secure for your buyers because they access your service through the Amazon private network rather than through the Internet.

For each region where you want to offer your service, you create or use existing resources to configure a VPC, set up your service instances, set up a network load balancer, and register your services with the network load balancer by creating a service endpoint. After you complete those steps and test your offering, you provide your configuration information to the the [AWS Marketplace Seller Operations](#) team.

AWS recommends that you provide a private DNS name that your buyers can use when they create VPC endpoints.

When buyers create their VPC endpoints, they have the option to enable a private DNS name. By choosing this option, the buyer's VPC service configures a [private hosted zone](#). If you provide the private DNS name, buyers can use it when configuring VPC endpoints to connect to your service. In the buyer's private hosted zone, the private DNS name (api.example.com) will point to the randomly generated DNS name(s) (vpce-1111111111111111-yyy-yyyyy.api.vpce.example.com) created for your endpoint service(s). The buyer's EC2 instances call the same unified DNS name (api.example.com) across different VPCs. Also, if public and private DNS names are same, the buyer can use the same public name when accessing your service from within or outside of the VPC.

For assistance with making your service available through AWS Marketplace, you can contact the [AWS Marketplace Seller Operations](#) team. When an AWS Marketplace buyer subscribes to your service and creates a VPC endpoint, your service is shown under **Your AWS Marketplace**

Services. The AWS Marketplace Seller Operations team uses the user-friendly DNS name for ease of discovery of your service when creating the VPC endpoint.

Your product is created as a software as a service (SaaS) product. Metering and billing is the same as with other AWS Marketplace SaaS products.

Configuring your product

To configure your product to be available through an Amazon VPC endpoint:

1. Create or use an existing [Amazon VPC](#).
2. Create (or use existing) [Amazon EC2](#) instance(s) for your product.
3. Create a [network load balancer](#) in each of the regions where you offer your product. AWS recommends that you include all [Availability Zones](#) (AZs) for a region.
4. Use the Amazon VPC console, the CLI, or supported SDKs to create a VPC endpoint service.
5. Verify that you can access the service through the network load balancer.
6. [Request a certificate from AWS Certificate Manager \(ACM\)](#) for your user-friendly DNS name. Before ACM issues a certificate, it validates that you own or control the domain names in your certificate request.
7. Delegate the subdomain of your user-friendly DNS name, such as `api.vpce.example.com`, to the name servers provided to you by the AWS Marketplace Seller Operations team. In your DNS system, you must create a name server (NS) resource record to point this subdomain to the Amazon Route 53 name servers provided by the AWS Marketplace Seller Operations team so that DNS names (such as `vpce-0ac6c347a78c90f8.api.vpce.example.com`) are publicly resolvable.
8. Allow access to your buyers' AWS accounts.

Note: You can use a supported SDK or this CLI command to automate access to accounts: `aws vpcev2 modify-vpc-endpoint-service-permissions --service-id vpce-svc-0123456789abcdef1 --add-allowed-principals arn:aws:iam::111111111111:root arn:aws:iam::222222222222:root`.

Submitting your product to AWS Marketplace

During the process of publishing your service to AWS Marketplace, you work with the AWS Marketplace Seller Operations team. To submit your PrivateLink-enabled product:

1. Email the following information to the [AWS Marketplace Seller Operations](#) team:

- a. The endpoint and the AWS account used to create the endpoint. The endpoint is similar to this: `com.amazonaws.vpce.us-east-1.vpce-svc-0daa010345a21646`
- b. The user-friendly DNS name for your service. This is the DNS name that AWS Marketplace buyers use to access your product.
- c. The AWS account that you used to request certificates and the private DNS name buyers use to access the VPC endpoint.

The AWS Marketplace Seller Operations team verifies your company's identity and the DNS name to use for the service you are registering (such as `api.vpce.example.com`). After verification, the DNS name overrides the default base endpoint DNS name.

Buyer access to VPC endpoints

AWS Marketplace buyers who are creating a VPC endpoint can discover your service in these situations:

- You followed the seller processes described earlier on this page to create or use an existing product.
- The buyer subscribes to your service.
- You added the buyer's AWS account to your list of allowed accounts.

When the buyer creates the VPC endpoint, they have the option to associate a private hosted zone with their VPC. The hosted zone contains a record set for the default private DNS name for the service that resolves to the private IP address of the endpoint network interfaces in their VPC.

Any buyer-hosted endpoint, including AWS Marketplace services, can provide permissions to all accounts (the "*" permission). However, when you use this approach, the services aren't included in the **Describe** calls or console unless you search by the service name. To display the services in the **Describe** calls, the buyer's AWS account must be explicitly added to the allow list by the service.

To access your service, buyers do the following:

1. Discover and subscribe to your service on AWS Marketplace.
2. Use the AWS Command Line Interface (AWS CLI), API, or the Amazon VPC console to discover your service and then establish a VPC endpoint to connect to your service in the subnets and AZs they use. The endpoints are shown as elastic network interfaces in the subnets. Local IP addresses and region and zonal DNS names are assigned to the endpoints.

Client-side DNS name	Name
Regional	Vpce<0dc9a211a78c90f8>.api.vpce.example.com
IAD2 (1a)	us-east-1a -Vpce<0dc9a211a78c90f8>.api.vpce.example.com
IAD2 (1b)	us-east-1b -Vpce<0dc9a211a78c90f8>.api.vpce.example.com

If you provided a default private DNS name and the buyer chooses **Enable Private DNS Name** (associated a private hosted zone) when creating a VPC endpoint, the buyer sees the regional default private DNS name to connect to your service.

Name	Alias	Alias hosted zone ID	(Notes)
api.example.com	vpce<0dc9a211a78c90f8>.api.vpce.example.com	Z00AABBCCDD	IAD1 IAD2

Appendix: Checklists

Use the following checklists to ensure that you configure and test your product before you submit it to the AWS Marketplace Seller Operations team.

Product creation checklist

- Create (or use an existing) VPC and then configure it.
- Create and configure a network load balancer within the VPC.
- Register your service with your network load balancer by creating a VPC endpoint service.
- Provide the AWS account ID you used to configure the VPC endpoint to the AWS Marketplace Seller Operations team.
- Provide the default endpoint service name (for example, com.amazonaws.vpce.us-east-1.vpce-svc-0bbb070044a2164) to the AWS Marketplace Seller Operations team.

- Provide a user-friendly service DNS name (required) to override the randomly generated service DNS name. Request SSL certificates from ACM for the subdomain used for your user-friendly service DNS name. Provide these certificates and the AWS account ID you used to request them to the AWS Marketplace Seller Operations team.
- Recommended: Provide a private DNS name.
- Create a process to inform and allow your AWS Marketplace buyers the option to connect to your service using AWS PrivateLink technology. Add AWS account IDs for your buyers to your allowed list of accounts.

Product testing

- Verify that your service is configured and discoverable.
- Verify that your service is discoverable over the network load balancer.
- Verify that a buyer can create a VPC endpoint and access your service. Use an AWS account you own that is not the account you used to set up your service.

Professional services products

As a seller, you can offer professional services to AWS Marketplace buyers. Professional services include services to assess, migrate, support, manage, and train others in how to use AWS services and products in AWS Marketplace. Sellers create a product offering that describes the services they provide, negotiate with customers to create an agreement on terms, and then create a custom offer for services through AWS Marketplace.

Note

As an independent software vendor (ISV), channel partner, or consulting partner, you can also authorize another partner to resell your professional services products using channel partner private offers. For more information, see [Creating a resell opportunity for a channel partner as an ISV](#).

Buyers can find professional services products on the AWS Marketplace catalog by selecting **Professional Services** under **Categories**, choosing **Professional Services** under **Delivery methods**, and refine their search by **Publisher**, **Pricing model**, and **Pricing unit**. They're charged for the services in their AWS bill. They can use tools such as AWS Cost Explorer to centralize payments and manage their costs.

For more information about professional services products, see:

- [Getting started with professional services products](#)
- [Providing details for a professional services product](#)
- [Requirements for professional services products](#)
- [Creating a resell opportunity for a channel partner as an ISV](#)

The following video explains more about managing professional services products in AWS Marketplace.

[Manage Professional service products in AWS Marketplace.](#)

Getting help

For assistance with your professional services products, contact your business development partner for AWS Marketplace or the [AWS Marketplace Seller Operations](#) team.

Getting started with professional services products

This topic describes how to get started with a professional services product, and goes through the steps to create your first product, and how to offer it to your customers. Your product definition tells your customers about the services that you offer and why they should select your company for those services. AWS Marketplace then allows them to contact you. You agree on a contract, and then you create a private offer that allows them to purchase your services for a fixed cost.

Topics

- [Prerequisites](#)
- [Creating a professional services product](#)
- [Creating private offers](#)
- [Editing product information](#)
- [Editing product pricing](#)
- [Editing product visibility](#)
- [Removing a professional services product](#)

The following video explains more about listing professional services products in AWS Marketplace.

Prerequisites

To sell professional services on AWS Marketplace, you must complete the following prerequisites:

- Have access to the AWS Marketplace Management Portal. This is the tool that you use to register as a seller and manage the products that you sell on AWS Marketplace. To learn more about getting access to the AWS Marketplace Management Portal, see [Policies and permissions for AWS Marketplace sellers](#).
- Register as an AWS Marketplace seller and submit your tax and banking information. To learn more about becoming an seller, see [Getting started as a seller](#).

- You must have a professional services product to offer that is related to an AWS service or at least one public product in AWS Marketplace. Your product must either directly support those products, or offer services that drive subscriptions to those products.

Note

Your product must be listed in at least one of these primary categories: Assessments, Implementation, Managed services, Premium support, or Training.

For more information about professional services product guidelines, see [Requirements for professional services products](#).

Creating a professional services product

The following procedure describes how to create a new professional services product in the AWS Marketplace Management Portal.

To create a professional services product

1. Open a web browser and sign into the [AWS Marketplace Management Portal](#).
2. From the **Products** menu, select **Professional services**. This page shows you all professional services products that you have already created, as well as any requests you have made for creating or modifying these products.
3. On the **Professional Services products** tab, select **Create professional services product**.
4. On the **Create product** page, provide the information for your product, and select **Submit**. For more information about the details that you must provide, see [Providing details for a professional services product](#).
5. (Optional) From the **Products** menu of AWS Marketplace Management Portal, select **Professional services**, then choose the **Requests** tab. Verify that you see your product request with the correct **Product title**, and that the **Request status** is **Under review**. Your product should be created in limited preview mode within a few minutes.

Note

You can return to the **Requests** tab of the **Professional services** page to see the status of your request at any time. Any errors in the creation process will appear here. You can select the request to see the request details or to fix errors.

When your product is initially created, it's only accessible to your AWS account (the one you used to create the product). If you view the product from the **Professional services** page, you can select **View on AWS Marketplace** to view the product details as they appear in AWS Marketplace for buyers. This detail listing isn't available to other AWS Marketplace users, unless you extend a private offer to them.

To learn how to make the product available publicly, see [Editing product visibility](#).

Creating private offers

When a potential buyer views your product on AWS Marketplace, they can't purchase it directly. When they attempt to subscribe, they are redirected to request a [private offer](#) from you. AWS Marketplace sends an email message to your AWS Marketplace seller account root user email address, informing you that the customer has requested a private offer. The following procedure describes how to respond to this request.

Note

When you create a private offer higher than \$250,000 through the AWS Marketplace Management Portal, additional approval may be required. For more information, contact your AWS Marketplace Business Development representative, or send an email message with your details to the AWS Marketplace Business operations team at mpcustdesk@amazon.com.

To create a private offer for a professional services product

1. Contact the customer to resolve any questions you have about the request. Agree on the offer terms before creating the private offer in AWS Marketplace. The buyer is not obligated to purchase your product, so it makes sense to agree before creating the offer.
2. Open a web browser and sign into the [AWS Marketplace Management Portal](#).

3. Select **Offers** from the menu, then select **Create private offer**.
4. On the **Create private offer** page, select the product that you want to create a private offer for. Only currently available products are included.
5. Enter the **Buyer account Ids** that you want to extend a private offer to. You can include up to 25 accounts in a single private offer. If the buyer used the request an offer feature, the email message that you received includes the buyer account Id for the requesting account.
6. Choose whether to allow buyers to pay for the product in installments. Typically, short contracts are paid in one payment. Longer contracts may have the option to pay in installments, but this is part of the agreement that you come to with the buyer. Select **Next**.
7. Complete the offer details, including the following information:
 - **Custom offer name** – Provide enough detail that you and the customers will recognize the offer. Include your company or product name and a description of the product. Do not include any personally identifiable information, including names, phone numbers, or addresses.
 - **Agreement end date** – The date that the agreed-to services end. For example, if you are offering support for 1 year, enter a date that is 1 year away from the date that the service will be available.
 - **Product dimensions** – The prices and units for the service that you are offering, as follows:
 - Lump sum payment offers – You can list each of the dimensions with their associated price (for example, you could have dimensions called *Silver*, *Gold*, and *Platinum*). The buyer can choose and pay for their preference.
 - Offers that include a payment schedule – You must choose a single dimension and provide a payment schedule with amounts and dates for each payment.

 **Note**

If you want to create a zero dollar offer, you must select **I want to enable zero dollar prices** for confirmation. This precaution helps to prevent you from accidentally creating a free offer.

- **Service agreement** – Documents that define your service agreement with the customer. The documents that you upload (in text or PDF formats) are appended together into a single PDF document, so make sure that the file name is not required to understand the content.

- **Offer expiration date** – The date the offer expires. This determines how long the buyer has to accept the offer and is unrelated to when the professional service will be available. You can extend the offer expiration date after your offer has been created.
8. Select **Next** when you're done editing the options.
 9. On the **Review offer** page, make sure that the offer details are correct, and then choose **Create offer**.

Note

Your offer may take some time to be published. After it's published, you can view the offer on the **Manage offers** page. If you need to edit an offer (that has not yet been accepted), you can do so from that page.

10. After the offer is published, and available on the **Manage private offers** page, from the **Actions** menu for that offer, select **Copy offer URL**, and then send it in an email message to the buyer to accept.

Editing product information

The following procedure describes how to edit the product information for an existing professional services product in the AWS Marketplace Management Portal.

To edit product information

1. Open a web browser and sign into the [AWS Marketplace Management Portal](#).
2. From the **Products** menu, select **Professional services**. This page shows you all professional services products that you have already created, as well as any requests you have outstanding for creating or modifying these products.
3. Select an existing product that you would like to edit. Then, from the **Request changes** menu, select **Update product information**.
4. Make the changes to the details. For more information about the fields you can edit, see [Providing details for a professional services product](#).
5. Select **Submit** to create the request.
6. (Optional) If you are not already on the **Requests** tab of the **Professional services** page, then from the **Products** menu of AWS Marketplace Management Portal, select **Professional services**, then choose the **Requests** tab. Verify that you see your request with the correct

Product title, and that the **Request status** is **Under review**. Your product will be updated with the changes you requested within a few minutes. If there is an error, you can view it here and resubmit your edits after fixing the errors.

Editing product pricing

The following procedure describes how to edit the pricing information for an existing professional services product in the AWS Marketplace Management Portal.

To edit product pricing

1. Open a web browser and sign into the [AWS Marketplace Management Portal](#).
2. From the **Products** menu, select **Professional services**. This page shows you all professional services products that you have already created, as well as any requests you have made for creating or modifying these products.
3. Select an existing product that you would like to edit, then from the **Request changes** menu, select **Update pricing dimensions**.

Note

You can only add new pricing dimensions through the AWS Marketplace Management Portal. To modify or remove previously created dimensions, contact the [AWS Marketplace Seller Operations team](#) with your request. In your request, include the product ID and details about what dimensions you want to change or remove.

4. Add any new pricing dimensions that you want. For more information about the pricing fields, see [Providing details for a professional services product](#).
5. Select **Submit** to create the request.
6. (Optional) From the **Products** menu of AWS Marketplace Management Portal, select **Professional services**, then choose the **Requests** tab. Verify that you see your request with the correct **Product title**, and that the **Request status** is **Under review**. Your product will be updated with the changes you requested within a few minutes. If there is an error, you can view it here and resubmit your edits after fixing the errors.

Editing product visibility

By default, products are created with limited visibility—a new product is only visible from your account. You can add other test accounts, or make the product publicly visible in the AWS Marketplace. The following procedure describes how to edit the visibility of an existing professional services product in the AWS Marketplace Management Portal.

To edit product visibility

1. Open a web browser and sign into the [AWS Marketplace Management Portal](#).
2. From the **Products** menu, select **Professional services**. This page shows you all professional services products that you have already created.
3. Select an existing product that you would like to edit or update the visibility. Then, from the **Request changes** menu, select **Update product visibility**.
4. Select an option to update your product visibility and choose **Submit** to submit your request for review.
5. Verify that the **Requests** tab shows the **Request** status as **Under review**. When the request completes, the status becomes **Succeeded**. If the status is **Failed**, select the request name to view the **Issues Found**.

Note

To make a product visible in the public AWS Marketplace catalog requires a product review by the AWS Marketplace Seller Operations team to ensure that the product meets the product guidelines (see [Requirements for professional services products](#)). The request can take several days to complete.

Removing a professional services product

The following procedure describes how to remove an existing professional services product from the AWS Marketplace Management Portal.

Note

Removing a professional services product would not affect active private offers.

To remove a product

1. Open a web browser and sign into the [AWS Marketplace Management Portal](#).
2. From the **Products** menu, select **Professional services**. This page shows you all professional services products that you have already created.
3. Select an existing product that you would like to remove. Then, from the **Request changes** menu, select **Update product visibility**.
4. Select **Restricted** as the visibility option to update your product visibility and choose **Submit**.
5. Verify that the **Requests** tab shows the **Request** status as **Under review**. When the request completes, the status becomes **Succeeded**.

Note

The request can take several days to complete. Products with active offers will be moved to restricted state until the last active subscription or contract is completed and then removed from AWS Marketplace. Restricted status means that existing users can continue to use the product. However, the product will no longer be visible to the public or be available to new users.

Providing details for a professional services product

When you publish a professional services product on AWS Marketplace, you must provide the product metadata. This topic discusses information that is useful when you prepare or edit your product's details.

Note

For information about guidelines and requirements for professional services products, see [Requirements for professional services products](#).

Topics

- [Product descriptions](#)
- [Additional resources](#)
- [Support information](#)

- [Pricing dimensions](#)
- [Product visibility](#)

Product descriptions

The product descriptions section in the product details is the core of your product. It describes your product to your potential buyers so that they can make a purchasing decision. This section of the product details includes the following data:

- **Product title** – The name of your product. This is used to identify your product; it's visible on the product page and within search results. Provide a meaningful name for your product. It must be unique within AWS Marketplace.
- **SKU** – (Optional) Used to track your products on AWS Marketplace. This information is for your own use; buyers don't see it.
- **Short description** – A concise description of your product that appears on the tiles and underneath the product title in the AWS Marketplace product catalog.
- **Long description** – A longer, formatted description that describes the details of your product to buyers. List the product features, benefits, usage, and other information specific to the product. Use the available formatting to make the information easier to understand and scan.
- **Product logo** – This field is a public S3 URL that points to an image file that represents your product. The file must be in .png, .jpg, or .gif format, with a transparent or white background, under 5MB, and be between 110-10,000 pixels wide and tall. The logo is uploaded during product submission, and stored in AWS Marketplace. Modifying the contents of the URL will not modify the logo in AWS Marketplace after it is submitted.

Note

The S3 URL that you provide must be publicly available. This is a property of the S3 bucket in which the file resides. For more information, see [How do I edit public access settings for S3 buckets?](#) in the Amazon Simple Storage Service Console User Guide.

- **Highlights** – A set of one to three short points about your product, describing its key features or differentiators. At least one highlight is required.
- **Product categories** – The types of service that you provide. You must choose at least one, and up to three, categories. There are many categories to choose from, but professional services products must include at least one of the following:

Assessment

Evaluation of the customer's current operating environment to find the right solutions for their organization.

Implementation

Help with configuration, setup, and deployment of third-party software.

Premium support

Access to guidance and assistance from experts, designed for the customer's needs.

Managed services

End-to-end environment management on the customer's behalf.

Training

Tailored workshops, programs, and educational tools provided by experts to help the customer employees learn best practices.

- **Keywords for search results** – Provide up to three keywords that buyers might use to search for your product. You can list keywords in a comma-separated list, up to 250 characters.
- **Associated products** – *optional* – Include at least one public product from AWS Marketplace that your service either works with or supports. AWS Marketplace uses these products as input when selecting products to show on your product's details page or in **Related products** for those products.

Additional resources

In the **Additional resources** section of the product details, you can provide links to resources that you have created to help your customers. This is an optional set of one to three downloadable resources that are stored online. Examples of resources include product information sheets, whitepapers, or product manuals. For each resource, provide a name and a URL for the resource.

Support information

This section is a formatted text field that allows you to describe the support that you provide for your service.

Customers expect support on issues such as using the services, troubleshooting, and requesting refunds (if applicable). The support description should contain a statement about the level of

support a customer can expect. Consider including support details for both pre-purchase questions and post-purchase issues.

Pricing dimensions

Pricing dimensions for professional services are packages that you offer. For example you might offer *Silver*, *Gold*, and *Platinum* support. Or you might offer 10, 20, or 50 hours of consulting. For each dimension you want to offer (at least one, up to 24), specify a name and a description. When you create a private offer for the product by working with a buyer directly, you set the actual prices for these dimensions.

Note

For information about how pricing dimensions are used, and how prices are set, see [Creating private offers](#).

Product visibility

Released products can be visible in AWS Marketplace to just your own account, to a small set of test accounts, or to all AWS accounts. By default, the product is published in private release. To change the product visibility, see [Editing product visibility](#).

Requirements for professional services products

AWS Marketplace maintains requirements for all products and offerings on AWS Marketplace. These requirements help to promote a safe, secure, and trustworthy curated digital catalog for our customers. We encourage sellers to review the implementation of additional controls and protocols as applicable to meet the needs of their specific products.

AWS Marketplace reviews all products and their related metadata when submitted to ensure that they meet or exceed current AWS Marketplace requirements. We review and adjust these requirements to meet our security requirements. In addition, AWS Marketplace continuously reviews products to verify that they meet any changes to these requirements. If products fall out of compliance, AWS Marketplace contacts you to update your product. In some cases, your products might temporarily be unavailable to new subscribers until issues are resolved.

Product setup guidelines

All professional services products must adhere to the following product setup guidelines:

- All pricing dimensions must relate to actual services offered and billed using AWS Marketplace.
- Your products must be listed in one of these categories: *Assessments, Implementation, Managed services, Premium support, or Training*.
- Besides the required professional services categories, your product should also be correctly categorized by choosing other appropriate categories that fit into services offered.
- Your product's logo must not be designed to confuse with the AWS logo, or any logo from an unrelated third party.
- Your product details must not contain offensive or explicit material. They must comply with the AWS Acceptable Use Policy available at <https://aws.amazon.com/aup/>.
- Your professional service product must directly support or offer services related to third-party software products listed on AWS Marketplace or help customers achieve specific outcomes related to the adoption or management of their AWS Cloud.

Customer information requirements

All professional services products must adhere to the following customer information requirements:

- Professional services products must be billed entirely through the listed dimensions on AWS Marketplace.
- You are not permitted to collect customer payment information for your professional services product listing on AWS Marketplace at any time, including credit card and bank account information.
- Any subscriber or prospective subscriber information provided by AWS to you in conjunction with your professional services products must be used solely in accordance with the Terms and Conditions for AWS Marketplace Sellers.

Product usage guidelines

All professional services products must adhere to the following product usage guidelines:

- After a customer contacts you through the professional service listing, you should contact them within two business days. After a customer accepts a private offer, you must contact them or provide them with next steps within two days unless otherwise outlined in the private offer.
- Customers must be able to easily get help with issues, such as using the services, troubleshooting, and requesting refunds (if applicable). Support contact options must be specified on the fulfillment landing page. The support description should contain a statement about the level of support a customer can expect.
- Your product's overview should include clear value propositions, key features, links to detailed documentation, and clear definitions of pre-purchase and post-purchase support of the services offered.
- Your products should have clear and straightforward service terms.

Architecture guidelines

All professional services products must adhere to the following architecture guidelines:

- Professional services products must be related to an AWS service or at least one public AWS Marketplace product (up to four) in which the product offers services for those related products directly or offers services that drive more subscribers to those related products.
- If the services offered require additional resources in the customer's infrastructure, follow these guidelines:
 - Provision resources in a secure way, such as by using the AWS Security Token Service or AWS Identity and Access Management (IAM).
 - Provide additional documentation including a description of all provisioned AWS services, IAM policy statements, and how an IAM role or user is deployed and used in the customer account.
 - Include a notification in the product description that explains that if the customer incurs additional AWS infrastructure costs, separate from their AWS Marketplace transaction, they're responsible for paying additional infrastructure charges.
 - If your product deploys an agent, provide instructions to the customer that describe how to deploy the agent in their AWS account.

Professional services product pricing

You can set the following product pricing model for your professional services products:

- **Private offers** - With seller private offers, there are options available for multi-year and custom duration contracts. For more information about multi-year and custom duration contracts, see [Preparing your private offer](#) and [Installment plans](#).

You can set only one price per product.

For more information about pricing AWS Marketplace products, see [Product pricing](#).

Data products

AWS Data Exchange is a service that makes it easy for AWS customers to securely exchange file-based data sets in the AWS Cloud. As a provider, AWS Data Exchange eliminates the need to build and maintain any data delivery, entitlement, or billing technology. Providers in AWS Data Exchange have a secure, transparent, and reliable channel to reach AWS customers and grant existing customers their subscriptions more efficiently. The process for becoming an AWS Data Exchange provider requires a few steps to determine eligibility.

A data product has the following parts:

- **Product details** – This information helps potential subscribers understand what the product is. This includes a name, descriptions (both short and long), a logo image, and support contact information. Product details are filled out by providers.
- **Product offers** – In order to make a product available on AWS Data Exchange, providers must define a public offer. This includes the prices and durations, data subscription agreement, refund policy, and the option to create custom offers.
- **Data sets** – A product can contain one or more data sets. A data set is a dynamic set of file-based data content. Data sets are dynamic and are versioned using revisions. Each revision can contain multiple assets.

For more information, including eligibility requirements, see [Providing data products on AWS Data Exchange](#) in the *AWS Data Exchange User Guide*.

Submitting your product for publication

You use the product submission process to make your products available on AWS Marketplace. Products can be quite simple, for example, a single Amazon Machine Image (AMI) that has one price structure. Or, products can be quite complicated, with multiple AMIs, AWS CloudFormation templates, and complex pricing options and payment schedules. You define your product offering and submit it through the AWS Marketplace Management Portal in one of two ways:

- Using the **Products** tab – For products that are less complex, you use the **Products** tab to completely define and submit your request.
- Using the **Assets** tab – For products that are more complex and require more definition, you download a product load form (PLF), add product details, and then upload the completed form using the **File upload** option.

Note

Data product providers must use the AWS Data Exchange console to publish products. For more information, see [Publishing a new product](#) in the *AWS Data Exchange User Guide*.

We recommend that you start by using the **Products** tab to determine which approach to use. The following table lists configurations and the approach you use to submit your request. The first column is the pricing model for your product, and the other three columns are how the product is deployed to the customer.

Pricing model	Products launched using single-node AMI	Products launched with AWS CloudFormation	Products launched as software as a service (SaaS)
Bring Your Own License (BYOL)	Products tab	Assets tab	
Free	Products tab	Assets tab	
Paid Hourly	Products tab	Assets tab	

Pricing model	Products launched using single-node AMI	Products launched with AWS CloudFormation	Products launched as software as a service (SaaS)
Paid Hourly with Annual	Products tab	Assets tab	
Paid Monthly	Products tab	Assets tab	
Hourly with Monthly	Assets tab	Assets tab	
Paid Usage (AWS Marketplace Metering Service)	Products tab	Assets tab	
Contract Pricing	Products tab		
SaaS Subscription			Products tab
SaaS Contract			Products tab
SaaS Legacy			Assets tab

You can submit products individually or, if you use a product load form, you can submit multiple products or product updates at the same time. You cannot submit multiple products at the same time using the **Products** tab. If you are unclear on what products can be submitted in what manner, start by using the **Products** tab. If you have any problems making your submissions, contact the [AWS Marketplace Seller Operations](#) team.

Using the Products tab

To access the **Products** tab, log in to the AWS Marketplace Management Portal. From the **Products** tab, choose either **Server**, **SaaS**, or **Machine learning**, depending on the type of product you are managing. A dashboard for that product type appears that contains all of your current products. If you choose the **Requests** tab, the dashboard displays any outstanding requests you have and your completed request history. Once you start creating a new product request, you can save your work in progress, and if necessary, create your request in several different sessions.

When you are ready to submit your product request, the request is reviewed by the AWS Marketplace team. You can monitor the status of your request on the product page for the type of product you are requesting. For new products, after your request is approved for publication, you receive a limited listing URL that you can use to preview and approve your submission. Your product offer is not published until you approve the submission. When you request an update to an existing product, the update is published without the need for you to review and approve the change. This includes adding or removing versions, and metadata changes.

You track the status of your requests under the **Requests** tab. The status will be one of the following:

- **Draft** – You have started the request process but have not submitted your request.
- **Submitted** – You have completed and submitted your request, and it is under review.
- **Action Required** – The AWS Marketplace team has reviewed your request and needs more information.
- **Approval Required** – The AWS Marketplace team has created the limited listing URL for your product. You must review and either approve or reject the URL before AWS Marketplace will publish. If you approve, the status changes to **Publishing Pending** while the site gets published. If you reject, the status returns to **Draft** so you can modify the request.
- **Publishing Pending** – You have approved the mock-up of your request and AWS Marketplace is publishing your product.
- **Expired** – You started the request process but did not complete it within six months, so the request expired.

If you have an entry with a status of **Submitted**, you can retract the submission. If you have an entry with a status of **Draft**, you can delete the request. This will allow you to start over. When you delete a **Draft** entry, the entry is moved to the **Request History** tab.

To add your product in the AWS GovCloud (US) AWS Region, you must [have an active AWS GovCloud \(US\) account](#) and comply with the AWS GovCloud (US) requirements, including export control requirements.

Company and product logo requirements

Your company logo and the logo for your products must conform to the following AWS Marketplace guidelines so that the user experience is uniform when browsing AWS Marketplace:

Product logo specifications – Your product logo image should have a transparent or white background and be 120 to 640 pixels in size, with a 1:1 or 2:1 (wide) ratio.

Company logo specifications – Your company logo image should have a transparent background and be 220 x 220 pixels in size, allowing for 10 pixels of padding on each side within.

Requirements for submitting paid repackaged software

If you are submitting a paid listing of either a repackaged open-source software (for example, open source AMI or container products with paid support), or software that was originally created by a vendor other than you (for example, reselling an AMI with Windows operating system), the following requirements must be met before submission:

- The product title must indicate the value added by your repackaging. Examples of product titles include: *Hardened <Product>*, *<Product> with added packages*, *<Product1> on <Product2>*.
- The product title must not contain any other language that is not otherwise supported with documentation. For example, the product title may not use the words *certified*, *original*, or *free* unless these are substantiated in the product details that you provide.
- The product short description must include a clear statement summarizing the product charges. The short description must begin with the phrase *This product has charges associated with it for...* For example, if a product includes charges for support from the seller, then the product description should state: *This product has charges associated with it for seller support*.
- The product logo must be same as the company logo which was used during your seller registration process. The product logo can differ from your company logo only if you use the official software logo, whereby you must receive explicit permission from the original software vendor. If explicit permission is obtained, a link to that documentation must be included in the notes section of the change request (or in the **Enter a brief description** field of the **File Uploads** page when using the product load form).
- For AMI products, the AMI name must not be reused from the original product. The AMI name must begin with the seller name and follow this format: [Seller Name] [name-given-to-ami].

If the paid listing is for a standalone software product that was not created by your company and there is no intellectual property added to the product (for example, bundling additional software libraries or adding special configuration) then, along with the earlier requirements, the following requirements must also be met:

- Product title must include the seller name (along with the value added, as described earlier). The seller name is the name used during seller registration. For example, *<Product> with maintenance support by <seller>*.
- The first line of the product's long description must begin with the phrase *This is a repackaged software product wherein additional charges apply for...* (or, if it's open source, *This is a repackaged open source software product wherein additional charges apply for...*). Then, the long description must include a clear statement summarizing what you are charging for, as well as additional details describing those features. For example, the long description of an open source product charging for additional support might start as: *This is a repackaged open source software product wherein additional charges apply for support with {SLA Details}*.

Requirements for products with a hardware component

The sale of hardware products isn't permitted on AWS Marketplace. If you're submitting a software product that requires a hardware component (for example, a SIM card, smart device, IoT device, or sensor), you must meet the following requirements:

- The hardware component can't be sold on AWS Marketplace.
- The cost of the hardware component can't be included in the listing price of your product.
- The **Product Overview** section of the listing must include the following statements: *Any hardware that may be required with this listing must be obtained separately. Review the product details for more information.*

AWS CloudFormation-launched product (free or paid) or usage-based paid AMI product

Use a product load form (PLF) to submit products that AWS Marketplace customers launch by using AWS CloudFormation templates. The PLF is available through the AWS Marketplace Management Portal (AMMP).

Submitting your product

1. From the [AMMP](#), download the product load form (PLF) for your product.

2. Add your product definition, which includes product information (title, description, highlights), technical information (AMI_ID, Regions, instance types, OS), and pricing details (pricing model, Free Trial).
3. Submit your PLF following the instructions under the Instructions table of the spreadsheet.

The AWS Marketplace team reviews your product for policy and security compliance, software vulnerabilities, and product usability. If there are any questions or issues with a request, the AWS Marketplace team will contact you through an email message to discuss your request. Once approved, a mock-up of your product's page is created. After you review the page, you accept or reject the mock-up. Once approved, we add the page to the AWS Marketplace.

Updating your product

For products that you created by using the product load form (PLF), you also use the PLF to make changes to those products. You can make changes to the original PLF you completed or, if it's not available, you can start with a new PLF. Just like using the **Products** tab, you can add a new version, remove existing versions, and update pricing, instance types, Region availability, and metadata. To make an update, you prepare any updated product the same way you prepare a new product. After the product update is prepared, follow these steps:

1. Use your existing PLF or, from the [AWS Marketplace Management Portal](#), under the **Assets** tab, choose **File upload**. Under **Product load forms and seller guides**, you can download the PLF for your product.
2. Update your product submission in the PLF.
3. From the [AWS Marketplace Management Portal](#), under the **Assets** tab, choose **File Upload**.
4. On the **File Uploads** page, upload your updated PLF and any AWS CloudFormation templates. The file uploader provides a secure transfer mechanism and a history of submitted files. The uploader automatically notifies the AWS Marketplace team to begin processing your request. Include a description of the submission (adding new version, changing price, changing metadata, and so forth).

Your product submission is reviewed for policy and security compliance, software vulnerabilities, and product usability. If there are any questions or issues with a request, the AWS Marketplace team will contact you through an email message. Updates to existing product pages are processed and released directly without additional reviews.

Product changes and updates

Sellers can submit changes to their product at any time, and they will be processed as described earlier. However, some changes can only be made every 90 or 120 days, or when pending changes are in place. Examples include price changes and AWS Region or instance type changes. Common changes include:

- **New Version** – New versions of the software and rollouts of patches or updates. At your request, we can notify customers who have subscribed to your AWS Marketplace content about the availability of new versions or send upgrade instructions on your behalf.
- **Metadata change** – Changes to product information (Description, URLs, and Usage Instructions).
- **Pricing Change** – A change to the pricing amount. A notification to current customers is sent after the request is complete. Once the notification is sent, the price change will take effect on the first of the month following a 90-day window. For example, if you make a change on March 16, 90 days after would be approximately June 16, but the price change happens on the first of the following month. The actual date of the change would be July 1.
- **Pricing Model Change** – A change to the pricing model (for example, Hourly, Free, Hourly_Annual). Not all pricing model changes are supported, and all requests to change models must be reviewed and approved by the AWS Marketplace team. Any change from a free to a paid model presents significant impact to existing customers. An alternative is to propose a new product with additional features and encourage current customers to migrate.
- **Region or Instance change** – Adding or removing instances types or Regions.
- **Product takedown** - Remove a product page from AWS Marketplace to prevent new customers from subscribing. A notification to current customers is sent after the request is complete.

Timing and expectations

While we strive to process requests as quickly as possible, requests can require multiple iterations and review by the seller and the AWS Marketplace team. Use the following as guidance for how long it will take to complete the process:

- Total request time normally takes 2–4 weeks of calendar time. More complex requests or products can take longer, due to multiple iterations and adjustments to product metadata and software.
- We require a completed product request and AMI at least 45 days in advance of any planned events or releases, so we can prioritize the request accordingly.

If you have any questions about your request, contact the [AWS Marketplace Seller Operations](#) team.

Submitting AMIs to AWS Marketplace

All AMIs built and submitted to AWS Marketplace must adhere to all product policies. We suggest a few final checks of your AMI prior to submission:

- Remove all user credentials from the system; for example, all default passwords, authorization keys, key pairs, security keys or other credentials.
- Ensure that root login is disabled or locked. Only sudo access accounts are allowed.
- If you are submitting an AMI to be deployed into the AWS GovCloud (US) Region, you need to [have an active AWS GovCloud account](#) and agree to the [AWS GovCloud Requirements](#), including applicable export control requirements.

AMI self-service scanning

Self-service AMI scanning is available within the AWS Marketplace Management Portal. With this feature, you can initiate scans of your AMIs and receive scanning results quickly—typically in less than an hour—with clear feedback in a single location.

To begin sharing and scanning your AMI with self-service scanning

1. Navigate to <https://aws.amazon.com/marketplace/management/manage-products/>.
2. Select the AMI to share.
3. View your scan results.

After your AMI has successfully been scanned, you can follow the current process to submit it to the AWS Marketplace Seller Operations team by [uploading](#) your product load form (PLF). If you have any issues, contact the [AWS Marketplace Seller Operations](#) team.

To include your AMI in the self-service scanning list, the AMI must be in the us-east-1 (N. Virginia) Region and owned by your AWS Marketplace seller account. If you need to grant other accounts access to the AWS Marketplace Management Portal, you must register those accounts as sellers. For more information, see [Seller registration process](#).

AMI cloning and product code assignment

After your AMI is submitted, AWS Marketplace creates cloned AMIs for each Region that you have indicated that software should be available in. During this cloning and publishing process, AWS Marketplace attaches a product code to the cloned AMIs. The product code is used to both control access and to meter usage. All submissions must go through this AMI cloning process.

Final checklist

To help avoid delays in publishing your product, use this checklist before you submit your product request.

Product usage

- Production-ready.
- Does not restrict product usage by time or other restrictions.
- Compatible with 1-click fulfillment experience.
- Everything required to use the product is contained within the software, including client applications.
- Default user uses a randomized password and/or creation of initial user requires verification that the buyer is authorized to use the instance using a value unique to the instance such as instance ID.

For free or paid products

- No additional license is required to use the product.
- Paid repackaged software meets the AWS Marketplace [Requirements for submitting paid repackaged software](#).
- Buyer does not have to provide personally identifiable information (for example, an email address) to use the product.

AMI preparation

- Use hardware virtual machine (HVM) virtualization and 64-bit architecture.
- Does not contain any known vulnerabilities, malware, or viruses.
- Buyers have operating system-level administration access to the AMI.

- Run your AMI through AMI Self-Service Scanning.

For Windows AMIs

- Use the most recent version of Ec2ConfigService, as described in [Configuring a Windows instance using the EC2Config service](#) in the *Amazon EC2 User Guide for Windows Instances*.
- The Ec2SetPassword, Ec2WindowsActivate, and Ec2HandleUserData plugins are enabled, as described in [Configuring a Windows instance using the EC2Config service](#) in the *Amazon EC2 User Guide for Windows Instances*.
- No Guest Accounts or Remote Desktop Users are present.

For Linux AMIs

- Root login is locked and disabled.
- No authorized keys, default passwords, or other credentials are included.
- All required fields are completed.
- All values are within specified character limits.
- All URLs load without error.
- Product image is at least 110px wide and between a 1:1 and 2:1 ratio.
- Pricing is specified for all enabled instance types (for hourly, hourly_monthly, and hourly_annual pricing models).
- Monthly pricing is specified (for hourly_monthly and monthly pricing models).

If you have any questions or comments about automated AMI building, contact the [AWS Marketplace Seller Operations](#) team.

Marketing your product

You can contribute to the success of your product by driving awareness of AWS Marketplace and by driving traffic directly to your product pages on AWS Marketplace. The following provides information and support to help you market the product or products that you have listed on AWS Marketplace.

180-day GTM Academy

The 180-day GTM Academy is available to all AWS Marketplace sellers, and it provides self-service go-to-market (GTM) resources to help you build, activate, and track demand generation campaigns for your offering in AWS Marketplace. You can:

- Fast track demand for your listings.
- Improve marketing return on investment and enhance customer messaging by integrating AWS and AWS Marketplace value proposition into your messaging.
- You can progress toward or within the [AWS Marketplace Go-to-Market Program Guide](#).

You can access the 180-day GTM Academy from the **Marketplace Resources** section of the [AWS Marketplace Management Portal](#).

Announcing your product's availability

We encourage you to broadly announce the availability of your product on AWS Marketplace. You can do this via press releases, tweets, blogs, or any other preferred media channels. We have provided sample text that you can include, along with guidelines and instructions for using our trademarks and issuing press releases.

We will review your blogs, tweets, and other non-press release announcements before going public to ensure consistency with AWS messaging and brand guidelines or voice. Submit your request for review to your AWS account manager. The review takes up to 10 business days to complete. Notify us when you post any tweets, blogs, or press releases, and we will do our best to repost to increase their visibility.

AWS Marketplace messaging

In your customer communications you might want to describe the purpose, goals, and benefits of purchasing your product using AWS Marketplace. Use the following messaging when referring to AWS Marketplace.

What is AWS Marketplace?

AWS Marketplace is an online store that makes it easy for customers to find, compare, and immediately start using the software and services that run on AWS. Visitors to AWS Marketplace can use 1-Click deployment to quickly launch preconfigured software and pay only for what they use, by the hour or month. AWS handles billing and payments, and software charges appear on the customer's AWS bill.

Why would a customer shop on AWS Marketplace?

Finding and deploying software can be challenging. AWS Marketplace features a wide selection of commercial and free IT and business software, including software infrastructure such as databases and application servers, IoT solutions, developer tools, and business applications, from popular sellers. AWS Marketplace enables customers to compare options, read reviews, and quickly find the software they want. Then they can deploy it to their own Amazon Elastic Compute Cloud instance using 1-Click or using the AWS Marketplace Management Portal.

Software prices are clearly posted on the website and customers can purchase most software immediately, with payment instruments already on file with Amazon Web Services. Software charges appear on the same monthly bill as AWS infrastructure charges.

Why would software or SaaS sellers sell on AWS Marketplace?

With AWS Marketplace, software and software as a service (SaaS) sellers with offerings that run on AWS can benefit from increased customer awareness, simplified deployment, and automated billing.

AWS Marketplace helps software and SaaS sellers of software and services that run on AWS find new customers by exposing their products to some of the hundreds of thousands of AWS customers, ranging from individual software developers to large enterprises.

Selling on AWS Marketplace enables independent software vendors (ISVs) to add hourly billing for their software without undertaking costly code changes. They simply upload an Amazon Machine Image (AMI) to AWS and provide the hourly cost. Billing is managed by AWS Marketplace, relieving

sellers of the responsibility of metering usage, managing customer accounts, and processing payments, leaving software developers more time to focus on building great software.

Additionally, customers benefit from the ability to easily deploy preconfigured images of the software, simplifying onboarding for new customers.

Reviews on AWS Marketplace

AWS Marketplace provides the ability for customers to submit reviews on your product. We also provide the ability for syndicated reviewers such as G2, a business-to-business marketplace that curates independent product reviews, to integrate their syndicated reviews on AWS Marketplace.

AWS Marketplace customer reviews must meet the review guidelines listed in the user guide for buyers. Review submissions are not released on AWS Marketplace until after the submission is reviewed to verify it meets our review criteria. For more information on review guidelines, see [Product Reviews](#). Syndicated review organizations use their own unique processes to validate their reviews and aren't reviewed by AWS Marketplace before release. If you think that a syndicated review on your product doesn't meet the product review guidelines, or a review on your product contains objectionable content, contact the [AWS Marketplace Seller Operations](#) team.

The reviewer can also provide a star rating for your product based on a five-star rating system. The ratings are averaged to give the overall star rating for your product. Syndicated reviews can also include a star rating, but star ratings from syndicated reviews are not averaged in with the AWS customer star ratings.

The following are additional key points about the product review feature:

- You can't have a product review removed from AWS Marketplace. However, you can leave a comment on any review as long as the comment meets the review criteria governing product reviews.
- If you think that a review doesn't meet the review guidelines or contains objectionable content, you can contact the [AWS Marketplace Seller Operations](#) team and describe your concern.
- AWS customers searching for products in AWS Marketplace can search and filter results based on ratings, verified reviews, and externally sourced reviews. AWS customers see the externally sourced ratings alongside AWS customer ratings in search results.
- Syndicated reviews for your product are automatically added to AWS Marketplace at no cost to you. Because reviews are automatically added, you don't need to submit a request to have a syndicated review added.

- If you don't have any syndicated reviews for your product, you can contact the syndicated reviewer and follow their process for getting your product reviewed. For example, with G2, you can visit their website and claim your product page to start their review process.

Linking to AWS Marketplace

Your company likely has a web presence where it describes and promotes your product. We encourage you to highlight that the product is available to run on AWS and can be purchased using AWS Marketplace. To simplify the process for your customers to discover and deploy your software, we have provided instructions for linking your customers to your product.

Using the AWS Marketplace logo

The AWS Marketplace logo is a way to easily tell your customers that your software runs on AWS and is available in AWS Marketplace. If you would like to promote your software in AWS Marketplace, [download the archived folder \(.zip file\)](#), which contains multiple color treatments and file formats. If you have questions regarding your obligations under the AWS Trademark Guidelines, see [AWS Trademark Guidelines & License Terms](#).

Linking directly to your product on AWS Marketplace

You can send your customers directly to the product's information page on AWS Marketplace by including deep links on your website or collateral. Use the following example link structure for browser-based linking.

```
https://aws.amazon.com/marketplace/pp/ASIN
```

Replace ASIN segment of the URL with your product's ASIN.

Example

```
https://aws.amazon.com/marketplace/pp/B00635Y2IW
```

The ASIN appears in the URL when you search for your application on aws.amazon.com/marketplace. Alternatively, you can consult with your account manager to find the ASIN.

Note

Test the links before using them to make sure that they direct your customers to the correct page.

Press releases

We encourage you to announce your product's availability on AWS Marketplace through any channel you prefer. However, all press releases that reference AWS Marketplace must be reviewed and signed off on by Amazon before any publication or announcement is made. While we encourage you to make announcements, we can't support joint press releases with AWS Marketplace sellers. We will, on a case-by-case basis, support press releases with a quote from AWS. The quote must meet several conditions, including but not limited to: it announces a new product or service listed on AWS Marketplace or it includes a customer reference that uses AWS Marketplace.

All press releases must be drafted by you. We suggest the following headline: [Insert product name] Now Available on AWS Marketplace. Use the messaging in this document for consistency.

The press release **should**:

- Clearly and accurately describe how the announcement relates to Amazon.com
- Clarify your role on AWS and with customers
- Be customer-focused and emphasize the customer benefit

The press release **should not**:

- Use the terms *partners*, *partnership*, or *alliance* to describe the relationship. We prefer *agreement*, *teamed*, or *relationship*.
- Include a quote from an Amazon Web Services executive unless previously agreed upon.
- Include any sales projections or use .com by the merchant unless referring to the website in your company boilerplate.
- Refer to your organization as an associate of Amazon.com because this could be confused with Amazon Associates, our online affiliate program.
- Disclose proprietary information about Amazon.com or refer to our stock ticker symbol.

Have your press release reviewed by submitting it in text format to your account manager. Additionally, review the [Amazon Web Services trademark guidelines](#) before using any AWS trademarks. Guidelines specific to the AWS Marketplace trademark are in the following section.

AWS Marketplace trademark usage guidelines

These Guidelines apply to your use of the AWS Marketplace logo and trademark, (each the “Trademark” and collectively the “Trademarks”) in materials that have been approved in advance by Amazon.com, Inc. and/or its affiliates (“Amazon”). Strict compliance with these Guidelines is required at all times, and any use of a Trademark in violation of these Guidelines will automatically terminate any license related to your use of the Trademarks.

1. You may use the Trademark solely for the purpose expressly authorized by Amazon and your use must: (i) comply with the most up-to-date version of all agreement(s) with Amazon regarding your use of any of the Trademarks (collectively “Agreements”); (ii) comply with the most up-to-date version of these Guidelines; and (iii) comply with any other terms, conditions, or policies that Amazon may issue from time to time that apply to the use of the Trademark.
2. We will supply an approved Trademark image for you to use. You may not alter the Trademark in any manner, including but not limited to, changing the proportion, color, or font of the Trademark, or adding or removing any element(s) from the Trademark.
3. You may not use the Trademark in any manner that implies sponsorship or endorsement by Amazon other than by using the Trademark as specifically authorized under the Agreements.
4. You may not use the Trademark to disparage Amazon, its products or services, or in a manner which, in Amazon’s sole discretion, may diminish or otherwise damage or tarnish Amazon’s goodwill in the Trademark.
5. The Trademark must appear by itself, with reasonable spacing between each side of the Trademark and other visual, graphic or textual elements. Under no circumstance should the Trademark be placed on any background which interferes with the readability or display of the Trademark.
6. You must include the following statement in any materials that display the Trademark: “AWS Marketplace and the AWS Marketplace logo are trademarks of Amazon.com, Inc. or its affiliates.
7. You acknowledge that all rights to the Trademark are the exclusive property of Amazon, and all goodwill generated through your use of the Trademark will inure to the benefit of Amazon. You will not take any action that is in conflict with Amazon’s rights in, or ownership of, the Trademark.

Amazon reserves the right, exercisable at its sole discretion, to modify these Guidelines and/or the approved Trademarks at any time and to take appropriate action against any use without permission or any use that does not conform to these Guidelines. If you have questions about these Guidelines, contact trademarks@amazon.com for assistance or write to us at the following address:

Amazon.com, Inc., Attention: Trademarks

PO Box 81226

Seattle, WA 98108-1226

Seller notifications for AWS Marketplace events

AWS Marketplace provides timely notifications through email, Amazon EventBridge events, and Amazon Simple Notification Service (Amazon SNS) topics.

Topics

- [Email notifications for AWS Marketplace events](#)
- [Amazon EventBridge events](#)
- [Amazon Simple Notification Service notifications for AWS Marketplace events](#)

Email notifications for AWS Marketplace events

AWS Marketplace uses the root user account to send automated email messages. The automated email messages are sent to the email address associated with your AWS account. These email messages provide you with visibility into events on AWS Marketplace and are sent automatically.

Note

You can add custom email aliases for notifications. For more information, see [the section called "Adding or updating email addresses"](#).

AWS Marketplace sends email notifications that verify the transaction for offers and agreements made in AWS Marketplace. The notifications are sent in real time based on the successful fulfillment of a buyer's subscription.

As a seller, you receive an email notification when a buyer accepts an offer. Notifications are sent to buyers and independent software vendors (ISVs) for public subscriptions. They're sent to buyers, ISVs, and channel partners for channel partner private offer subscriptions.

When sent to an ISV or a channel partner, email notifications contain the following details:

- Purchase date, time, and time zone
- Customer AWS account ID
- Product name

- Product identification
- Offer name
- Offer identification
- Agreement identification
- Service start date
- Service end date
- Purchase amount (for contract and channel partner)

Note

Certain email providers (for example, Google or Yahoo) may filter out your AWS Marketplace notification emails. If you haven't received notifications from AWS Marketplace, or if you see them in your spam folder, adjust your email settings. For example, see [Google Group instructions](#) or [Yahoo instructions](#).

The following topics describe the event types that are supported by email notifications and how to manage notifications.

Event types

The following event types are supported by email notifications for all products and pricing types:

- Buyer has requested a professional service product
- Recurring scan vulnerability or recurring scan reminder
- Reseller opportunity has been created, updated, or expired
- New or updated private offer has been published
- New or updated channel partner private offer has been published
- Email notifications to buyer and seller for offer acceptance

Manage notifications

The following topics explain how to manage email notifications for events.

Adding or updating email addresses

You can add up to 10 email addresses for custom email notifications using the AWS Marketplace Management Portal.

To add or update email addresses

1. Sign in to the [AWS Marketplace Management Portal](#).
2. From **Settings**, choose the **Notifications** tab.
3. Under **Email for custom notifications**, choose **Add email address**.
4. For **Recipient details**, enter a custom email address in the **Email address** field.
5. (Optional) Choose **Add new recipients** to add another email address (up to 10 total).
6. Choose **Submit**.

Unsubscribing recipients from notifications

You can remove an email address so the recipient is unsubscribed from custom email notifications.

To unsubscribe recipients from event notifications

1. Sign in to the [AWS Marketplace Management Portal](#).
2. From **Settings** choose the **Notifications** tab.
3. Under **Email for custom notifications**, choose **Update email address**.
4. For **Recipient details**, choose **Remove** to remove the email address.
5. Choose **Submit**.

The recipient will no longer receive email notifications for custom events.

Note

You can also unsubscribe using the link in the email.

Amazon EventBridge events

AWS Marketplace is integrated with Amazon EventBridge, formerly called Amazon CloudWatch Events. EventBridge is an event bus service that you can use to connect your applications with data from a variety of sources. For more information, see the [Amazon EventBridge User Guide](#).

As a seller, you receive an *event* from AWS Marketplace when an offer is created. The *event* contains details like the ID, expiration date, and product details.

Topics

- [AWS Marketplace Catalog API Amazon EventBridge events](#)

AWS Marketplace Catalog API Amazon EventBridge events

This topic provides detailed information about events under the Marketplace Catalog service in the EventBridge console.

Action by seller	Event received	Related topic
Independent software vendor (ISV) creates an offer and makes it available for purchase	Offer Released	the section called "Events for new offers"
ISV's product is used by a channel partner to create an offer	Offer Released	the section called "Events for new offers"
Channel partner creates an offer	Offer Released	the section called "Events for new offers"
Change set succeeds	Change Set Succeeded	the section called "Events for change sets"
Change set fails	Change Set Failed	the section called "Events for change sets"

Action by seller	Event received	Related topic
Change set is cancelled	Change Set Cancelled	the section called "Events for change sets"
Security vulnerabilities were detected on the ISV's product	Products Security Report Created	the section called "Events for security summary report"

Events for new offers

When sellers create an offer and make it available for purchase, they can receive an event with the following detail type: Offer Released.

Note

For information on creating EventBridge rules, see [Amazon EventBridge rules](#) in the *Amazon EventBridge User Guide*.

The following is an example event body for a new offer created by an ISV.

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-0123456789ab",
  "detail-type": "Offer Released",
  "source": "aws.marketplacecatalog",
  "account": "123456789012",
  "time": "2023-08-26T00:00:00Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:aws-marketplace:us-east-1:123456789012:AWSMarketplace/Offer/offer-1234567890123"
  ],
  "detail": {
    "requestId": "3d4c9f9b-b809-4f5e-9fac-a9ae98b05cbb",
    "catalog": "AWSMarketplace",
    "offer": {
      "id": "offer-1234567890123",
      "arn": "arn:aws:catalog:us-east-1:123456789012:Offer/offer-1234567890123",
      "name": "Offer Name",

```

```

    "expirationDate": "2025-08-26T00:00:00Z"
  },
  "product": {
    "id": "bbbbaaaa-abcd-1111-abcd-666666666666",
    "arn": "arn:aws:aws-marketplace:us-east-1:123456789012:AWSMarketplace/
SaaSProduct/bbbbbaaaa-abcd-1111-abcd-666666666666",
    "title": "Product Title"
  },
  "manufacturer": {
    "accountId": "123456789012",
    "name": "Manufacturer Account Name"
  },
  "sellerOfRecord": {
    "accountId": "123456789012",
    "name": "Seller Account Name"
  },
  "targetedBuyerAccountIds": [
    "999988887777",
    "111122223333"
  ]
}

```

The following is an example event body for when an ISV's product is used by a channel partner to create an offer.

```

{
  "version": "0",
  "id": "01234567-0123-0123-0123-0123456789ab",
  "detail-type": "Offer Released",
  "source": "aws.marketplacecatalog",
  "account": "123456789012",
  "time": "2023-08-26T00:00:00Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:aws-marketplace:us-east-1:987654321098:AWSMarketplace/Offer/
offer-1234567890123"
  ],
  "detail": {
    "requestId": "3d4c9f9b-b809-4f5e-9fac-a9ae98b05cbb",
    "catalog": "AWSMarketplace",
    "offer": {
      "id": "offer-1234567890123",

```

```

    "arn": "arn:aws:catalog:us-east-1:987654321098:Offer/offer-1234567890123",
    "name": "Offer Name",
    "expirationDate": "2025-08-26T00:00:00Z"
  },
  "product": {
    "id": "bbbbaaaa-abcd-1111-abcd-666666666666",
    "arn": "arn:aws:aws-marketplace:us-east-1:123456789012:AWSMarketplace/
SaaSProduct/bbbbbaaaa-abcd-1111-abcd-666666666666",
    "title": "Product Title"
  },
  "manufacturer": {
    "accountId": "123456789012",
    "name": "Manufacturer Account Name"
  },
  "sellerOfRecord": {
    "accountId": "987654321098",
    "name": "Seller Account Name"
  },
  "targetedBuyerAccountIds": ["999988887777", "111122223333"],
}
}
}

```

The following is an example event body for when a channel partner creates an offer.

```

{
  "version": "0",
  "id": "01234567-0123-0123-0123-0123456789ab",
  "detail-type": "Offer Released",
  "source": "aws.marketplacecatalog",
  "account": "987654321098",
  "time": "2023-08-26T00:00:00Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:aws-marketplace:us-east-1:987654321098:AWSMarketplace/Offer/
offer-1234567890123"
  ],
  "detail": {
    "requestId": "3d4c9f9b-b809-4f5e-9fac-a9ae98b05cbb",
    "catalog": "AWSMarketplace",
    "offer": {
      "id": "offer-1234567890123",
      "arn": "arn:aws:catalog:us-east-1:987654321098:Offer/offer-1234567890123",

```

```
    "name": "Offer Name",
    "expirationDate": "2025-08-26T00:00:00Z"
  },
  "product": {
    "id": "bbbbaaaa-abcd-1111-abcd-666666666666",
    "arn": "arn:aws:aws-marketplace:us-east-1:123456789012:AWSMarketplace/
SaaSProduct/bbbbbaaaa-abcd-1111-abcd-666666666666",
    "title": "Product Title"
  },
  "manufacturer": {
    "accountId": "123456789012",
    "name": "Manufacturer Account Name"
  },
  "sellerOfRecord": {
    "accountId": "987654321098",
    "name": "Seller Account Name"
  },
  "targetedBuyerAccountIds": ["999988887777", "111122223333"],
}
}
```

Events for change sets

When a change set completes, sellers, channel partners, and private marketplace administrators can receive an event. The AWS Marketplace Catalog API sends an event when a change set completes with a status of succeeded, failed, or cancelled. The source for these events is `aws.marketplacecatalog`, and the possible detail type values are `Change Set Succeeded`, `Change Set Failed`, and `Change Set Cancelled`.

Note

For information on change sets, see [Working with change sets](#) in the *AWS Marketplace Catalog API Reference*.

Each event contains change request details, such as the change set ID, change set name, event detail type, failure code (for failed requests), and start and end times of the request. This enables you to monitor your change sets without continuously querying the `DescribeChangeSet` action or checking the AWS Marketplace Management Portal for the status of your change requests.

Note

For information on creating EventBridge rules, see [Amazon EventBridge rules](#) in the *Amazon EventBridge User Guide*.

The following is an example event body for the Change Set Succeeded detail type.

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-0123456789ab",
  "detail-type": "Change Set Succeeded",
  "source": "aws.marketplacecatalog",
  "account": "123456789012",
  "time": "2022-11-01T13:12:22Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:aws-marketplace:us-east-1:123456789012:AWSMarketplace/
ChangeSet/76yesvf8y165pa4f98td2crtg"
  ],
  "detail": {
    "requestId" : "3d4c9f9b-b809-4f5e-9fac-a9ae98b05cbb",
    "Catalog": "AWSMarketplace",
    "ChangeSetId": "76yesvf8y165pa4f98td2crtg",
    "ChangeSetName": "Create my product",
    "StartTime": "2018-02-27T13:45:22Z",
    "EndTime": "2018-02-27T14:55:22Z"
  }
}
```

The following is an example event body for the Change Set Failed detail type.

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-0123456789ab",
  "detail-type": "Change Set Failed",
  "source": "aws.marketplacecatalog",
  "account": "123456789012",
  "time": "2022-11-01T13:12:22Z",
  "region": "us-east-1",
  "resources": [
```

```

    "arn:aws:aws-marketplace:us-east-1:123456789012:AWSMarketplace/
ChangeSet/76yesvf8y165pa4f98td2crtg"
  ],
  "detail": {
    "requestId" : "3d4c9f9b-b809-4f5e-9fac-a9ae98b05cbb",
    "Catalog": "AWSMarketplace",
    "ChangeSetId": "76yesvf8y165pa4f98td2crtg",
    "ChangeSetName": "Create my product",
    "StartTime": "2018-02-27T13:45:22Z",
    "EndTime": "2018-02-27T14:55:22Z",
    "FailureCode": "CLIENT_ERROR"
  }
}

```

The following is an example event body for the Change Set Cancelled detail type.

```

{
  "version": "0",
  "id": "01234567-0123-0123-0123-0123456789ab",
  "detail-type": "Change Set Cancelled",
  "source": "aws.marketplacecatalog",
  "account": "123456789012",
  "time": "2022-11-01T13:12:22Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:aws-marketplace:us-east-1:123456789012:AWSMarketplace/
ChangeSet/76yesvf8y165pa4f98td2crtg"
  ],
  "detail": {
    "requestId" : "3d4c9f9b-b809-4f5e-9fac-a9ae98b05cbb",
    "Catalog": "AWSMarketplace",
    "ChangeSetId": "76yesvf8y165pa4f98td2crtg",
    "ChangeSetName": "Create my product",
    "StartTime": "2018-02-27T13:45:22Z",
    "EndTime": "2018-02-27T14:55:22Z"
  }
}

```

Events for security summary report

When security vulnerabilities are detected on a seller's products, they can receive a summary report event and periodic reminders for outstanding product issues. The source for these events is `aws.marketplacecatalog`, and the detail type is `Products Security Report Created`.

Each event includes a summary of the count of products and versions with detected issues, a count of how many latest versions are affected, and the date when resolution is required to prevent a temporary restriction of these products or versions.

Note

For information on creating EventBridge rules, see [Amazon EventBridge rules](#) in the *Amazon EventBridge User Guide*.

For details on managing security events, see the [How to improve the security of your product catalog in AWS Marketplace](#) blog post on the *AWS Blog*.

The following is an example event body for the Products Security Report Created detail type.

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-0123456789ab",
  "detail-type": "Products Security Report Created",
  "source": "aws.marketplacecatalog",
  "account": "123456789012",
  "time": "2023-10-31T00:00:00Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "numberOfProductsWithIssues": 1,
    "numberOfVersionsWithIssues": 1,
    "numberOfLatestVersionsWithIssues": 1,
    "newIssuesFound": true,
    "upcomingResolutionDueDate": "2023-12-01T00:00:00Z",
    "requestId": "533fa17d-3e97-5051-bcaf-1fae45fb3f8b"
  }
}
```

Amazon Simple Notification Service notifications for AWS Marketplace events

AWS Marketplace can also send notifications through Amazon Simple Notification Service (Amazon SNS) about changes to buyers' subscriptions and contract entitlements for the following product types:

- [Software as a service \(SaaS\) products](#)
- [Amazon Machine Image \(AMI\) products](#)
- [Container products](#)

Seller reports, data feeds, and dashboards

AWS Marketplace provides the following tools for collecting and analyzing information about your product sales:

- [Reports](#) that are automatically created and are available to all registered AWS Marketplace sellers.
- An [API](#) that you can use to access sections of those reports.
- [Data feeds](#) that provide additional customer information that you can use to identify customer information for transactions listed in the reports.
- [Dashboards](#) powered by [Amazon QuickSight](#) with charts, graphs, and insights that help you to access and analyze financial data.

AWS Marketplace provides as much data as possible in reports, data feeds, and dashboards while adhering to the following:

- Amazon standards and tenets for protecting customer data.
- The terms and conditions that buyers accept when they buy a product on AWS Marketplace. As a seller, you are contractually bound to securely manage buyer data and to delete data upon buyer's request.

Seller delivery data feeds service

AWS Marketplace provides data feeds as a mechanism to send structured, up-to-date product and customer information from AWS Marketplace systems to seller Amazon S3 buckets for ETL (extract, transform, and load) between seller-owned business intelligence tools.

The transactional data is delivered and appended in a bi-temporal structure so sellers can store and query data along two timelines with timestamps for both

- valid time: when a fact occurred in the real world (“what you knew”)
- system time: when that fact was recorded to the database (“when you knew it”).

Data feeds are delivered daily at 4pm PST (midnight UTC) following an update from the prior day containing 24 hours of data from the previous day. An update can be defined by a customer subscribing, a customer being invoiced, or AWS disbursing payment.

This section provides an overview of data feeds and explains how to access and use them. Subsequent sections describe each data feed.

Storage and structure of data feeds

Data feeds collect and deliver comma-separated value (CSV) files to an encrypted Amazon S3 bucket that you provide. The CSV files have the following characteristics:

- They follow [4180 standards](#).
- Character encoding is UTF-8 without BOM.
- Commas are used as separators between values.
- Fields are escaped by double quotation marks.
- \n is the line feed character.
- Dates are reported in the UTC time zone, are in ISO 8601 date and time format, and are accurate within 1 second.
- All *_period_start_date and *_period_end_date values are inclusive, which means that 23:59:59 is the last possible timestamp for any day.
- All monetary fields are preceded with a currency field.
- Monetary fields use a period (.) character as a decimal separator, and don't use a comma (,) as a thousands separator.

Data feeds are generated and stored as follows:

- Data feeds are generated within a day, and contain 24 hours of data from the previous day.
- In the Amazon S3 bucket, data feeds are organized by month using the following format:

bucket-name/data-feed-name_version/year=YYYY/month=MM/data.csv

- As each daily data feed is generated, it is appended to the existing CSV file for that month. When a new month starts, a new CSV file is generated for each data feed.
- Information in data feeds is backfilled from 2010/01/01 to 2020/04/30 (inclusive) and is available in the [CSV file](#) in the year=2010/month=01 subfolder.

You may notice cases where the current month's file for a given data feed contains only column headers, and no data. This means that there were no new entries for that month for the feed. This can happen with data feeds that are updated less frequently, like the product feed. In these cases, data is available in the backfilled folder.

- In Amazon S3, you can create an [Amazon S3 lifecycle policy](#) to manage how long to keep files in the bucket.
- You can configure Amazon SNS to notify you when data is delivered to your encrypted S3 bucket. For information on how to configure notifications, see [Getting started with Amazon SNS](#) in the *Amazon Simple Notification Service Developer Guide*.

Historization of the data

Each data feed includes columns that document the history of the data. Except for `valid_to`, these columns are common to all data feeds. They're included as a common history schema and are useful in querying the data.

Column name	Description
<code>valid_from</code>	The first date that the value for the primary key is valid for in relation to values for other fields.
<code>valid_to</code>	This column is only shown on the Address data feed and is always blank.
<code>insert_date</code>	The date a record was inserted into the data feed.
<code>update_date</code>	The date the record was last updated.
<code>delete_date</code>	This column is always blank.

The following shows an example of these columns.

<code>valid_from</code>	<code>valid_to</code>	<code>insert_date</code>	<code>update_date</code>	<code>delete_date</code>
2018-12-12T02:00:00Z		2018-12-1 2T02:00:00Z	2018-12-1 2T02:00:00Z	
2019-03-29T03:00:00Z		2019-03-2 9T03:00:00Z	2019-03-2 9T03:00:00Z	
2019-03-29T03:00:00Z		2019-03-2 9T03:00:00Z	2019-04-2 8T03:00:00Z	

The `valid_from` and `update_date` field together form a *bi-temporal data model*. The `valid_from` field, as it is named, tells you when the item is valid from. If the item was edited, it can have multiple records in the feed, each with a different `update_date`, but the same `valid_from` date. For example, to find the current value for an item, you would find the record with the most recent `update_date`, from the list of records with the most recent `valid_from` date.

In the example above, the record was originally created 2018-12-12. It was then changed on 2019-03-29 (for example, if the address in the record changed). Later, on 2019-04-28, the address change was corrected (so the `valid_from` didn't change, but the `update_date` did). Correcting the address (a rare event) retroactively changes the record from the original `valid_from` date, so that field didn't change. A query to find the most recent `valid_from` would return two records, the one with the latest `update_date` gives you the actual current record.

Accessing data feeds

To access data feeds, you need to configure your environment to receive data feeds to an encrypted Amazon S3 bucket. AWS Marketplace provides an [AWS CloudFormation template](#) that you can use to simplify configuration.

To use the AWS CloudFormation template to configure your environment to receive data feeds

1. Open a web browser and sign into the [AWS Marketplace Management Portal](#), then go to [Set up customer data storage](#).
2. Choose **Create resources with AWS CloudFormation template** to open the template in the AWS CloudFormation console in another window.

3. In the template, specify the following and then choose **Next**:
 - Stack name – The collection of resources you're creating to enable access to data feeds.
 - Amazon S3 bucket name – The bucket for storing data feeds.
 - (Optional) Amazon SNS topic name – The topic for receiving notifications when AWS delivers new data to the Amazon S3 bucket.
4. On the **Review** page, confirm your entries and choose **Create stack**. This will open a new page with the CloudFormation status and details.
5. From the **Resources** tab, copy Amazon Resource Names (ARNs) for the following resources from the CloudFormation page into the fields on the AWS Marketplace [Set up customer data storage](#) page:
 - Amazon S3 bucket for storing data feeds
 - AWS KMS key for encrypting the Amazon S3 bucket
 - (Optional) Amazon SNS topic for receiving notifications when AWS delivers new data to the Amazon S3 bucket
6. On the **Set up customer data storage** page, choose **Submit**.
7. (Optional) Edit the policies created by the CloudFormation template. See [Data feed policies](#) for more details.

You are now subscribed to data feeds. The next time data feeds are generated, you can access the data.

For more information about AWS CloudFormation templates, see [Working with AWS CloudFormation templates](#) in the *AWS CloudFormation User Guide*.

Data feed policies

When your Amazon S3 bucket is created by the CloudFormation template, it will create policies for access attached to that bucket, the AWS KMS key, and the Amazon SNS topic. The policies allow the AWS Marketplace reports service to write to your bucket and SNS topic with the data feed information. Each policy will have a section like the following (this example is from the Amazon S3 bucket).

```
{
  "Sid": "AwsMarketplaceDataFeedsAccess",
  "Effect": "Allow",
```

```

    "Principal": {
      "Service": "reports.marketplace.amazonaws.com"
    },
    "Action": [
      "s3:ListBucket",
      "s3:GetObject",
      "s3:PutObject",
      "s3:GetEncryptionConfiguration",
      "s3:GetBucketAcl",
      "s3:PutObjectAcl"
    ],
    "Resource": [
      "arn:aws:s3:::datafeed-bucket",
      "arn:aws:s3:::datafeed-bucket/*"
    ]
  },

```

In this policy, `reports.marketplace.amazonaws.com` is the service principal that AWS Marketplace uses to push data to the Amazon S3 bucket. The **datafeed-bucket** is the bucket that you specified in the CloudFormation template.

When the AWS Marketplace reports service calls Amazon S3, AWS KMS, or Amazon SNS, it will provide the ARN of the data it is intending to write to the bucket when it does. To ensure that the only data written to your bucket is data written on your behalf, you can specify the `aws:SourceArn` in the condition of the policy. In the following example, you must replace the *account-id* with the ID for your AWS account.

```

{
  "Sid": "AwsMarketplaceDataFeedsAccess",
  "Effect": "Allow",
  "Principal": {
    "Service": "reports.marketplace.amazonaws.com"
  },
  "Action": [
    "s3:ListBucket",
    "s3:GetObject",
    "s3:PutObject",
    "s3:GetEncryptionConfiguration",
    "s3:GetBucketAcl",
    "s3:PutObjectAcl"
  ],

```



```
"Resource": [
  "arn:aws:s3:::datafeed-test-bucket",
  "arn:aws:s3:::datafeed-test-bucket/*"
],
"Condition": {
  "StringEquals": {
    "aws:SourceAccount": "account-id",
    "aws:SourceArn": ["arn:aws:marketplace::account-
id:AWSMarketplace/SellerDataSubscription/DataFeeds_V1",
    "arn:aws:marketplace::account-id:AWSMarketplace/
SellerDataSubscription/Example-Report"]
  }
},
```

Unsubscribing from data feeds

Open a web browser and sign in to the [AWS Marketplace Management Portal](#). Then, go to the [Contact us page](#) to submit an unsubscribe request to the AWS Marketplace Seller Operations team. The unsubscribe request can take up to 10 business days to process.

Using data feeds

When data is available in your Amazon S3 bucket, you can use data feeds in the following ways:

- Download the .CSV files from the Amazon S3 bucket you created in [Accessing data feeds](#) so that you can view the data in a spreadsheet.
- Use ETL (extract, transform, and load), SQL query, business analytics tools to collect and analyze the data.

You can use AWS services to collect and analyze data, or any third-party tool that can perform analysis of .CSV-based datasets.

Example: Use AWS services to collect and analyze data

The following procedure assumes that you've already configured your environment to receive data feeds to an Amazon S3 bucket and that the bucket contains data feeds.

To collect and analyze data from data feeds

1. From the [AWS Glue console](#), [create a crawler](#) to connect to the Amazon S3 bucket that stores the data feeds, extract the data you want, and create metadata tables in the AWS Glue Data Catalog.

For more information about AWS Glue, see the [AWS Glue Developer Guide](#).

2. From the [Athena console](#), [run SQL queries on the data in the AWS Glue Data Catalog](#).

For more information about Athena see the [Amazon Athena User Guide](#).

3. From the [Amazon QuickSight console](#), [create an analysis](#) and then [create a visual](#) of the data.

For more information about Amazon QuickSight, see the [Amazon QuickSight User Guide](#).

For a detailed example of one way to use AWS services to collect and analyze data in data feeds, see [Using Seller Data Feed Delivery Service, Amazon Athena, and Amazon QuickSight to create seller reports](#) at the AWS Marketplace Blog.

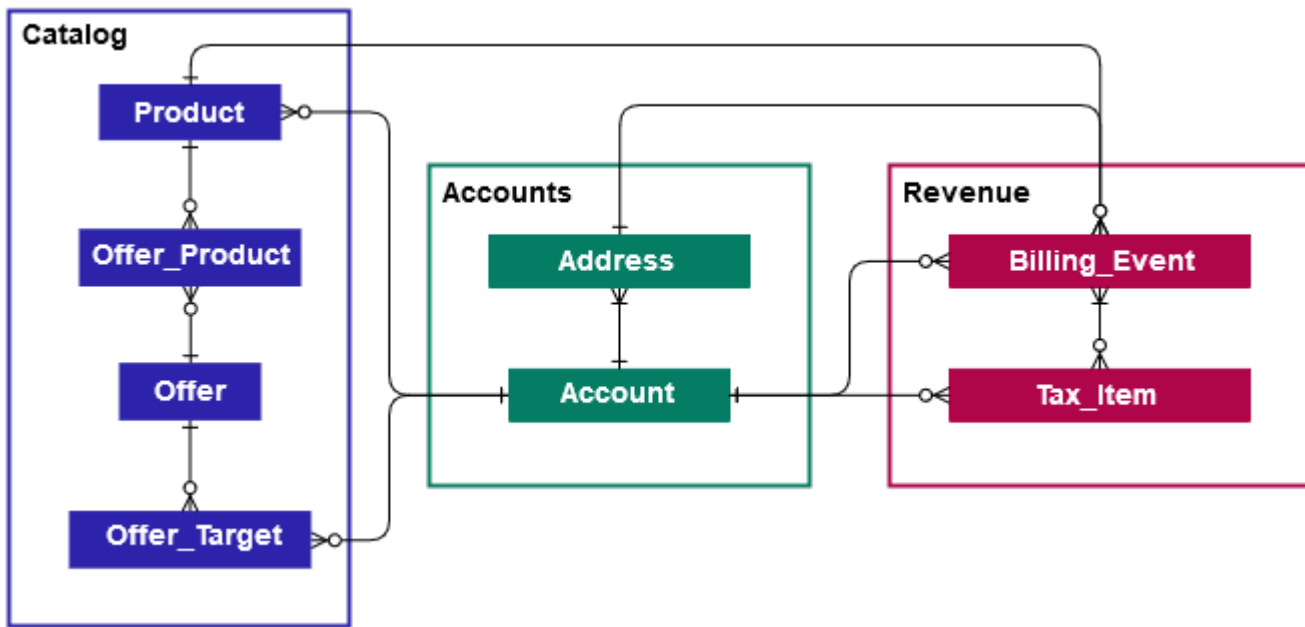
Data feed tables overview

The AWS Marketplace provided data feeds are a set of tables that you can join together to provide more context for your queries.

There are three general domains, or categories of interest, in your data feeds:

- **Catalog** – Includes information about the products and offers in your account.
- **Accounts** – Includes information about the accounts that provide or purchase products on AWS Marketplace (your own accounts or accounts of parties that you work with such as channel partners or buyers).
- **Revenue** – Includes information about billing, disbursements, and taxes.

The following diagram shows the tables in each domain, and how they are related to each other. This diagram shows the Catalog, Accounts, and Revenue domains, including the tables within them.






The following sections provide *entity relationship* (ER) diagrams for each domain. Each ER diagram shows the tables and the fields within each table, as well as the fields that you can use to join the tables.

Note

The ER diagrams in this section do not include the common fields for all data feeds. For more information about the common fields, see [Storage and structure of data feeds](#).

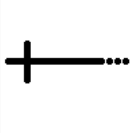
The following table describes the symbols that are used in the ER diagrams.




Symbol	Description
PK	Primary key – A primary key for the table. When used with the <code>valid_from</code> and <code>update_date</code> fields, it is unique. For more details about using these fields together, see Historization of the data . If more than one field is marked as primary key, then the fields together form the primary key.

Symbol	Description
	<p>Foreign key – A field that represents a primary key in a different table. Not necessarily unique in the table.</p> <div style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px; margin-top: 10px;"> <p>Note</p> <p>In some cases, the foreign key can be blank if the record in the current table does not have a corresponding record in the foreign table.</p> </div>
	<p>Alternate key – A key that can be used as a key in the table. Follows the same uniqueness rules as the primary key.</p>
	<p>Connector – Lines between fields represent a connection, which is two fields that can be used to join tables. The ends of the line represent the type of connection. This example represents a one-to-many connection.</p>

Connector types

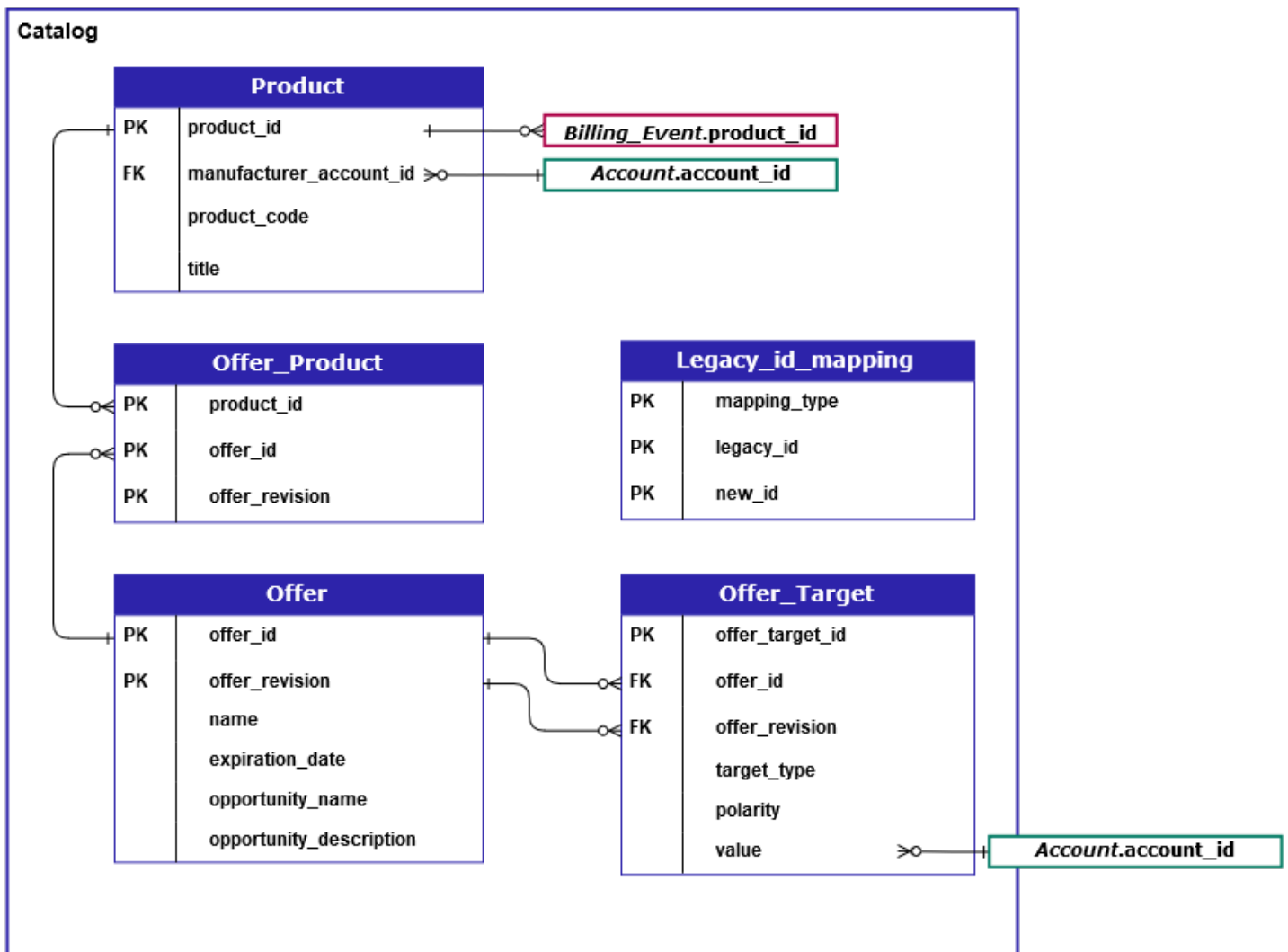
The following table shows the types of ends that each connector can have.

Connector type	Description
	<p>One to n – A connector with this end represents a join that has exactly one value on this side of the join.</p>

Connector type	Description
	Zero or one to n – A connector with this end represents a join that has zero or one values on this side of the join.
	Zero or more to n – A connector with this end represents a join that has zero, one, or many values on this side of the join.
	One or more to n – A connector with this end represents a join that has one or many values on this side of the join.

Catalog-related tables

The following diagram shows the relationships between tables in the Catalog domain, as well as the fields within the tables.



The Product, Offer_Product, Offer, Offer_Target, and Legacy_id_mapping tables are in the Catalog domain.

The Offer_Target table includes a value field for the account_id of the target, but only when the target_type value is account.

The Legacy_id_mapping table is not used for current data.

Note

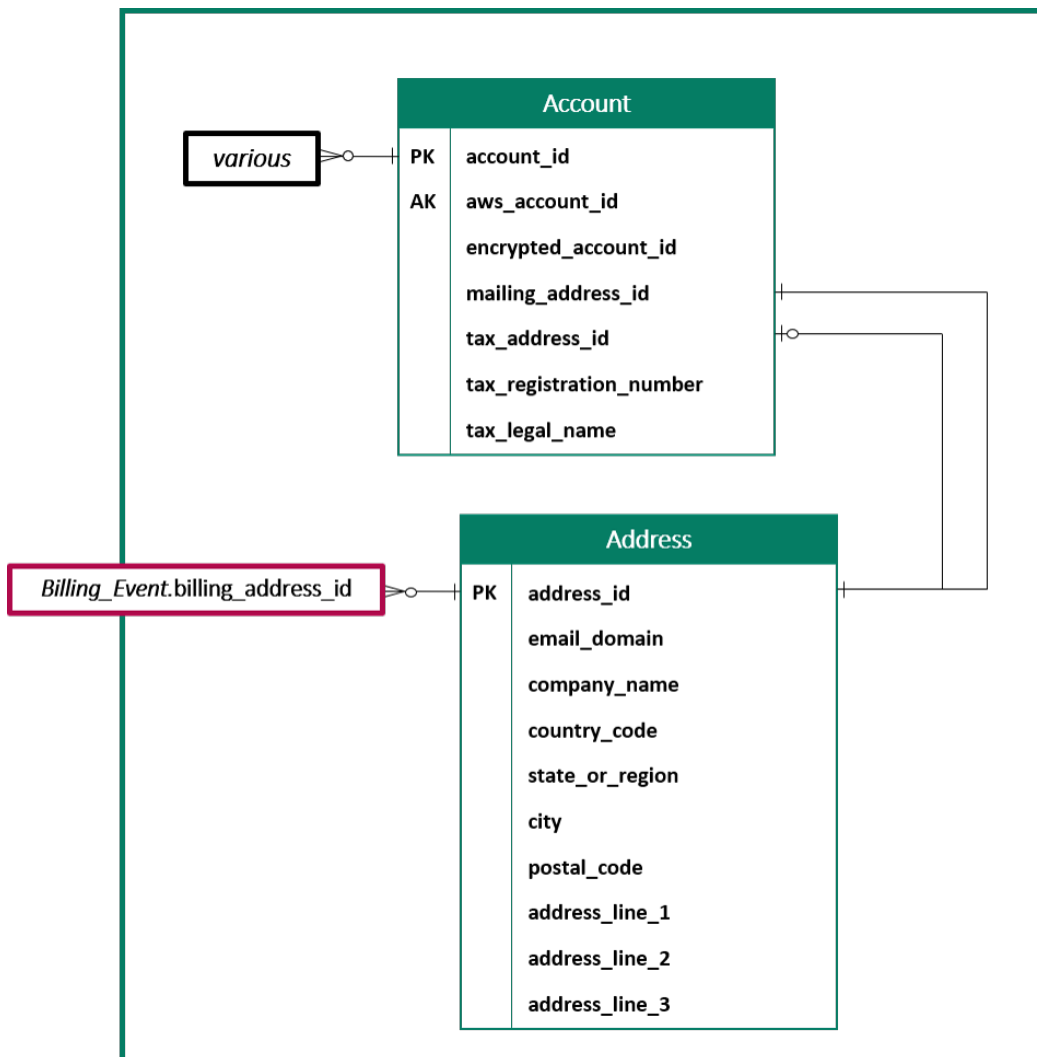
For more information about these tables, including a description of each field in the table and the joins that can be created, see the following topics:

- [Product data feed](#)
- [Offer product data feed](#)

- [Offer data feed](#)
- [Offer target data feed](#)
- [Legacy mapping data feed](#)

Accounts-related tables

The following diagram shows the relationships between the Account and Address tables in the Accounts domain, as well as the fields within the tables.



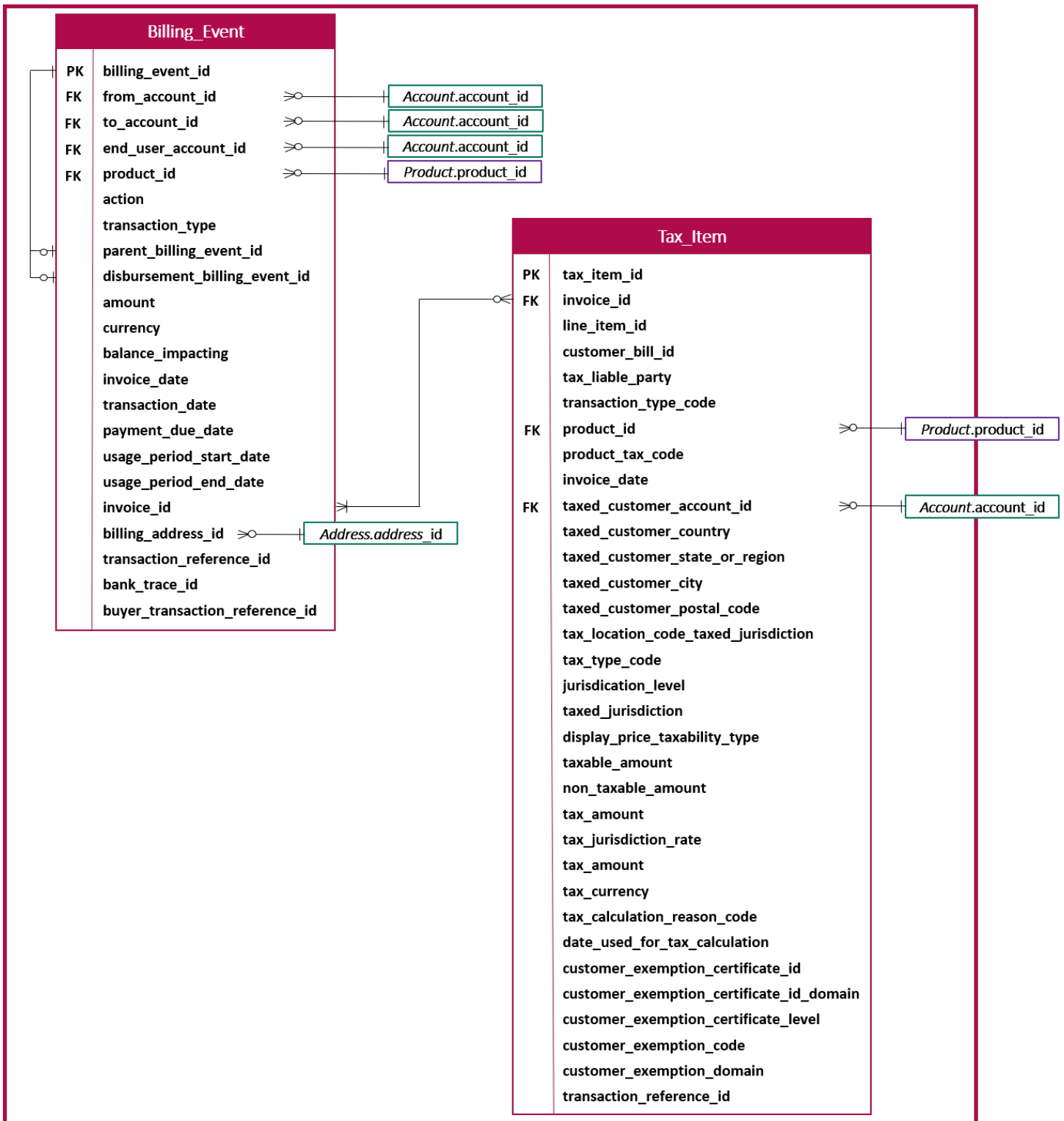
Note

For more information about these tables, including a description of each field in the table and the joins that can be created, see the following topics:

- [Account data feed](#)
- [Address data feed](#)

Revenue-related tables

The following diagram shows the relationships between the `Billing_Event` and `Tax_Item` tables in the Revenue domain, as well as the fields within the tables. The `Billing_Event` table includes information about disbursements, as well as billing events.



Note

For more information about these tables, including a description of each field in the table and the joins that can be created, see the following topics:

- [Billing event data feed](#)
- [Tax item data feed](#)

Data feed query examples

This section gives examples of complex queries using the data feeds provided by AWS Marketplace. These examples are similar to the [Seller reports](#) that you get from the AWS Marketplace Management Portal. You can customize these queries to create other reports that you need.

Example 1: Disbursements by product

To find out the amount that's been disbursed by product, you can run a query like the following. This example is comparable to the [Disbursement report](#) that you can get as a seller report. However, you can use this sample to build your own queries and customize it to get exactly the report that you need.

This set of sample queries build upon each other to create the final list of product details with disbursements. It also shows how to get the product information at a specific point in time. The comments in the queries explain what the queries are doing, as well as how you can modify them to get different views of the data.

Note

When running this query, we are assuming that the data ingested is using two time axes (the `valid_from` column and the `update` column). For more details, see [Storage and structure of data feeds](#).

```
-- Get all the products and keep the latest product_id, valid_from tuple
with products_with_uni_temporal_data as (
  select
    *
  from
```

```
(
  select
    *,
    ROW_NUMBER() OVER (PARTITION BY product_id, valid_from
      ORDER BY from_iso8601_timestamp(update_date) desc)
      as row_num
  from
    productfeed_v1
)
where
  -- A product_id can appear multiple times with the same
  -- valid_from date but with a different update_date column,
  -- making it effectively bi-temporal. By only taking the most
  -- recent tuple, we are converting to a uni-temporal model.
  row_num = 1
),

-- Gets the latest revision of a product
-- A product can have multiple revisions where some of the
-- columns, like the title, can change.
-- For the purpose of the disbursement report, we want
-- to get the latest revision of a product
products_with_latest_version as (
  select
    *
  from
    (
      select
        *,
        ROW_NUMBER() OVER (PARTITION BY product_id
          ORDER BY from_iso8601_timestamp(valid_from) desc)
          as row_num_latest_version
      from
        products_with_uni_temporal_data
    )
  where
    row_num_latest_version = 1
),

-- Get all the accounts and keep the latest account_id, valid_from tuple
accounts_with_uni_temporal_data as (
  select
    *
  from
```

```
(
  select
    *,
    ROW_NUMBER() OVER (PARTITION BY account_id, valid_from ORDER BY
from_iso8601_timestamp(update_date) desc) as row_num
  from
    accountfeed_v1
)
where
  -- An account_id can appear multiple times with the same
  -- valid_from date but with a different update_date column,
  -- making it effectively bi-temporal. By only taking the most
  -- recent tuple, we are converting to a uni-temporal model.
  row_num = 1
),

-- Gets the latest revision of an account
-- An account can have multiple revisions where some of the
-- columns, like the mailing_address_id, can change.
-- For the purpose of the disbursement report, we want
-- to get the latest revision of a product
accounts_with_latest_version as (
  select
    *
  from
    (
      select
        *,
        ROW_NUMBER() OVER (PARTITION BY account_id
          ORDER BY from_iso8601_timestamp(valid_from) desc)
          as row_num_latest_version
      from
        accounts_with_uni_temporal_data
    )
  where
    row_num_latest_version = 1
),

-- Get all the billing events and keep the
-- latest billing_event_id, valid_from tuple:
billing_events_with_uni_temporal_data as (
  select
    *
  from (
```

```
select
  billing_event_id,
  from_iso8601_timestamp(valid_from) as valid_from,
  from_iso8601_timestamp(update_date) as update_date,
  from_iso8601_timestamp(invoice_date) as invoice_date,
  transaction_type,
  transaction_reference_id,
  product_id,
  disbursement_billing_event_id,
  action,
  from_account_id,
  to_account_id,
  end_user_account_id,
  CAST(amount as decimal(20, 10)) invoice_amount,
  bank_trace_id,
  ROW_NUMBER() OVER (PARTITION BY billing_event_id, valid_from
    ORDER BY from_iso8601_timestamp(update_date) desc)
    as row_num
from
  billingeventfeed_v1
)
where row_num = 1
),

-- Get all the disbursements
-- The billing events data is immutable.
-- It is not required to use time windows based on the
-- valid_from column to get the most recent billing event
disbursement_events as (
  select
    billing_events_raw.billing_event_id as disbursement_id,
    billing_events_raw.invoice_date as disbursement_date,
    billing_events_raw.bank_trace_id
  from
    billing_events_with_uni_temporal_data billing_events_raw
  where
    -- Only interested in disbursements, so filter out
    -- non-disbursements by selecting transaction type
    -- to be DISBURSEMENT:
    billing_events_raw.transaction_type = 'DISBURSEMENT'
    -- Select a time period, you can adjust the dates
    -- below if need be. For billing events use the
    -- invoice date as the point in time of the
    -- disbursement being initiated:
```

```
    and billing_events_raw.invoice_date >=
      from_iso8601_timestamp('2020-10-01T00:00:00Z')
    and billing_events_raw.invoice_date <
      from_iso8601_timestamp('2020-11-01T00:00:00Z')
  ),

-- Get the invoices along with the line items that
-- are part of the above filtered disbursements
disbursed_line_items as (
  select
    line_items.transaction_reference_id,
    line_items.product_id,
    line_items.transaction_type,
    (case
      -- Get the payer of the invoice from any
      -- transaction type that is not AWS and
      -- not BALANCE_ADJUSTMENT.
      -- For AWS and BALANCE_ADJUSTMENT, the billing
      -- event feed will show the "AWS Marketplace"
      -- account as the receiver of the funds and the
      -- seller as the payer. Filter those out.
      when line_items.transaction_type
        not like '%AWS%' and transaction_type
        not like 'BALANCE_ADJUSTMENT'
        then line_items.from_account_id
      end) as payer_account_id,
    line_items.end_user_account_id,
    invoice_amount,
    disbursements.disbursement_date,
    disbursements.disbursement_id,
    disbursements.bank_trace_id
  from
    billing_events_with_uni_temporal_data line_items
    -- Each disbursed line item is linked to the parent
    -- disbursement via the disbursement_billing_event_id
  join disbursement_events disbursements
    on disbursements.disbursement_id
    = line_items.disbursement_billing_event_id
  where
    -- we are interested only in the invoice line
    -- items that are DISBURSED
    line_items.action = 'DISBURSED'
),
```

```
-- An invoice can contain multiple line items
-- Create a pivot table to calculate the different
-- amounts that are part of an invoice.
-- The new row is aggregated at
-- transaction_reference_id - end_user_account_id level
invoice_amounts_aggregated as (
  select
    transaction_reference_id,
    product_id,
    -- a given disbursement id should have the
    -- same disbursement_date
    max(disbursement_date) as disbursement_date,
    -- Build a pivot table in order to provide all the
    -- data related to a transaction in a single row.
    -- Note that the amounts are negated. This is because
    -- when an invoice is generated, we give you the
    -- positive amounts and the disbursement event
    -- negates the amounts
    sum(case when transaction_type = 'SELLER_REV_SHARE'
      then -invoice_amount else 0 end) as seller_rev_share,
    sum(case when transaction_type = 'AWS_REV_SHARE'
      then -invoice_amount else 0 end) as aws_rev_share,
    sum(case when transaction_type = 'SELLER_REV_SHARE_REFUND'
      then -invoice_amount else 0 end) as seller_rev_refund,
    sum(case when transaction_type = 'AWS_REV_SHARE_REFUND'
      then -invoice_amount else 0 end) as aws_rev_refund,
    sum(case when transaction_type = 'SELLER_REV_SHARE_CREDIT'
      then -invoice_amount else 0 end) as seller_rev_credit,
    sum(case when transaction_type = 'AWS_REV_SHARE_CREDIT'
      then -invoice_amount else 0 end) as aws_rev_credit,
    sum(case when transaction_type = 'SELLER_TAX_SHARE'
      then -invoice_amount else 0 end) as seller_tax_share,
    sum(case when transaction_type = 'SELLER_TAX_SHARE_REFUND'
      then -invoice_amount else 0 end) as seller_tax_refund,
    -- This is the account that pays the invoice:
    max(payer_account_id) as payer_account_id,
    -- This is the account that subscribed to the product:
    end_user_account_id as customer_account_id,
    bank_trace_id
  from
    disbursed_line_items
  group by
    transaction_reference_id,
    product_id,
```

```

    disbursement_id,
    -- There might be a different end-user for the same
    -- transaction reference id. Distributed licenses
    -- is an example
    end_user_account_id,
    bank_trace_id
),

disbursed_amount_by_product as (
  select
    products.title as ProductTitle,
    products.product_code as ProductCode,
    -- We are rounding the sums using 2 decimal precision
    -- Note that the rounding method might differ
    -- between SQL implementations.
    -- The disbursement seller report is using
    -- RoundingMode.HALF_UP. This might create
    -- discrepancies between this SQL output
    -- and the disbursement seller report
    round(invoice_amounts.seller_rev_share, 2) as SellerRev,
    round(invoice_amounts.aws_rev_share, 2) as AWSRefFee,
    round(invoice_amounts.seller_rev_refund, 2) as SellerRevRefund,
    round(invoice_amounts.aws_rev_refund, 2) as AWSRefFeeRefund,
    round(invoice_amounts.seller_rev_credit, 2) as SellerRevCredit,
    round(invoice_amounts.aws_rev_credit, 2) as AWSRefFeeCredit,
    (
      round(invoice_amounts.seller_rev_share, 2) +
      round(invoice_amounts.aws_rev_share, 2) +
      round(invoice_amounts.seller_rev_refund, 2) +
      round(invoice_amounts.aws_rev_refund, 2) +
      round(invoice_amounts.seller_rev_credit, 2) +
      round(invoice_amounts.aws_rev_credit, 2)
    ) as NetAmount,
    invoice_amounts.transaction_reference_id
      as TransactionReferenceID,
    round(invoice_amounts.seller_tax_share, 2)
      as SellerSalesTax,
    round(invoice_amounts.seller_tax_refund, 2)
      as SellerSalesTaxRefund,
    payer_info.aws_account_id
      as PayerAwsAccountId,
    customer_info.aws_account_id
      as EndCustomerAwsAccountId,
    invoice_amounts.disbursement_date

```



```

        as DisbursementDate,
    invoice_amounts.bank_trace_id
        as BankTraceId
from
    invoice_amounts_aggregated invoice_amounts
join products_with_latest_version products
    on products.product_id = invoice_amounts.product_id
left join accounts_with_latest_version payer_info
    on payer_info.account_id = invoice_amounts.payer_account_id
left join accounts_with_latest_version customer_info
    on customer_info.account_id = invoice_amounts.customer_account_id
)

select * from disbursed_amount_by_product;

```

Example 2: Sales compensation report

To find the billed revenue by customer, you can run a query like the following. This example is comparable to the [Sales compensation report](#) that you can get as a seller report. However, you can use this sample to build your own queries and customize it to get exactly the report that you need.

This is a set of sample queries that build upon each other to create the final list of customer details with the total amount billed to each customer for usage of your software. The comments in the queries explain what the queries are doing, as well as how you can modify them to get different views of the data.

Note

When running this query, we are assuming that the data ingested is using two time axes (the `valid_from` column and the `update` column). For more details, see [Storage and structure of data feeds](#).

```

-- Gets all the products and keeps the latest product_id,
-- valid_from tuple.
with products_with_uni_temporal_data as (
    select
        *
    from
        (
            select

```

```
    *,
    ROW_NUMBER() OVER (PARTITION BY product_id, valid_from
                       ORDER BY from_iso8601_timestamp(update_date) desc)
                       as row_num
from
  productfeed_v1
)
where
  -- A product_id can appear multiple times with the same
  -- valid_from date but with a different update_date column,
  -- making it effectively bi-temporal. By only taking the most
  -- recent tuple, we are converting to a uni-temporal model.
  row_num = 1
),

-- Gets the latest revision of a product
-- A product can have multiple revisions where some of the
-- columns, like the title, can change.
-- For the purpose of the sales compensation report, we want
-- to get the latest revision of a product
products_with_latest_revision as (
select
  *
from
  (
  select
    *,
    ROW_NUMBER() OVER (PARTITION BY product_id ORDER BY
from_iso8601_timestamp(valid_from) desc) as row_num_latest_revision
  from
    products_with_uni_temporal_data
  )
where
  row_num_latest_revision = 1
),

-- Gets all the addresses and keeps the latest address_id,
-- aws_account_id, and valid_from combination.
-- We're transitioning from a bi-temporal data model to an
-- uni-temporal data_model
piifeed_with_uni_temporal_data as (
select
  *
from
```

```

(
  select
    *,
    ROW_NUMBER() OVER (
      PARTITION BY address_id, aws_account_id, valid_from
      ORDER BY from_iso8601_timestamp(update_date) desc)
      as row_num
  from
    piifeed
)
where
  -- An address_id can appear multiple times with the same
  -- valid_from date but with a different update_date column.
  -- We are only interested in the most recent.
  row_num = 1
),

-- Gets the latest revision of an address.
-- An address_id can have multiple revisions where some of
-- the columns can change.
-- For the purpose of the sales compensation report, we want to
-- get the latest revision of an address + account_id pair.
pii_with_latest_revision as (
  select
    *
  from
    (
      select
        *,
        ROW_NUMBER() OVER (PARTITION BY address_id, aws_account_id
          ORDER BY from_iso8601_timestamp(valid_from) desc)
          as row_num_latest_revision
      from
        piifeed_with_uni_temporal_data
    )
  where
    row_num_latest_revision = 1
),

-- Gets all the accounts and keeps the latest
-- account_id, valid_from tuple.
-- We're transitioning from a bi-temporal data
-- model to an uni-temporal data_model.
accounts_with_uni_temporal_data as (

```

```
select
  *
from
  (
    select
      *,
      ROW_NUMBER() OVER (PARTITION BY account_id, valid_from
        ORDER BY from_iso8601_timestamp(update_date) desc)
        as row_num
    from
      accountfeed_v1
  )
where
  -- An account_id can appear multiple times with the same
  -- valid_from date but with a different update_date column.
  -- We are only interested in the most recent tuple.
  row_num = 1
),

-- Gets all the historical dates for an account
-- An account can have multiple revisions where some of the
-- columns like the mailing_address_id can change.
accounts_with_history as (
  select
    *,
    -- This interval's begin_date
    case
      when
        -- First record for a given account_id
        lag(valid_from, 1) over (partition by account_id
          order by from_iso8601_timestamp(valid_from) asc) is null
        then
          -- 'force' begin_date a bit earlier because of different
          -- data propagation times. We'll subtract one day as one
          -- hour is not sufficient
          from_iso8601_timestamp(valid_from) - INTERVAL '1' DAY
        else
          -- not the first line -> return the real date
          from_iso8601_timestamp(valid_from)
        end as begin_date,
    -- This interval's end date.
    COALESCE(
      LEAD(from_iso8601_timestamp(valid_from), 1)
      OVER (partition by account_id
```

```

        ORDER BY from_iso8601_timestamp(valid_from)),
        from_iso8601_timestamp('9999-01-01T00:00:00Z')
    ) as end_date
from
    accounts_with_uni_temporal_data
),

-- Gets all the billing events and keeps the latest
-- billing_event_id, valid_from tuple.
-- We're transitioning from a bi-temporal data
-- model to an uni-temporal data_model.
billing_events_with_uni_temporal_data as (
    select
        *
    from (
        select
            billing_event_id,
            from_iso8601_timestamp(valid_from) as valid_from,
            from_iso8601_timestamp(update_date) as update_date,
            from_iso8601_timestamp(invoice_date) as invoice_date,
            transaction_type,
            transaction_reference_id,
            product_id,
            disbursement_billing_event_id,
            action,
            currency,
            from_account_id,
            to_account_id,
            end_user_account_id,
            -- convert an empty billing address to null. This will
            -- later be used in a COALESCE call
            case
                when billing_address_id <> '' then billing_address_id else null
            end as billing_address_id,
            CAST(amount as decimal(20, 10)) invoice_amount,
            ROW_NUMBER() OVER (PARTITION BY billing_event_id, valid_from
                ORDER BY from_iso8601_timestamp(update_date) desc)
                as row_num
        from
            billingeventfeed_v1
        where
            -- The Sales Compensation Report does not contain BALANCE
            -- ADJUSTMENTS, so we filter them out here
            transaction_type <> 'BALANCE_ADJUSTMENT'
    )

```

```

        -- Keep only the transactions that will affect any
        -- future disbursed amounts.
        and balance_impacting = '1'
    )
    where row_num = 1
),

-- Gets the billing address for all DISBURSED invoices. This
-- will be the address of the payer when the invoice was paid.
-- NOTE: For legal reasons, for CPP0 transactions, the
-- manufacturer will not see the payer's billing address id
billing_addresses_for_disbursed_invoices as (
    select
        billing_events_raw.transaction_reference_id,
        billing_events_raw.billing_address_id,
        billing_events_raw.from_account_id
    from
        billing_events_with_uni_temporal_data billing_events_raw
    where
        -- the disbursed items will contain the billing address id
        billing_events_raw.action = 'DISBURSED'
        -- we only want to get the billing address id for the
        -- transaction line items where the seller is the receiver
        -- of the amount
        and billing_events_raw.transaction_type like 'SELLER_%'
    group by
        billing_events_raw.transaction_reference_id,
        billing_events_raw.billing_address_id,
        billing_events_raw.from_account_id
),

-- An invoice can contain multiple line items.
-- We create a pivot table to calculate the different amounts
-- that are part of an invoice.
-- The new row is aggregated at
-- transaction_reference_id - end_user_account_id level
invoiced_and_forgiven_transactions as (
    select
        transaction_reference_id,
        product_id,
        -- A transaction will have the same invoice date for all
        -- of its line items (transaction types)
        max(invoice_date) as invoice_date,
        -- A transaction will have the same billing_address_id

```

```
-- for all of its line items. Remember that the billing event
-- is uni temporal and we retrieved only the latest valid_from item
max(billing_address_id) as billing_address_id,
-- A transaction will have the same currency for all
-- of its line items
max(currency) as currency,
-- We're building a pivot table in order to provide all the
-- data related to a transaction in a single row
sum(case when transaction_type = 'SELLER_REV_SHARE'
      then invoice_amount else 0 end) as seller_rev_share,
sum(case when transaction_type = 'AWS_REV_SHARE'
      then invoice_amount else 0 end) as aws_rev_share,
sum(case when transaction_type = 'SELLER_REV_SHARE_REFUND'
      then invoice_amount else 0 end) as seller_rev_refund,
sum(case when transaction_type = 'AWS_REV_SHARE_REFUND'
      then invoice_amount else 0 end) as aws_rev_refund,
sum(case when transaction_type = 'SELLER_REV_SHARE_CREDIT'
      then invoice_amount else 0 end) as seller_rev_credit,
sum(case when transaction_type = 'AWS_REV_SHARE_CREDIT'
      then invoice_amount else 0 end) as aws_rev_credit,
sum(case when transaction_type = 'SELLER_TAX_SHARE'
      then invoice_amount else 0 end) as seller_tax_share,
sum(case when transaction_type = 'SELLER_TAX_SHARE_REFUND'
      then invoice_amount else 0 end) as seller_tax_refund,
-- this is the account that pays the invoice.
max(case
  -- Get the payer of the invoice from any transaction type
  -- that is not AWS and not BALANCE_ADJUSTMENT.
  -- For AWS and BALANCE_ADJUSTMENT, the billing event feed
  -- will show the "AWS Marketplace" account as the
  -- receiver of the funds and the seller as the payer. We
  -- are not interested in this information here.
  when
    transaction_type not like '%AWS%'
    and transaction_type not like 'BALANCE_ADJUSTMENT'
    then from_account_id
  end) as payer_account_id,
-- this is the account that subscribed to your product
end_user_account_id as customer_account_id
from
  billing_events_with_uni_temporal_data
where
  -- Get invoiced or forgiven items. Disbursements are
  -- not part of the sales compensation report
```

```

        action in ('INVOICED', 'FORGIVEN')
    group by
        transaction_reference_id,
        product_id,
        -- There might be a different end-user for the same
        -- transaction reference id. Distributed licenses
        -- is an example.
        end_user_account_id
    ),

invoiced_items_with_product_and_billing_address as (
    select
        invoice_amounts.*,
        products.product_code,
        products.title,
        payer_info.aws_account_id as payer_aws_account_id,
        payer_info.account_id as payer_reference_id,
        customer_info.aws_account_id as end_user_aws_account_id,
        (
            invoice_amounts.seller_rev_share +
            invoice_amounts.aws_rev_share +
            invoice_amounts.seller_rev_refund +
            invoice_amounts.aws_rev_refund +
            invoice_amounts.seller_rev_credit +
            invoice_amounts.aws_rev_credit +
            invoice_amounts.seller_tax_share +
            invoice_amounts.seller_tax_refund
        ) as seller_net_revenue,
        -- Try to get the billing address from the DISBURSED event
        -- (if any). If there is no DISBURSEMENT, get the billing
        -- address from the INVOICED item. If still no billing address,
        -- then default to getting the mailing address of the payer.
        coalesce(billing_add.billing_address_id,
            invoice_amounts.billing_address_id,
            payer_info.mailing_address_id)
            as final_billing_address_id
    from
        invoiced_and_forgiven_transactions invoice_amounts
    join products_with_latest_revision products
        on products.product_id = invoice_amounts.product_id
    left join accounts_with_history payer_info
        on payer_info.account_id = invoice_amounts.payer_account_id
        -- Get the Payer Information at the time of invoice creation
        and payer_info.begin_date <= invoice_amounts.invoice_date

```



```
        and invoice_amounts.invoice_date < payer_info.end_date
left join accounts_with_history customer_info
    on customer_info.account_id = invoice_amounts.customer_account_id
    -- Get the End User Information at the time of invoice creation
    and customer_info.begin_date <= invoice_amounts.invoice_date
    and invoice_amounts.invoice_date < customer_info.end_date
left join billing_addresses_for_disbursed_invoices billing_add
    on billing_add.transaction_reference_id =
        invoice_amounts.transaction_reference_id
    and billing_add.from_account_id =
        invoice_amounts.payer_account_id
),

invoices_with_full_address as (
    select
        payer_aws_account_id as "Customer AWS Account Number",
        pii_data.country as "Country",
        pii_data.state_or_region as "State",
        pii_data.city as "City",
        pii_data.postal_code as "Zip Code",
        pii_data.email_domain as "Email Domain",
        product_code as "Product Code",
        title as "Product Title",
        seller_rev_share as "Gross Revenue",
        aws_rev_share as "AWS Revenue Share",
        seller_rev_refund as "Gross Refunds",
        aws_rev_refund as "AWS Refunds Share",
        seller_net_revenue as "Net Revenue",
        currency as "Currency",
        date_format(invoice_date, '%Y-%m')as "AR Period",
        transaction_reference_id as "Transaction Reference ID",
        payer_reference_id as "Payer Reference ID",
        end_user_aws_account_id as "End Customer AWS Account ID"
    from
        invoiced_items_with_product_and_billing_address invoice_amounts
    left join pii_with_latest_revision pii_data
        on pii_data.aws_account_id = invoice_amounts.payer_aws_account_id
        and pii_data.address_id = invoice_amounts.final_billing_address_id
    -- Filter out FORGIVEN and Field Demonstration Pricing transactions
    where seller_net_revenue <> 0
)

select * from invoices_with_full_address;
```

Data feeds

AWS Marketplace provides a number of data feeds to help sellers collect and analyze information about your product sales. Data feeds are available to all registered AWS Marketplace sellers. Since data feeds are generated within a day, they contain the most current data available.

The following sections describe each data feed.

Topics

- [Account data feed](#)
- [Address data feed](#)
- [Billing event data feed](#)
- [Legacy mapping data feed](#)
- [Offer data feed](#)
- [Offer product data feed](#)
- [Offer target data feed](#)
- [Product data feed](#)
- [Tax item data feed](#)

Account data feed

This data feed provides information about all the accounts you interact with: your own, any channel partners you work with, buyers, payers, and all taxed accounts.

Account data is immutable, and it is not associated with a version number. Changes to fields are appended, so this data feed may have several rows with the same `account_id` and different `valid_from` values. For information about data history fields, see [Historization of the data](#).

The account data feed is refreshed every 24 hours, so new data is available daily.

The following table explains the names and descriptions of the data feed's columns.

Column name	Description
<code>account_id</code>	The globally unique identifier (GUID) of the account.

Column name	Description
	Can be used to join to fields in the Product, Offer_Target , Billing_Event , and Tax_Item data feeds. See those data feeds for information about the fields that can be used to join.
aws_account_id	The AWS account number of the seller's AWS account, which is unique by AWS partition.
encrypted_account_id	The unique, encrypted ID for an individual buyer of your application. The value for encrypted_account_id is used by the AWS Marketplace Metering Service, for example, as the value for CustomerIdentifier that is returned by the ResolveCustomer action.
mailing_address_id	The mailing address reference for this account.
tax_address_id	The tax address reference for this account.
tax_registration_number	For non-US accounts, the tax registration number for this account.
tax_legal_name	For non-US accounts, the legal company name. This is the name used on tax invoices.

Example of account data feed

The following shows an example of the account data feed. For readability, the data history columns aren't shown. For information about data history fields, see [Historization of the data](#).

account_id	aws_account_id	encrypted_account_id	mailing_address_id	tax_address_id	tax_registration_number	tax_legal_name
xk0CSmiAm6PQ4QqEog9iiaochlz uPlkMfba7a1oDLZ	444456660000	Zf7oMzheGWpH	25o3k46eN6eViOfFiiqtxwX8e3kaOiPalUiofjyFa3			
7nyo5jwTRoPlyX81vx9ji04eEwTurO1Ff8biQi88W8	555567679999	373vuQUqmQ8v	5oJ6vTjSzMrrF2gvh2Vj9HfqiM800MuLEHmyFY5Lr42s8	5oJ6vTjSzMrrF2gvh2Vj9HfqiM800MuLEHmyFY5Lr42s8	SE823935083345	
VleGa2t9j3MuxioH9wc8lsndXXCgGCGUreeXriocM5	73739998888	8SPxAYmi8MwX	NLUc5UeiMlGFTTrDWCofDPhDUF1oaSd8xgl5QM8Db7	V5NhBYBiYogwy0WMhrdGU4AfMggmuoTC2j7Pm8ZKKNyT	DE469558025	

Address data feed

Important

AWS Marketplace will discontinue the PIIFeed, which is delivered using the [seller delivery data feeds service](#), in December 2023. Use the AddressFeed_V1 data feed for your address data needs. If you have questions or require support, contact the [AWS Marketplace Seller Operations](#) team.

This data feed provides contact information for all the accounts you interact with: your own, any channel partners you work with, buyers, payers, and all taxed accounts. Each time a new transaction occurs, the customer address for the transaction is scanned, and if it's not in your data feed, a new entry is added to your data feed file.

Address data is immutable.

The address data feed is refreshed every 24 hours, so new data is available daily.

The following table explains the names and descriptions of the data feed's columns.

Column name	Description
address_id	The unique key of the address. Can be used to join from the Billing_Event data feed on the billing_address_id field, or from the Account data feed on the mailing_address_id or tax_address_id fields.
email_domain	The domain for the email address on file for this account.
company_name	The company name on file for this account.
country_code	The ISO 3166 alpha-2 country code on file for this address.
state_or_region	The state or region on file for this address.
city	The city on file for this address.
postal_code	The postal code on file for this address.
address_line_1	The first line of the address on file for this address.
address_line_2	The second line of the address on file for this address.
address_line_3	The third line of the address on file for this address.

Example of address data feed

The following shows an example of the address data feed. In the data feed, this information is presented in a single table. For readability, the data is shown in two tables here, and the data history columns aren't shown. For information about data history fields, see [Historization of the data](#).

address_id	email_domain	company_name	country_code	state_or_region	city	postal_code
V5NhBYBiYogwy0WMhrdGU4AfMggmuoTC2j7Pm8ZKKNNyT	a.com	Mateo Jackson's Company	DE		Hamburg	67568
G68xdbkZQDVVHzfBGw6yf5yos0A6NiSVWHm+5ViLjf	b.com	Mary Major's Company	US	OH	Dayton	57684
NLUc5UeiMlGFTTrDWCoftDPhDUF1oaSd8xgl5QM8Db7	c.com	Our Seller	US	NY	New York	89475

address_line_1	address_line_2	address_line_3
	19th Floor	

Billing event data feed

This data feed provides information about billing events, including invoicing and disbursements.

For example, you can use this data feed to learn when and what a buyer is invoiced. You can also use the [example SQL queries](#) to analyze the data from this data feed.

This data feed contains information associated with billing events for which you are the seller of record. For agreements made through channel partners, this data feed contains information about billing events between the manufacturer and seller of record.

The billing event data feed is refreshed every 24 hours, so new data is available daily.

Billing event data is immutable.

The following table explains the names and descriptions of the data feed's columns.

Column name	Description
billing_event_id	An identifier for a billing event. This ID is unique in the seller's environment.
from_account_id	The account that initiated the billing event. If <code>transaction_type</code> is <code>SELLER_RE V_SHARE</code> , it is the buyer's payer account. This is a foreign key to the account data feed. Can be used to join to the Account data feed on the <code>account_id</code> field.
to_account_id	The account that receives the transaction amount for the product. This is a foreign key to the account data feed. Can be used to join to the Account data feed on the <code>account_id</code> field.
end_user_account_id	The account that uses the product. This account may be different from the buyer and payer accounts.

Column name	Description
	Can be used to join to the Account data feed on the <code>account_id</code> field.
product_id	<p>The identifier of the product. This is a foreign key to the product data feed.</p> <p>Can be used to join to the Product data feed on the <code>product_id</code> field.</p>
action	<p>The type of action for this event. Possible values are as follows:</p> <ul style="list-style-type: none">• INVOICED – The buyer was invoiced for the amount.• FORGIVEN – The buyer was invoiced for the amount, and AWS reverted the charge.• DISBURSED – The seller was paid this amount. This can include a month of invoices, or be an on-demand disbursement.

Column name	Description
transaction_type	<p>The type of transaction. For examples, see Taxing scenarios. Possible values are as follows:</p> <ul style="list-style-type: none"> • SELLER_REV_SHARE – A positive amount; this is the price that the seller set in the agreement with the buyer. • SELLER_TAX_SHARE – A positive amount; this is the amount added to SELLER_REV_SHARE to cover taxes that the seller owes. • AWS_REV_SHARE – A negative amount; this is the listing fee. • AWS_TAX_SHARE – A positive amount; this is the amount of taxes AWS collected in addition to SELLER_REV_SHARE . This amount doesn't affect the seller's balance. This amount is not disbursed and is provided for the seller's awareness of taxes invoiced to the buyer and remitted to authorities on the seller's behalf. • <i>transaction_type</i> _REFUND – The amount of refund requested by the buyer. • <i>transaction_type</i> _CREDIT – The amount AWS credits the buyer. • BALANCE_ADJUSTMENT – An adjustment made by AWS to resolve invoicing issues. • DISBURSEMENT – If the value of action is DISBURSED and the value of balance_impacting is 1, this is the amount paid to the seller. If the value for action is INVOICED, this record negates the parent_billing_event_id record

Column name	Description
	<p>either in full or in part. In this case, the related disbursement disbursement_billing_event_id is shown and the value of balance_impacting is 0.</p> <ul style="list-style-type: none">• DISBURSEMENT_FAILURE – Negates the transaction.

Column name	Description
parent_billing_event_id	<p>When the value of <code>broker_id</code> is <code>AWS_INC</code>, the value of <code>action</code> is <code>DISBURSED</code> or <code>FORGIVEN</code>, and the value of <code>transaction_type</code> is <code>DISBURSEMENT</code>, the <code>parent_billing_event_id</code> refers to the original <code>billing_event_id</code> that initiated this billing event. If <code>action</code> has another value, this field is null.</p> <p>When the value of <code>broker_id</code> is <code>AWS_EUROPE</code>, the <code>parent_billing_event_id</code> refers to the original <code>billing_event_id</code> that initiated this billing event for the following scenarios:</p> <ul style="list-style-type: none"> • The value of <code>action</code> is <code>FORGIVEN</code> or <code>INVOICED</code> and the value of <code>transaction_type</code> is <code>AWS_REV_SHARE</code>, <code>AWS_REV_SHARE_REFUND</code>, or <code>SELLER_REV_SHARE_REFUND</code>. • The value of <code>action</code> is <code>DISBURSED</code> and the value of <code>transaction_type</code> is <code>ANY</code> (excluding <code>DISBURSEMENT_FAILURE</code>). • The value of <code>transaction_type</code> is <code>AWS_TAX_SHARE</code>, <code>AWS_TAX_SHARE_REFUND</code>, <code>SELLER_TAX_SHARE</code>, or <code>SELLER_TAX_SHARE_REFUND</code>. • The value of <code>action</code> is <code>DISBURSED</code> and the value of <code>transaction_type</code> is <code>DISBURSEMENT_FAILURE</code>. <p>When the value of <code>broker_id</code> is <code>AWS_EUROPE</code>, the <code>parent_billing_event_id</code> refers to the original <code>billing_event_id</code> of</p>

Column name	Description
	<p>the previous successful disbursement billing event for the following scenario:</p> <ul style="list-style-type: none"> The value of <code>action</code> is <code>DISBURSED</code> and the value of <code>transaction_type</code> is <code>DISBURSEMENT_FAILURE</code>. <p>When the value of <code>broker_id</code> is <code>AWS_EUROPE</code>, this field is null for all remaining scenarios.</p>
disbursement_billing_event_id	<p>The related disbursement when the value of <code>action</code> is <code>DISBURSED</code> and one of the following is true:</p> <ul style="list-style-type: none"> <code>transaction_type like ('SELLER%')</code> <code>transaction_type like ('AWS%')</code> <p>In all other scenarios, this value is null.</p>
amount	The billing event amount.
currency	The ISO 639 currency code.
balance_impacting	<p>Whether the amount is taken into account in calculating seller disbursements. A value of <code>0</code> indicates the amount is shown for informational purposes and has no effect on the balance. A value of <code>1</code> indicates that this amount takes into account in determining seller disbursements.</p>
invoice_date	The date the invoice was created.
payment_due_date	When the value of <code>action</code> is <code>INVOICED</code> , the due date for the invoice.

Column name	Description
usage_period_start_date	The start date for the period in the record.
usage_period_end_date	The end date for the period in the record.
invoice_id	The AWS invoice ID.
billing_address_id	<p>The payer's billing address reference in the address data feed.</p> <p>Can be used to join to the Address data feed on the <code>address_id</code> field.</p>
transaction_reference_id	<p>An identifier that allows you to cross-reference data from the following reports:</p> <ul style="list-style-type: none">• Disbursement report• Monthly billed revenue report• Sales compensation report• US sales and use tax report
bank_trace_id	For disbursement transactions (<code>transaction_type</code> =is DISBURSEMENT and action is DISBURSED), the trace ID assigned by the bank. The trace ID can be used to correlate with bank-provided reports from the seller bank.

Column name	Description
broker_id	<p>An identifier of the business entity which facilitated the transaction. Possible values are as follows:</p> <ul style="list-style-type: none"> • AWS_INC <ul style="list-style-type: none"> – The identifier for AWS, Inc. (based in the United States). • AWS_EUROPE <ul style="list-style-type: none"> – The identifier for Amazon Web Services EMEA SARL (based in Luxembourg). • NULL <ul style="list-style-type: none"> – Previous transactions without an explicit broker_id were facilitated by AWS_INC.
buyer_transaction_reference_id	<p>An identifier that groups all related records in the billing feed together using GROUP BY or the window functions construct in SQL. These related records can contain the buyer invoice, seller invoice, and value added taxes (VATs) on the listing fee.</p>

Taxing scenarios

The taxation model that is in place for the country and state of the buyer and seller dictates how taxes are collected and remitted. Following are the possible scenarios:

- Taxes are collected and remitted by AWS. In these cases, the transaction_type is AWS_TAX_SHARE.
- Taxes are collected by AWS, disbursed to the seller, and remitted by the seller to the tax authorities. In these cases, the transaction_type is SELLER_TAX_SHARE.

- Taxes are not collected by AWS. The seller must calculate the taxes and remit them to the tax authorities. In these cases, AWS Marketplace doesn't perform tax calculations or receive tax information. The seller pays the taxes from the revenue share.

Examples of billing event data feed

This section shows examples of the billing event data period at the time of invoicing and one month later. Note the following for all tables in this section:

- In data feeds, `billing_event_id` values are 40-character alphanumeric strings. They're shown here as two-character strings for readability.
- In the data feed, this information is presented in a single table. For readability, the data is shown in multiple tables here, and all columns aren't shown.

For the examples in this section, assume the following:

- Arnav is the buyer.
 - His account ID is 737399998888.
 - He's located in France, which is subject to marketplace facilitator laws. For more information, see [Amazon Web Service Tax Help](#).
 - He purchased `prod-o4g1xfafcxxxx` and was invoiced \$120.60 for his monthly usage of that product.
 - He paid the invoice within the month.
- Jane is the manufacturer.
 - Her account ID is 111122223333.
- Paulo is the seller of record.
 - His account ID is 777788889999.
 - He lives in Kansas, which is not subject to market facilitator laws.

Billing event data feed for seller of record

As the seller of record, Paulo invoices the buyer, Arnav.

The following tables show the relevant information in Paulo's data feed when he invoices Arnav.

billing_event_id	from_account_id	to_account_id	end_user_account_id	product_id	action	transaction_type
I0	737399998888	777788889999	737399998888	prod-o4grxfafcxxxx	INVOICED	SELLER_REV_SHARE
I1	737399998888	AWS	737399998888	prod-o4grxfafcxxxx	INVOICED	AWS_TAX_SHARE
I2	777788889999	111122223333	737399998888	prod-o4grxfafcxxxx	INVOICED	SELLER_REV_SHARE
I3	777788889999	AWS	737399998888	prod-o4grxfafcxxxx	INVOICED	AWS_REV_SHARE

parent_billing_event_id	disbursement_billing_event_id	amount	currency	invoice_date	invoice_id
		100	USD	2018-12-31T00:00:00Z	781216640
		20.6	USD	2018-12-31T00:00:00Z	781216640
		-80	USD	2018-12-31T00:04:07Z	788576665
		-0.2	USD	2018-12-31T00:04:07Z	788576665

The following tables show the relevant information in Paulo's data feed at the end of the month, after Arnav pays the invoice.

billing_event_id	from_account_id	to_account_id	end_user_account_id	product_id	action	transaction_type
I10	737399998888	777788889999	737399998888		DISBURSED	SELLER_RE V_SHARE
I12	777788889999	111122223333	737399998888		DISBURSED	SELLER_RE V_SHARE
I13	777788889999	AWS	737399998888	prod-o4gr xfafcxxxx	DISBURSED	AWS_REV_S HARE
I14	AWS	777788889999			DISBURSED	DISBURSEM ENT

parent_billing_event_id	disbursement_billing_event_id	amount	currency	invoice_date	invoice_id
I0	I14	-100	USD	2018-12-3 1T00:00:00Z	781216640
I2	I14	80	USD	2018-12-3 1T00:04:07Z	788576665
I3	I14	0.2	USD	2018-12-3 1T00:04:07Z	788576665
		19.8	USD		

Billing event data feed for manufacturer

The following tables show the relevant information in the Jane's data feed when Paulo invoices Arnav.

billing_event_id	from_account_id	to_account_id	end_user_account_id	product_id	action	transaction_type
15	77778888999	11112222333		prod-o4grxfafcxxxx	INVOICED	SELLER_REV_SHARE
16	77778888999	11112222333		prod-o4grxfafcxxxx	INVOICED	SELLER_TAX_SHARE
17	11112222333	AWS		prod-o4grxfafcxxxx	INVOICED	AWS_REV_SHARE

parent_billing_event_id	disbursement_billing_event_id	amount	currency	invoice_date	invoice_id
		73.5		2018-12-31T00:04:07Z	788576665
		6.5		2018-12-31T00:04:07Z	788576665
		-7.35		2018-12-31T00:04:07Z	788576665

The following tables show the relevant information in Jane's data feed at the end of the month, after the invoice is paid.

billing_event_id	from_account_id	to_account_id	end_user_account_id	product_id	action	transaction_type
130	77778888999	11112222333		prod-o4grxfafcxxxx	DISBURSED	SELLER_REV_SHARE

billing_event_id	from_account_id	to_account_id	end_user_account_id	product_id	action	transaction_type
I31	77778888999	11112222333		prod-o4grxfafcxxxx	DISBURSED	SELLER_TAX_SHARE
I32	11112222333	AWS		prod-o4grxfafcxxxx	DISBURSED	AWS_REV_SHARE
I33	AWS	11112222333			DISBURSED	DISBURSEMENT

parent_billing_event_id	disbursement_billing_event_id	amount	currency	invoice_date	invoice_id
I5	I33	-73.5	USD		
I6	I33	-6.5	USD		
I7	I33	7.35	USD		
		72.65	USD		

Example queries

As described in [Using data feeds](#), you can use [Athena](#) to run queries on the data that's collected and stored as data feeds in your managed Amazon S3 bucket. This section provides some examples of common ways you might do this. All examples assume that a single currency is used.

Example 1: Amount invoiced, including taxes

To find out how much buyers were invoiced, including taxes, you can run a query as shown in the following example.

```
SELECT sum(amount) FROM billing_event
WHERE
  action = 'INVOICED'
```

```

AND
(
  (transaction_type in ('SELLER_REV_SHARE', 'SELLER_TAX_SHARE')
    -- to discard SELLER_REV_SHARE from Manufacturer to Channel Partner, aka cost of
    goods
    AND to_account_id='seller-account-id'
  )
OR transaction_type= 'AWS_TAX_SHARE'
);

```

Example 2: Amount invoiced to buyers on seller's behalf

To find out how much buyers were invoiced on a seller's behalf, you can run a query as shown in the following example.

```

SELECT sum(amount) FROM billing_event
WHERE
  action = 'INVOICED'
  AND transaction_type in ('SELLER_REV_SHARE', 'SELLER_TAX_SHARE')
  AND to_account_id='seller-account-id'
;

```

Example 3: Amount AWS can collect on seller's behalf

To find out how much AWS can collect on a seller's behalf, minus any refunds, credits, and forgiven accounts, you can run a query as shown in the following example.

```

SELECT sum(amount) FROM billing_event
WHERE
  -- what is invoiced on behalf of SELLER, incl. refunds/ credits and cost of goods
  transaction_type like 'SELLER_%'
  -- FORGIVEN action records will "negate" related INVOICED
  and action in ('INVOICED','FORGIVEN')
;

```

Example 4: Amount seller can collect

To find out how much sellers can collect, you can run a query as shown in the following example. This example removes listing fees and taxes that AWS collects, and adds any exceptional balance adjustments.

```

SELECT sum(amount) FROM billing_event

```

```
WHERE
  (transaction_type like 'SELLER_%' -- what is invoiced on behalf of SELLER
  or transaction_type like 'AWS_REV_%' -- what is owed to AWS
  or transaction_type = 'BALANCE_ADJUSTMENT' -- exceptionnal case
  )
  and action in ('INVOICED','FORGIVEN')
;
```

You can also use the following query to collect the same information, as shown in the following example.

```
SELECT sum(amount) FROM billing_event
WHERE
  balance_impacting = 1
  and action in ('INVOICED','FORGIVEN')
;
```

The following example shows the same information, but is restricted to 2018 transactions and assumes all buyers paid their invoices.

```
SELECT sum(amount) FROM billing_event
WHERE
  invoice_date between '2018-01-01' and '2018-12-31'
  and balance_impacting = 1
  and action in ('INVOICED','FORGIVEN')
;
```

Example 5: Amount of disbursements

To find out the amount that's been disbursed, you can run a query as shown in the following example.

```
select sum(amount) FROM billing_event
WHERE
  action = 'DISBURSED'
  and transaction_type like 'DISBURSEMENT%'
;
```

Example 6: Amount pending disbursement

To find out the amount that's pending disbursement, you can run a query as shown in the following example. This query removes amounts that have already been disbursed.

```

SELECT sum(amount) FROM billing_event targeted
WHERE
  (transaction_type like 'SELLER_%' -- what is invoiced on behalf of SELLER
  or transaction_type like 'AWS_REV_%' -- what is owed to AWS
  or transaction_type = 'BALANCE_ADJUSTMENT' -- exceptionnal case
  )
-- DISBURSEMENT action records will "negate" 'INVOICED'
-- but do not take into account failed disbursements
AND
  (not exists
    (select 1
      from billing_event disbursement
      join billing_event failed_disbursement
      on disbursement.billing_event_id=failed_disbursement.parent_billing_event_id
      where
        disbursement.transaction_type='DISBURSEMENT'
        and failed_disbursement.transaction_type='DISBURSEMENT_FAILURE'
        and targeted.disbursement_billing_event_id=disbursement.billing_event_id
    )
  )
;

```

Another way to get the same information is to run a query to get the seller's balance, as shown in the following example.

```

SELECT sum(amount) FROM billing_event
WHERE
  balance_impacting = 1
;

```

The following query extends our example. It restricts the results to 2018 transactions and returns additional details about the transactions.

```

select sum(residual_amount_per_transaction)
from
  (SELECT
    max(billed_invoices.amount) invoiced_amount,
    sum(nvl(disbursed_invoices.amount,0)) disbursed_amount,
    -- Exercise left to the reader:
    -- use transaction_type to distinguish listing fee vs seller-owed money
    -- still pending collection
    max(transaction_type) transaction_type,

```

```

max(billed_invoices.amount)
  + sum(nvl(disbursed_invoices.amount,0)) residual_amount_per_transaction
FROM billing_event billed_invoices
-- find related disbursements
left join billing_event disbursed_invoices
  on disbursed_invoices.action='DISBURSED'
  and disbursed_invoices.parent_billing_event_id=billed_invoices.billing_event_id
WHERE
  billed_invoices.invoice_date between '2018-01-01' and '2018-12-31'
  and billed_invoices.transaction_type like 'SELLER_%' -- invoiced on behalf of
SELLER
  and billed_invoices.action in ('INVOICED','FORGIVEN')
-- do not take into account failed disbursements
AND not exists
  (select 1 from billing_event failed_disbursement
    where disbursed_invoices.disbursement_billing_event_id =
failed_disbursement.parent_billing_event_id
  )
GROUP BY billed_invoices.billing_event_id
);

```

Example 7: Balance of set of invoices

To learn the sum of a set of invoices, you can run a query as shown in the following example.

```

SELECT invoice_id, sum(amount) FROM billing_event targeted
WHERE
-- invoice_id is only not null for invoiced records AND disbursed records
-- linking them to related disbursement -> no need to filter more precisely
invoice_id in ('XXX','YYY')
-- filter out failed disbursements
AND not exists
  (select 1
    from billing_event disbursement
      join billing_event failed_disbursement
        on disbursement.billing_event_id=failed_disbursement.parent_billing_event_id
    where
      disbursement.transaction_type='DISBURSEMENT'
      and failed_disbursement.transaction_type='DISBURSEMENT_FAILURE'
      and targeted.disbursement_billing_event_id=disbursement.billing_event_id
  )
group by invoice_id;

```

Legacy mapping data feed

This data feed lists how product IDs and offer IDs map to legacy globally unique identifiers (GUIDs). The legacy GUIDs were used in older reports, and the new IDs are used in data feeds and in AWS Marketplace APIs.

This data feed provides information about all products you've created as the seller of record and all products you're authorized to resell.

The legacy mapping data feed is refreshed every 24 hours, so new data is available daily.

The following table explains the names and descriptions of the data feed's columns.

Column name	Description
mapping_type	Whether this is a product ID or offer ID.
legacy_id	The legacy ID for this product or offer.
new_id	The friendly ID for this product or offer. This ID is used as the primary key and with all current API actions.

Example of legacy mapping data feed

The following shows an example of the legacy mapping data feed. For readability, the data history columns aren't shown. For information about data history fields, see [Historization of the data](#).

mapping_type	legacy_id	new_id
OFFER	8a806c74-dbd6-403e-9362-bb08f417ff37	offer-dacpxznflfwin
PRODUCT	1368541d-890b-4b6c-9bb9-4a55306ab642	prod-o4grxfafcxy
OFFER	558d8382-6b3a-4c75-8345-a627b552f5f1	offer-gszhmle5npzip

Offer data feed

The offer data feed provides information about all offers that you've created as the seller of record. If a single offer has multiple revisions, all revisions are included in the data feed.

When you make an offer revision and the data in an exposed field changes, a new record is created in the data feed for the same primary key (`offer_id` plus `offer_revision`). However, the `valid_from` field has a different value. For more information about the data feed history columns, see [Historization of the data](#).

The offer data feed is refreshed every 24 hours, so new data is available daily.

The following table provides the names and descriptions of the data feed's columns.

Column name	Description
<code>offer_id</code>	The friendly identifier for the offer. Can be used to join to the <code>offer_id</code> field of the <code>Offer_Product</code> data feed.
<code>offer_revision</code>	The offer revision. This field and the <code>offer_id</code> field combine to form the primary key. With <code>offer_id</code> , can be used to join to the <code>offer_id</code> and <code>offer_revision</code> fields of the <code>Target_Offer</code> data feed.
<code>name</code>	The seller-defined name of the offer.
<code>expiration_date</code>	The date and time that the offer expires.
<code>opportunity_name</code>	Any opportunity data linked to this offer. If the offer is bound to an AWS opportunity, this field is populated.
<code>opportunity_description</code>	Any descriptive information linked to this offer. If the offer is bound to an AWS opportunity, this field is populated.

Column name	Description
seller_account_id	The globally unique identifier (GUID) of the seller's account. Can be used to join with the account_id field in the account data feed.
opportunity_id	An identifier for the opportunity is only populated if a reseller is selling your product. All offers created by different channel partners (or sellers) have the same opportunity_id if the product is the same.

Example of offer data feed

The following shows an example of the offer data feed. For readability, the data history columns aren't shown. For information about data history fields, see [Historization of the data](#).

offer_id	offer_revision	name	expiration_date	opportunity_name	opportunity_description	seller_account_id	opportunity_id
offer-dacpxznflwinn	1	Enterprise Contract Program Offer	9999-01-01T00:00:00Z				
offer-gszhmle5npzip	1	Private offer created by seller	2020-10-31T00:00:00Z				
offer-hmzhyle8nphlp	1	Enterprise Contract Program Offer	9999-01-01T00:00:00Z				

Offer product data feed

One offer can have several products, and one product can be included in different offers. This data feed lists information about the relationships between offers and products.

This data feed provides information about all product offers you've created as the seller of record.

When you add or remove a product from an offer, you create an offer revision.

The offer product data feed is refreshed every 24 hours, so new data is available daily.

The following table explains the names and descriptions of the data feed's columns. For information about the data feed history columns, see [Historization of the data](#).

Column name	Description		
offer_id	The friendly identifier of this offer. Can used to join to the offer_id field of the Offer data feed.		
offer_revision	Combines with offer_id field to form the foreign key to the offer revision.		
product_id	The friendly identifier of the product, this is the foreign key to the product that this offer exposes. Can used to join to the product_id field of the Product data feed.		

Example of Offer product data feed

The following shows an example of the Offer product data feed.

offer_id	offer_revision	product_id
offer-dacpxznflfwin	10	prod-o4grxfafcxxxx
offer-gszhmle5npzip	24	prod-o4grxfafcxyy

Offer target data feed

This data feed lists targets of an offer's revision for all offers you've created as the seller of record. If a single offer has multiple revisions, all revisions are included in the data feed.

When you make an offer revision and the data in an exposed field changes, a new record is created in the data feed for the same primary key (`offer_id` plus `offer_revision`), but with a different value for `valid_from` field.

The offer target data feed is refreshed every 24 hours, so new data is available daily.

The following table explains the names and descriptions of the data feed's columns.

Column name	Description
<code>offer_target_id</code>	The primary key of the feed.
<code>offer_id+offer_revision</code>	The identifier and revision of the offer. These two columns reference the offer that this target relates to. Can used to join to the <code>offer_id</code> and <code>offer_revision</code> fields of the Target data feed.
<code>target_type</code>	Indicates whether the offer recipient is <code>BuyerAccounts</code> , which indicates a private offer, or <code>ParticipatingPrograms</code> .

Column name	Description
polarity	<p>Indicates whether the offer is intended to be made to the <code>target_type</code> . Acceptable values are as follows:</p> <ul style="list-style-type: none"> <code>PositiveTargeting</code> – The offer applies for this <code>target_type</code> . <code>NegativeTargeting</code> – The offer doesn't apply for this <code>target_type</code> .
value	<p>A string that represents the target: either an AWS account ID or a program that can be used with an offer. For example, Standard Contract for AWS Marketplace (SCMP), or AWS Marketplace Field Demonstration Program (FDP).</p>

Example of offer target data feed

The following shows an example of the offer target data feed. For readability, the data history columns aren't shown. For information about data history fields, see [Historization of the data](#).

offer_target_id	offer_id	offer_revision	target_type	polarity	value
925ddc73f6a373b7d5544ea3210610803b600	offer-dacpxznflfwin	1	ParticipatingPrograms	PositiveTargeting	EnterpriseContract
471ff22ae3165278f1fb960d3e14517bcd601	offer-gszhml5npzip	1	ParticipatingPrograms	PositiveTargeting	FieldDemonstration

offer_target_id	offer_id	offer_revision	target_type	polarity	value
511ff22ad fj65278f1 fb960d3e1 4517bcd6e 602	offer-gsz hml5npzip	1	ParticipatingPrograms	PositiveTargeting	EnterpriseContract

Product data feed

This data feed provides information about all products you've created as the seller of record and all products you're authorized to resell.

Product data is mutable. This means that when you change the value for one of the following fields, a new record is created in the data feed with a different value for `valid_from` field. For more information about the data feed history columns, see [Historization of the data](#).

The product data feed is refreshed every 24 hours, so new data is available daily.

The following table explains the names and descriptions of the data feed's columns.

Column name	Description
product_id	The friendly identifier of the product. Can used to join to the <code>product_id</code> fields of the <code>Account</code> , <code>Billing_Event</code> , and <code>Offer_Product</code> data feeds.
manufacturer_account_id	The identifier of the product owner. This is a foreign key to the Account data feed. Can used to join to the <code>account_id</code> field of the <code>Account</code> data feed.
product_code	The existing entitlement product code used to meter the product. This value is also used to join data with a report, or to reference what

Column name	Description
	is provided in AWS Marketplace Metering Service.
title	The title of the product.

Example of product data feed

The following shows an example of the offer target data feed. For readability, the data history columns aren't shown. For information about data history fields, see [Historization of the data](#).

product_id	manufacturer_account_id	product_code	title
prod-o4grxfafcxxxx	555568000000	product_code_1	Product1
prod-t3grxfafcxyy	444457000000	product_code_2	Product2
prod-x8faxxfafcxy	666678000000	product_code_3	Product3

Tax item data feed

This data feed provides information about tax calculations for a customer invoice.

There can be multiple line items (`line_item_id`) for a given product (`product_id`) of a given customer invoice (`invoice_id`), one or more for each tax jurisdiction. This happens, for example, with usage-based bills for customers who are using different AWS Region rules by different AWS entities (say, the US and Ireland). To learn more about where AWS collects sales tax, VAT, or GST on your sales and remits such taxes to the local tax authorities, in the name of AWS, Inc., see [Amazon Web Service Tax Help](#).

The tax item data feed is refreshed every 24 hours, so new data is available daily.

Tax item data is immutable.

The following table explains the names and descriptions of the data feed's columns. For information about data history columns, see [Historization of the data](#).

Column name	Description
tax_item_id	A unique identifier for a tax item record.
invoice_id	The AWS invoice ID. You can use this value with the value of <code>product_id</code> to find related tax billing events.
line_item_id	A unique identifier for a customer bill line item. Refund transactions have the same line item ID as their forward tax transactions.
customer_bill_id	The unique identifier of the customer bill. Buyers can share this identifier with the seller to help identify and resolve tax calculation questions.
tax_liable_party	<p>Either AWS or Seller. If the seller is the tax liable party, taxes are collected. If AWS is the tax liable party, sales tax is collected and remitted by AWS. For more information, see AWS Marketplace Sellers & Tax Collection.</p> <p>If no taxes are collected, there is no value shown here. The seller needs to determine whether some taxes were collected for each invoice, as the seller is liable for tax collection.</p>
transaction_type_code	<p>The type of transaction. Possible values are as follows:</p> <ul style="list-style-type: none"> • AWS – A forward tax transaction • REFUND – A full or partial refund • TAXONLYREFUND – A tax-only refund <p>Refund transactions share the line item ID with their original forward transactions.</p>

Column name	Description
product_id	A foreign key to the product. Can be used to join to the Product data feed on the product_id field.
product_tax_code	A standard code to identify the tax properties for a product. Sellers choose the properties when creating or modifying the product.
invoice_date	The date the invoice was created.
taxed_customer_account_id	A foreign key to the account entity who is taxed. Can be used to join to the Account data feed on the account_id field.
taxed_customer_country	The ISO 3166 alpha 2 country code of the address used for tax calculations.
taxed_customer_state_or_region	The state, region, or province used for tax calculations.
taxed_customer_city	The city used for tax calculations.
taxed_customer_postal_code	The postal code used for tax calculations.
tax_location_code_taxed_jurisdiction	The vertex geocode that is associated with the taxed location.
tax_type_code	The type of tax that is applied to the transaction. The possible values are None, Sales, and SellerUse .
jurisdiction_level	The jurisdiction level of the address that is used for tax location. The possible values are State, County, City, and District.

Column name	Description
taxed_jurisdiction	The name of the tax jurisdiction.
display_price_taxability_type	Whether the price that buyers see is inclusive or exclusive of taxes. All AWS Marketplace offerings are exclusive of taxes.
taxable_amount	The amount of the transaction that is taxable, at this jurisdiction level.
nontaxable_amount	The amount of the transaction that is nontaxable, at this jurisdiction level.
tax_jurisdiction_rate	The tax rate that is applied, at this jurisdiction level.
tax_amount	The amount of tax that is charged, at this jurisdiction level.
tax_currency	The ISO 4217 alpha 3 currency code for above amounts.
tax_calculation_reason_code	Whether the transaction is taxable, not taxable, exempt, or zero-rated, organized by the jurisdiction level.
date_used_for_tax_calculation	The date that is used for calculating tax on the transaction.
customer_exemption_certificate_id	The certificate ID of the exemption certificate.
customer_exemption_certificate_id_domain	The location where the certificate is stored on Amazon systems.
customer_exemption_certificate_level	The jurisdiction level that supplied the exemption.
customer_exemption_code	The code that specifies the exemption; for example, RESALE.

Column name	Description
customer_exemption_domain	The Amazon system that is used to capture the customer exemption information, if available.
transaction_reference_id	An identifier that allows you to cross-reference data from the following reports: <ul style="list-style-type: none"> Disbursement report Monthly billed revenue report Sales compensation report US sales and use tax report

Note

Beginning August 5, 2021, international Marketplace Facilitator taxes for AWS Marketplace sales will have entries in the tax item data feed. This means that, beginning August 5, 2021, every AWS_TAX_SHARE and SELLER_TAX_SHARE record in the billing event data feed is expected to have a corresponding record in the tax item data feed.

Example of tax item data feed

The following shows an example of the tax item data feed. In the data feed, this information is presented in a single table. For readability, the data is shown in multiple tables here, and all columns aren't shown.

tax_item_id	invoice_id	line_item_id	customer_bill_id
6p2ni6tu041xagvhby anbgxl3xameha16txj oav_0001	781216640	710000000 00000000000	221000000 0000000000
6p2ni6tu041xagvhby anbgxl3xameha16txj oav_0002	781216640	530000000 00000000000	221000000 0000000000

tax_item_id	invoice_id	line_item_id	customer_bill_id
flr4jobxjzww8czdsr q4noue2ux d56j39wxw0k7_0001	250816266	764000000 00000000000	572000000 0000000000
gfkjjobxjzw56jgkrs rqjtk52uxd56j39wg j567d_0002	280336288	764000000 00000000000	572439000 0000000000
wwk1qpvb8 ran3geiw8 e3mp6dgs2 qj7wpkuwhgk1_0001	451431024	993000000 00000000000	123000000 0000000000
wwk1qpvb8 ran3geiw8 e3mp6dgs2 qj7wpkuwhgk1_0002	451431024	993000000 00000000000	312000000 0000000000
fnohdid8kwgqq9lvii 2k30spn3ftgwihbe8h 75x_0001	229987654	921000000 00000000000	639000000 0000000000

tax_liable_party	transacti on_type_code	product_id	product_t ax_code	invoice_date
Seller	AWS	prod-o4gr xfafcxxx	AWSMP_SOF TWARE_RA	2018-12-3 1T00:00:00Z
Seller	AWS	prod-o4gr xfafcxxx	AWSMP_SOF TWARE_RA	2018-12-3 1T00:00:00Z
Seller	AWS	prod-t3gr xfafcxxx	AWS_REMOT E_ACCESS_ SOFTWARE	2018-08-3 1T00:00:00Z

tax_liable_party	transacti on_type_code	product_id	product_t ax_code	invoice_date
Seller	REFUND	prod-t3gr xfafcxyy	AWS_REMOT E_ACCESS_ SOFTWARE	2018-08-3 1T00:00:00Z
Seller	AWS	prod-x8fa xxfafcxyy	AWS_REMOT E_ACCESS_ SOFTWARE	2018-08-3 1T00:00:00Z
Seller	TAXONLYRE FUND	prod-x8fa xxfafcxyy	AWS_REMOT E_ACCESS_ SOFTWARE	2018-05-3 1T00:00:00Z
AWS	AWS	prod-wghj 8xfafrhgj	AWS_REMOT E_ACCESS_ SOFTWARE	2019-07-3 1T00:00:00Z

taxed_cus tomer_acc ount_id	taxed_cus tomer_country	taxed_cus tomer_sta te_or_region	taxed_cus tomer_city	taxed_cus tomer_pos tal_code
VleGa2t9j 3MuxioH9w c8lsndXXC gGCGUreeX riocM5	US	GA	MILTON	48573-4839
VleGa2t9j 3MuxioH9w c8lsndXXC gGCGUreeX riocM5	US	GA	MILTON	48573-4839

taxed_customer_account_id	taxed_customer_country	taxed_customer_state_or_region	taxed_customer_city	taxed_customer_postal_code
7nyo5jwTR oPlyX81vx 9ji04eEwT urO1Ff8bi Qi88W8	US	NC	DURHAM	27517-4834
7nyo5jwTR oPlyX81vx 9ji04eEwT urO1Ff8bi Qi88W8	US	NC	DURHAM	27517-4834
7nyo5jwTR oPlyX81vx 9ji04eEwT urO1Ff8bi Qi88W8	US	TX	NOT APPLICABLE	75844-1235
7nyo5jwTR oPlyX81vx 9ji04eEwT urO1Ff8bi Qi88W8	US	TX	HOUSTON	75844-1235
192a04213 13e41f069 b52962ed7 babf71629 1b688	US	CT	NEW HAVEN	06002-2948

tax_location_code_taxed_jurisdiction	tax_type_code	jurisdiction_level	taxed_jurisdiction	display_price_taxability_type	taxable_amount	nontaxable_amount
460473664	Sales	State	GA	Exclusive	100	0
66301164	Sales	County	FULTON	Exclusive	0	100
692938178	SellerUse	State	NC	Exclusive	58.1	523.8
692938178	SellerUse	State	NC	Exclusive	-58.1	523.8
356794387	Sales	State	TX	Exclusive	1105.14	0
528887443	Sales	City	HOUSTON	Exclusive	-36	0
171248162	Sales	State	CT	Exclusive	0	114.55

tax_jurisdiction_rate	tax_amount	tax_currency	tax_calculation_reason_code	date_used_for_tax_calculation
0.206	20.6	USD	Taxable	2018-10-3 1T00:00:00Z
0	0	USD	NonTaxable	2018-10-3 1T00:00:00Z
0.1	5.8	USD	Taxable	2018-07-3 1T00:00:00Z
0.1	-5.8	USD	Taxable	2018-07-3 1T00:00:00Z
0.06	66.3	USD	Taxable	2018-07-3 1T00:00:00Z

tax_jurisdiction_rate	tax_amount	tax_currency	tax_calculation_reason_code	date_used_for_tax_calculation
0.01	-0.36	USD	NonTaxable	2018-07-31T00:00:00Z
0	0	USD	Exempt	2019-06-30T00:00:00Z

Seller reports

AWS Marketplace provides reports that include information about product usage, buyers, billing, and payment information. Reports are available to all registered AWS Marketplace sellers.

Here are some key points about report generation:

- Reports are generated daily, weekly, or monthly, depending on the report.
- Reports are generated at 00:00 UTC and cover through 24:00 UTC of the previous day.
- Reports are generated as .csv files.
- You can configure Amazon SNS to notify you when data is delivered to your encrypted S3 bucket. After you configure notifications, AWS sends notifications to the email address that is associated with the AWS account that you registered with on AWS Marketplace.

For information on how to configure notifications, see [Getting started with Amazon SNS](#) in the *Amazon Simple Notification Service Developer Guide*.

To cancel getting notification emails, contact the [AWS Marketplace Seller Operations](#) team.

- To learn about each report, you can download [sample reports](#).

Accessing reports

AWS Marketplace provides two ways to configure your reports:

- Using an API interface. The [AWS Marketplace Commerce Analytics Service](#) enables you to automatically access the data in your reports through an API interface. You can automate

ingesting your information and download a portion of a report instead of the whole report. The service returns data asynchronously to a file in Amazon Simple Storage Service (Amazon S3) rather than directly as with a traditional API. The data is delivered in a machine-readable format so that you can import or incorporate the data into your systems.

- Using the reports dashboard in the [AWS Marketplace Management Portal](#). This dashboard provides reports for previous reporting periods.

You can control access to reports by using AWS Identity and Access Management (IAM) permissions.

Daily business report

The daily business report helps you understand how AWS customers are using your products on a daily basis and the estimated revenue from that usage. You only receive this report if relevant information is available. If you don't receive this report and think that you should have received it, contact the [AWS Marketplace Seller Operations](#) team.

You can access this report at the [AWS Marketplace Management Portal](#). If you are registered for the [the section called "AWS Marketplace Commerce Analytics Service"](#), you can also access your reports using the AWS SDK.

You can use a unique identifier for each customer to identify customers over time and across reports. The identifier enables you to track customer usage patterns so that you can estimate customer spend, gain insights into free trial usage, and annual usage trends.

Publication schedule

This report is published daily at 00:00 UTC and covers from 00:00 UTC through 23:59 UTC of the previous day. Any exceptions to the schedule are noted at the introduction of the daily business report section.

Topics

- [Section 1: Usage by instance type](#)
- [Section 2: Fees](#)
- [Section 3: Free trial conversions](#)
- [Section 4: New instances](#)
- [Section 5: New product subscribers](#)

- [Section 6: Canceled product subscribers](#)

Section 1: Usage by instance type

This section lists data with a row for each instance type that the customer uses. For instance, when the customer uses a product on one instance type and the same product on a different instance type, the report includes a row for each of the two instance types.

Column name	Description
Customer Reference ID	A unique identifier that isn't the account ID. It helps track usage, revenue, and subscriptions by customers.
User's State	The billing address state that is associated with the account that is subscribed to the product.
User's Country	The two-character country code that is associated with the account that is subscribed to the product. This report uses ISO 3166-1 alpha-2 standard.
Product Title	The title of the product.
Product Code	The unique identifier for the product.
Instance Type	The instance type associated with the product usage: for example, t2.micro.
Usage Units	The number of units of usage that the customer used during the reporting period.
Usage Unit Type	The unit of measurement that meters the customer's usage. For example, hours or days.
Offering Description	The description for product offering. For example, the product is offered for hourly usage, free trial usage, or annual usage.

Column name	Description
Estimated Revenue	The estimated revenue from the product usage. The billing is finalized at the end of the month.
Currency	The currency of the transaction. For example, if the transaction is in US dollars, the entry is USD.
Offer ID	The identifier for the offer that the buyer signed.
Offer Visibility	Whether the offer is a public, private, or enterprise contract offer.
Customer AWS Account Number	The ID of the account that the charges are billed to.
Customer Country	The two-character country code that is associated with the account that the charges are billed to.
Solution Title	The name of the solution.
Solution ID	The unique identifier for the solution.
Payer Reference ID	A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports.
Payer Address ID	A unique identifier that represents the customer's address.

Section 2: Fees

This section includes fee-based transactions that are associated with products: for example, annual, monthly, SaaS contracts product fees, and data product subscription fees. The data in this section covers the 24-hour period 72 hours before the time that the report is generated. For example, if the report is generated on May 24, the data covers the 24-hour period for May 21.

Column name	Description
Customer Reference ID	A unique identifier that isn't the account ID. It helps track usage, revenue, and subscriptions by customers.
User's State	The billing address state that is associated with the account that is subscribed to the product.
User's Country	The two-character country code that is associated with the account that is subscribed to the product. This report uses ISO 3166-1 alpha-2 standard.
Product Title	The title of the product.
Product Code	The unique identifier for the product.
Amount	The usage fee. If there is a refund, this value is negative. If this entry is for an AWS Marketplace SaaS contract, the amount represents the fee for the dimension, not the entire contract.
Currency	The currency of the transaction. For example, if the transaction is in US dollars, the entry is USD.
Fee Description	The reason for the fee: for example, monthly fee, annual fee, or refund.

Column name	Description
Customer AWS Account Number	The ID of the account that the charges are billed to.
Customer Country	The two-character country code that is associated with the account that the charges are billed to. This report uses ISO 3166-1 alpha-2 standard.
Customer State	The billing address state that is associated with the account that the charges are billed to.
Customer City	The billing address city that is associated with the account that charges are billed to.
Customer Zip Code	The billing address zip code that is associated with the account that the charges are billed to.
Customer Email Domain	The email domain that is associated with the account that the charges are billed to. For example, if the email address is <code>liu-jie@example.com</code> , the entry is <code>example.com</code> .
Start Date	The start date for an AWS Marketplace SaaS contract or data product subscription.
End Date	The end date for an AWS Marketplace SaaS contract or data product subscription.
Quantity	The number of units for a dimension that the contract specifies.
Dimension	The dimension that the contract specifies.
Solution Title	The name of the solution.
Solution ID	The unique identifier for the solution.

Column name	Description
Payer Reference ID	A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports.
Payer Address ID	A unique identifier that represents the customer's address.

Section 3: Free trial conversions

This section lists data for free trial starts, conversions and cancellations, and covers the previous 24-hour period.

Column name	Description
Product Title	The title of the product.
Product Code	The unique identifier representing the product.
New Free Trials	The number of new free trials that are initiated in the reporting period.
Total Current Free Trials	The total number of active free trial subscriptions.
Converted Free Trials	The total number of subscriptions that moved from free trial to paid usage during the reporting period.
Non-Converted Free Trials	The total number of subscriptions that ended the free trial and didn't convert to paid usage.
Solution Title	The name of the solution.

Column name	Description
Solution ID	The unique identifier for the solution.

Section 4: New instances

This section lists data for new EC2 instance and instances types, and covers the previous 24-hour period.

Column name	Description
Customer Reference ID	A unique identifier that isn't the account ID. It helps track usage, revenue, and subscriptions by customers.
User's State	The billing address state that is associated with the account that is subscribed to the product.
User's Country	The two-character country code that is associated with the account that is subscribed to the product. This report uses ISO 3166-1 alpha-2 standard.
Product Title	The title of the product.
Product Code	The unique identifier for the product.
Type	The Amazon EC2 instance type.
Count	The number of EC2 instances.
Customer AWS Account Number	The ID of the account that the charges are billed to.
Customer Country	The two-character country code that is associated with the account that the charges

Column name	Description
	are billed to. This report uses ISO 3166-1 alpha-2 standard.
Customer State	The billing address state that is associated with the account that the charges are billed to.
Customer City	The billing address city that is associated with the account that charges are billed to.
Customer Zip Code	The billing address zip code that is associated with the account that the charges are billed to.
Customer Email Domain	The email domain that is associated with the account that the charges are billed to. For example, if the email address is <code>liu-jie@example.com</code> , the entry is <code>example.com</code> .
Solution Title	The name of the solution.
Solution ID	The unique identifier for the solution.
Payer Reference ID	A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports.
Payer Address ID	A unique identifier that represents the customer's address.

Section 5: New product subscribers

This section lists data for new buyers, and covers the previous 24-hour period.

Column name	Description
Customer Reference ID	A unique identifier that isn't the account ID. It helps track usage, revenue, and subscriptions by customers.
User's State	The billing address state that is associated with the account that is subscribed to the product.
User's Country	The two-character country code that is associated with the account subscribed to the product. This report uses ISO 3166-1 alpha-2 standard.
Product Title	The title of the product.
Product Code	The unique identifier for the product.
Offer ID	The identifier for the offer the buyer signed.
Offer Visibility	Whether the offer is a public, private, or enterprise contract offer.
Customer Country	The two-character country code that is associated with the account that the charges are billed to. This report uses ISO 3166-1 alpha-2 standard.
Customer State	The billing address state that is associated with the account that the charges are billed to.
Customer City	The billing address city that is associated with the account that charges are billed to.
Customer Zip Code	The billing address zip code that is associated with the account that the charges are billed to.
Customer Email Domain	The email domain that is associated with the account that the charges are billed to. For

Column name	Description
	example, if the email address is liu-jie@example.com , the entry is example.com .
Solution Title	The name of the solution.
Solution ID	The unique identifier for the solution.
Payer Reference ID	A unique identifier that isn't the account. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports.
Payer Address ID	A unique identifier that represents the customer's address.

Section 6: Canceled product subscribers

This section lists data for buyer cancellations, and covers the previous 24-hour period.

Column name	Description
Customer Reference ID	A unique identifier that isn't the account ID. It helps track usage, revenue, and subscriptions by customers.
User's State	The billing address state that is associated with the account that is subscribed to the product.
User's Country	The two-character country code that is associated with the account that is subscribed to the product. This report uses ISO 3166-1 alpha-2 standard.
Product Title	The title of the product.

Column name	Description
Product Code	The unique identifier for the product.
Subscribed Date	The date when the subscription started.
Offer ID	The identifier for the offer that the buyer signed.
Offer Visibility	Whether the offer is a public, private, or enterprise contract offer.
Customer AWS Account Number	The ID of the account that the charges are billed to.
Customer Country	The two-character country code that is associated with the account that the charges are billed to. This report uses ISO 3166-1 alpha-2 standard.
Customer State	The billing address state that is associated with the account that the charges are billed to.
Customer City	The billing address city that is associated with the account that charges are billed to.
Customer Zip Code	The billing address zip code that is associated with the account that the charges are billed to.
Customer Email Domain	The email domain that is associated with the account that the charges are billed to. For example, if the email address is <code>liu-jie@example.com</code> , the entry is <code>example.com</code> .
Solution Title	The name of the solution.
Solution ID	The unique identifier for the solution.

Column name	Description
Payer Reference ID	A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports.
Payer Address ID	A unique identifier that represents the customer's address.

Daily customer subscriber report

This report lists data for customers who purchased your products. This report doesn't specify current or past usage, only that a customer is subscribed to your product. You only receive this report if relevant information is available. If you don't receive this report and think that you should have, contact the [AWS Marketplace Seller Operations](#) team.

You can access this report at the [AWS Marketplace Management Portal](#). If you are registered for the [the section called "AWS Marketplace Commerce Analytics Service"](#), you can also access your reports using the AWS SDK.

The report has two sections: one for hourly and monthly subscriptions and one for annual subscriptions. The report includes the list of AWS account IDs for all customers who are subscribed to your products.

Publication schedule

This report is published daily at 00:00 UTC and covers from 00:00 UTC through 23:59 UTC of the previous day.

Topics

- [Section 1: Hourly and monthly subscriptions](#)
- [Section 2: Variable length subscriptions](#)

Section 1: Hourly and monthly subscriptions

This section lists data for all usage-based subscriptions as of the previous day at 23:59:59 UTC.

Column name	Description
Customer AWS Account Number	The account that is subscribed to the product.
Product Title	The title of the product.
Product Id	A unique identifier for the software product.
Product Code	The unique identifier for the software product.
Subscription Start Date	The start date for the subscription, formatted as YYYY-MM-DD .
Offer ID	The identifier for the offer that the buyer signed.
Offer Visibility	Whether the offer is a public, private, or enterprise contract offer.
Solution Title	The name of the solution.
Solution ID	The unique identifier for the solution.
Payer Reference ID	A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports.
Reseller account ID	The unique identifier for the channel partner reseller.
Reseller account name	The name of the channel partner reseller.

Section 2: Variable length subscriptions

This section lists data for all fee-based subscriptions as of the previous day at 23:59:59 UTC.

Column name	Description
Customer AWS Account Number	The ID of the account that is subscribed to the product.
Product Title	The title of the product.
Product Id	The unique identifier for the software product.
Product Code	A unique identifier for the software product. This information is also available as part of the Amazon EC2 instance metadata.
Subscription Id	The ID for the subscription.
Subscription Quantity	The total number of licenses that the customer purchased.
Subscription Type	The type of subscription.
Subscription Intent	Whether this offer is an upgrade or renewal of an earlier offer.
Offer ID	The identifier for the offer that the buyer signed.
Subscription Start Date	The date when the customer subscribed to the product, formatted as YYYY-MM-DD .
Previous Offer ID	The ID of the offer that preceded the upgrade or renewal offer, if one exists.
Offer Visibility	Whether the offer is a public, private, or enterprise contract offer.
Solution Title	The name of the solution.
Solution ID	The unique identifier for the solution.

Column name	Description
Payer Reference ID	A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports.
Reseller account ID	The unique identifier for the channel partner reseller.
Reseller account name	The name of the channel partner reseller.

Disbursement report

The disbursement report provides information about funds that we collected and disbursed to your bank accounts since the previous disbursement. Disbursements can include customer payments or refunds for a subscription to your product, and some taxes collected or refunded to the customer. You don't receive disbursement of funds until the funds are collected from the customer. Different customers have different payment terms with AWS, so some of the funds in each of the uncollected age categories might not be due from the customer.

Refunds appear as negative amounts because the money is returned to your customer after you authorize a refund.

This report is available on the AWS Marketplace Management Portal under the **Reports** tab. To create your own report similar to this one you can use the AWS Marketplace [Seller delivery data feeds service](#), including the [Example 1: Disbursements by product](#) as a base report to customize to meet your needs.

Publication schedule

This report is published 3-5 days after a disbursement has been initiated to transfer funds to your bank. In general, this is a report for sellers who receive disbursements on a monthly cadence. If there is no disbursement initiated, no disbursement report is generated.

Topics

- [Section 1: Disbursed amount by product](#)

- [Section 2: Disbursed amount by customer geography](#)
- [Section 3: Disbursed amount by instance hours](#)
- [Section 4: Age of uncollected funds](#)
- [Section 5: Age of disbursed funds](#)
- [Section 6: Age of past due funds](#)
- [Section 7: Uncollected funds breakdown](#)

Section 1: Disbursed amount by product

This section lists data for disbursements by product.

Column name	Description
Product	The title of the product.
Product Code	The unique identifier for the product.
SellerRev	The amount that is billed to the customer for the usage or fees of the product.
AWSRefFee	The amount of the AWS Marketplace fee.
SellerRevRefund	The amount of the subscription cost that is refunded to customers if any refunds were processed during the data coverage period.
AWSRefFeeRefund	The amount of the AWS Marketplace fee that is refunded if any refunds were processed during the data coverage period.
SellerRevCredit	The AWS credits that AWS Marketplace placed on the customer's account.
AWSRefFeeCredit	The AWS credits that AWS Marketplace placed on your account.
Net Amount	The total funds that we disbursed to you. This column is equal to the SellerRev column minus

Column name	Description
	the AWSRefFee column. When a refund is given to a customer, this column is a negative number equal to the SellerRevRefund column minus the AWSRefFeeRefund column.
Transaction Reference ID	A unique identifier for the transaction that helps you correlate transactions across AWS Marketplace reports.
SellerUSSalesTax	The total amount of US sales and use tax that is billed for this transaction.
SellerUSSalesTaxRefund	The total amount of US sales and use tax that is refunded for this transaction if a refund was processed.
Customer AWS Account Number	The ID of the account that the charges are billed to.
Customer Country	The two-character country code that is associated with the account that the charges are billed to. This report uses ISO 3166-1 alpha-2 standard.
Customer State	The billing address state that is associated with the account that the charges are billed to.
Customer City	The billing address city that is associated with the account that charges are billed to.
Customer Zip Code	The billing address postal code that is associated with the account that the charges are billed to.

Column name	Description
Customer Email Domain	The email domain that is associated with the account that the charges are billed to. For example, if the email address is liu-jie@example.com , the entry is example.com .
Solution Title	The name of the solution.
Solution ID	The unique identifier for the solution.
Payer Reference ID	A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports.
Payer Address ID	A unique identifier that represents the customer's address.

Section 2: Disbursed amount by customer geography

This section lists data for disbursements by the customer's geographic location.

Column name	Description
Settlement ID	The unique identifier of the disbursement.
Settlement Period Start Date	The starting date and time of the disbursement period.
Settlement Period End Date	The ending date and time of the disbursement period.
Deposit Date	The date and time when the disbursement occurred.
Disbursed Amount	The total amount of the disbursement.

Column name	Description
Country Code	The two-character country code that is associated with the account that the charges are billed to. This report uses ISO 3166-1 alpha-2 standard.
State or Region	The billing address state that is associated with the account that the charges are billed to.
City	The billing address city that is associated with the account that charges are billed to.
Postal Code	The billing address postal code that is associated with the account that the software charges are billed to.
Net Amount by Tax Location	The total funds that are disbursed to the seller by tax location, less AWS Marketplace fees, refunds, and US sales and use tax.
Gross Amount by Tax Location	The total funds that are disbursed to the seller by tax location.
Seller U.S. Sales Tax	The total amount of US sales and use tax that is billed for this transaction on behalf of the Seller. (That is, related records in US Sales and Tax reports show "tax liable party" == "SELLER".)
Seller U.S. Sales Tax Refund	The total amount of US sales and use tax that is refunded for this transaction if a refund was processed, when such taxes were collected on behalf of the Seller. (That is, related records in US Sales and Tax reports show "tax liable party" == "SELLER".)

Section 3: Disbursed amount by instance hours

This section lists data for disbursements by Amazon EC2 instance hours.

Column name	Description
Product	The title of the product.
Product Code	The unique identifier for the product.
Usage Type Description	The description of the usage, including offer type, Region, and instance type.
Rate	The rate per hour for the offer type, Region, and instance type.
User Count	The number of unique customers using the offer type, Region, and instance type.
Instance Hours	The number of hours that the instance consumed for the offer type, Region, and instance type.
Solution Title	The name of the solution.
Solution ID	The unique identifier for the solution.

Section 4: Age of uncollected funds

This section lists data for uncollected funds, organized by the age. Uncollected funds might include amounts that aren't due yet.

Column name	Description
Uncollected (< 31 days pending)	The total of funds billed but not collected for less than 31 days.
Uncollected (31–60 days pending)	The total of funds billed but not collected for between 31–60 days.

Column name	Description
Uncollected (61–90 days pending)	The total of funds billed but not collected for between 61–90 days.
Uncollected (91–120 days pending)	The total of funds billed but not collected for between 91–120 days.
Uncollected (> 120 days pending)	The total of funds billed but not collected for more than 120 days.
Uncollected (overall)	The total of all funds billed but not collected.

Section 5: Age of disbursed funds

This section lists data for collected funds since the previous disbursement.

Column name	Description
Collected (< 31 days pending)	The total of funds collected that were billed in the 0–31 day range.
Collected (31–60 days pending)	The total of funds collected that were billed in the 31–60 day range.
Collected (61–90 days pending)	The total of funds collected that were billed in the 61–90 days range.
Collected (91–120 days pending)	The total of funds collected that were billed in the 91–120 days range.
Collected (> 120 days pending)	The total of funds collected that were billed in the greater than 120 days range.
Collected (overall)	The total of all collected funds.

Section 6: Age of past due funds

This section lists data for funds that have been accrued and are payable by the customer, but have not been paid in accordance with the customer's agreement with AWS.

Column name	Description
Past Due (< 31 days)	The total of funds that have accrued in the last 0–31 days and are due but that the customer hasn't paid.
Past Due (31–60 days)	The total of funds that have accrued in the last 31–60 days and are due but that the customer hasn't paid.
Past Due (61–90 days)	The total of funds that have accrued in the last 61–90 days that are due but that the customer hasn't paid.
Past Due (91–120 days)	The total of funds that have accrued in the last 91–120 days and are due but that the customer hasn't paid.
Past Due (> 120 days)	The total of funds that have accrued in the last 121 or more days and are due but that the customer hasn't paid.
Past Due (overall)	The total of funds that have accrued and are due but that the customer hasn't paid.

Section 7: Uncollected funds breakdown

This section lists all uncollected funds, sorted by the payment due date.

Column name	Description
Payer AWS Account Number	The account that the software charges are billed to.

Column name	Description
Product Code	The unique identifier for the product.
Gross Revenue	The amount that is billed for using the product or the fees for using the product.
AWS Revenue Share	The AWS fee amount that is deducted from the billed amount at settlement time.
Gross Refunds	The total amount of any refunds for the transaction.
AWS Refunds Share	The portion of the AWS fee that is refunded for the transaction.
Net Revenue	The net amount that is billed for this transaction, minus AWS fees, refunds, and US sales and use tax.
Currency	The currency of the transaction. For example, if the transaction is in US dollars, the entry is USD.
AR Period	The month and year of the transaction, in the format of YYYY-MM.
Transaction Reference ID	A unique identifier that represents the transaction, which you can use to correlate transactions across AWS Marketplace reports.
Opportunity Name	The unique identifier for a registered opportunity.
Opportunity Description	Any metadata in the registered opportunity.
Solution Title	The name of the solution.
Solution ID	The unique identifier of the solution.

Column name	Description
Payer Reference ID	A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports.
Payer Address ID	A unique identifier that represents the customer's address.
Payment Due date	The payment due date in the format of YYYY-MM-DD .

Monthly billed revenue report

The monthly billed revenue report provides you authoritative information about billed revenue every month for accounting and other financial reporting purposes. This report shows the total amounts that AWS bills to customers for hourly, annual, or monthly usage of your products. The report has four sections: billed amounts for hourly usage and monthly fees, variable-length subscriptions, field demonstration usage, and flexible payments.

Important

The amounts in this report reflect only revenue that we billed to customers, not amounts that we collected.

This report is available on the AWS Marketplace Management Portal under the **Reports** tab. If you're enrolled in the AWS Marketplace commerce analytics service, you can use API calls to pull down sections of this report. For more information, see [the section called "AWS Marketplace Commerce Analytics Service"](#).

Publication schedule

This report is published monthly on the fifteenth day of each month at 00:00 UTC.

Billing and Revenue Data covers from 15th to 14th of next month.

Long-term Subscriptions covers the first day of the month at 00:00 UTC through the last day of the month at 23:59 UTC of the previous month.

For example, the report that is published on May 15 covers from April 1 at 00:00 UTC through April 30 at 23:59 UTC.

Topics

- [Section 1: Billing and revenue data](#)
- [Section 2: Variable length subscriptions](#)
- [Section 3: AWS field demonstration usage](#)
- [Section 4: Contracts with flexible payment schedule](#)

Section 1: Billing and revenue data

This section lists data for usage billing, refunds, fees, and US sales and use tax that is collected.

Column name	Description
Customer Reference ID	A unique identifier that isn't the account ID. It helps track usage, revenue, and subscriptions by customers.
Country	The two-character country code that is associated with the account that the charges are billed to. This report uses ISO 3166-1 alpha-2 standard.
State	The billing address state that is associated with the account that the charges are billed to.
City	The billing address city that is associated with the account that charges are billed to.
Zip Code	The billing address postal code that is associated with the account that the charges are billed to.
Product Title	The title of the product.

Column name	Description
Product Code	The unique identifier for the product.
Customer Billed Amount	The amount that is billed to the customer for the usage or monthly fees of the product.
AWS Listing Fee	The AWS Marketplace fee amount to be deducted from the billed amount.
Refunds Amount	The total amount of the subscription cost refunded to customers if any refunds were processed during the data coverage period.
AWS Fee Refund	The portion of the AWS Marketplace fee refunded if any refunds were processed during the data coverage period.
Cost	The cost of goods to a reseller: for example, what a reseller pays you when they sell your product.
Partner Revenue Amount	The total amount billed for the transaction, net of AWS Marketplace fees, refunds, and US sales and use tax.
Currency	The currency of the transaction. For example, if the transaction is in US dollars, the entry is USD.
Transaction Reference ID	A unique identifier for the transaction that helps you correlate transactions across AWS Marketplace reports.
U.S. Sales Tax Customer Billed Amount	The total amount of US sales and use tax that is billed for this transaction on behalf of the Seller. (That is, related records in US Sales and Tax reports show "tax liable party" == "SELLER".)

Column name	Description
U.S. Sales Tax Refunds Amount	The total amount of US sales and use tax that is refunded for this transaction if a refund was processed, when such taxes were collected on behalf of the Seller. (That is, related records in US Sales and Tax reports show "tax liable party" == "SELLER".)
Offer ID	The identifier for the offer that the buyer signed.
Offer Visibility	Whether the offer is a public, private, or enterprise contract offer.
Customer AWS Account Number	The ID of the account that the charges are billed to.
Customer Email Domain	The email domain that is associated with the account that the charges are billed to. For example, if the email address is <i>liu-jie@example.com</i> , the entry is <i>example.com</i> .
Opportunity Name	The unique identifier for a registered opportunity.
Opportunity Description	The metadata for the registered opportunity.
Solution Title	The name of the solution.
Solution ID	The unique identifier for the solution.
Payer Reference ID	A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports.

Column name	Description
Payer Address ID	A unique identifier that represents the customer's address.

Section 2: Variable length subscriptions

This section lists data for fee-based charges.

Column name	Description
Customer Reference ID	A unique identifier that isn't the account ID. It helps track usage, revenue, and subscriptions by customers.
Country	The two-character country code that is associated with the account that the charges are billed to. This report uses ISO 3166-1 alpha-2 standard.
State	The billing address state that is associated with the account that the charges are billed to.
City	The billing address city that is associated with the account that charges are billed to.
Zip Code	The billing address zip code that is associated with the account that the charges are billed to.
Product Title	The title of the product.
Product Code	The unique identifier for the product.
Subscription Quantity	The number of total licenses that is specified as part of the variable-length subscription purchase.

Column name	Description
Subscription Start Date	The start date of the variable-length subscription purchase.
Subscription End Date	The end date of the variable-length subscription purchase.
Subscription Instance Type	The instance type that is associated with the variable-length subscription purchase.
Customer Billed Amount	The amount that is billed for the usage, monthly fees, or both.
AWS Listing Fee	The AWS Marketplace fee amount that is deducted from the billed amount.
Refunds Amount	The total amount refunded to customers if any refunds were processed during the data coverage period.
AWS Fee Refund	The portion of the AWS Marketplace fee refunded if any refunds were processed during the data coverage period.
Cost	The cost of goods to a reseller: for example, what a reseller pays you when they sell your product.
Partner Revenue Amount	The total amount that is billed for this transaction, net of AWS Marketplace fees, refunds, and US sales and use tax.
Currency	The currency of the transaction. For example, if the transaction is in US dollars, the entry is USD.

Column name	Description
Transaction Reference ID	A unique identifier for the transaction that helps you correlate transactions across AWS Marketplace reports.
U.S. Sales Tax Customer Billed Amount	The total amount of US sales and use tax that is billed for this transaction on behalf of the Seller. (That is, related records in US Sales and Tax reports show "tax liable party" == "SELLER".)
U.S. Sales Tax Refunds Amount	The total amount of US sales and use tax that is refunded for this transaction if a refund was processed, when such taxes were collected on behalf of the Seller. (That is, related records in US Sales and Tax reports show "tax liable party" == "SELLER".)
Customer AWS Account Number	The ID of the account that the charges are billed to.
Customer Email Domain	The email domain that is associated with the account that the charges are billed to. For example, if the email address is liu-jie@example.com, the entry is example.com.
Offer ID	The identifier for the offer that the buyer signed.
Offer Visibility	Whether the offer is a public, private, or enterprise contract offer.
Contract Start Date	The start date for an AWS Marketplace SaaS contract.
Contract End Date	The end date for an AWS Marketplace SaaS contract.

Column name	Description
Opportunity Name	The unique identifier for a registered opportunity.
Opportunity Description	The metadata for the registered opportunity.
Solution Title	The name of the solution.
Solution ID	The unique identifier for the solution.
Payer Reference ID	A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports.
Payer Address ID	A unique identifier that represents the customer's address.

Section 3: AWS field demonstration usage

The section lists data for AWS [field demonstration usage](#) of your product. You can configure your product to allow us to demonstrate your product to potential customers. Any usage from the demonstrations is listed here.

Column name	Description
Product Title	The title of the product.
Product Code	The unique identifier for the product.
Instance Type	The Amazon EC2 instance type that is associated with the field demonstration.
Usage Units	The number of units of usage that is associated with the product.

Column name	Description
Usage Unit Types	The usage units that are associated with the usage unit count: for example, hours.

Section 4: Contracts with flexible payment schedule

This section lists data for all contracts that you created with a flexible payment schedule in the previous reporting period.

Column name	Description
Customer AWS Account Number	The ID of the payer account that the charges are billed to.
Customer Country	The two-character country code that is associated with the payer account that the charges are billed to. This report uses ISO 3166-1 alpha-2 standard.
Customer State	The billing address state that is associated with the payer account that the charges are billed to.
Customer City	The billing address city that is associated with the payer account that charges are billed to.
Customer ZIP Code	The billing address zip code that is associated with the payer account that the charges are billed to.
Customer Email Domain	The email domain that is associated with the payer account that the charges are billed to. For example, if the email address is <i>liu-jie@example.com</i> , the entry is <i>example.com</i> .

Column name	Description
User Reference ID	The account of the payer account that the charges are billed to.
User AWS Account Number	The ID of the account that subscribed to the product.
Product ID	The unique identifier for the product.
Product Title	The title of the product.
Product Type	The type of product.
AWS Marketplace Offer ID	The identifier for the offer that the buyer signed.
Contract Create Date	The contract creation date, which is the date that an account subscribes to the offer.
Contract Expiration Date	The date when the contract expires.
Total Contract Value (USD)	The total value of the contract in USD.
# of Payments	The number of payments that are scheduled for the contract.
Invoice Date	The date the invoice is created.
Invoice Amount (USD)	The amount that is billed on the invoice in USD.
Payer Reference ID	A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports.

Sales compensation report

The report lists monthly billed revenue with additional customer information that isn't in the standard [the section called "Monthly billed revenue report"](#). The report lists the total amounts that AWS bills to customers for hourly, annual, or monthly usage of your product.

Important

The amounts in this report reflect only revenue that is billed to customers, not amounts that are collected from customers.

The sales compensation report and the information that is shared with you as part of this program constitute Amazon's Confidential Information under our nondisclosure agreement with you or, if no such agreement exists, the Terms and Conditions for AWS Marketplace sellers. You can use this information only for compensating your sales representatives by mapping AWS Marketplace revenue to the representatives by company name, geography, and AWS account ID. You can share this information with employees who need to know it to understand the source of commissions that is payable to them. Your use and sharing of such information must comply with the obligations in our nondisclosure agreement with you and the terms and conditions for AWS Marketplace sellers, including, without limitation, Section 3.8 of the Terms and Conditions for AWS Marketplace sellers.

To create your own report similar to this one you can use the AWS Marketplace [Seller delivery data feeds service](#), including the [Example 2: Sales compensation report](#) as a base report to customize to meet your needs.

Publication schedule

This report is published monthly, on the fifteenth day of each month at 00:00 UTC. The report covers the previous calendar month from 00:00 UTC through 23:59 UTC of the last calendar day of the month. For example, the report published on May 15 covers from April 1 at 00:00 UTC through April 30 at 23:59 UTC.

Billed revenue

The billed revenue section of this report includes usage and fee-based charges from the previous calendar month. The following are the column names and descriptions.

Note

In this report, *listing fee* is the percentage of transaction proceeds (except for those from resale by authorized resellers of authorized resale products) determined in accordance with the tiered listing fee.

Column name	Description
Customer AWS Account Number	The account that the charges are billed to.
Country	The two-character country code that is associated with the account that the charges are billed to. This report uses ISO 3166-1 alpha-2 standard.
State	The billing address state that is associated with the account that the charges are billed to.
City	The billing address city that is associated with the account that the charges are billed to.
Zip Code	The billing address zip code that is associated with the account that the charges are billed to.
Email Domain	The email domain that is associated with the account that the charges are billed to. For example, if the email address is <code>liu-jie@example.com</code> , the entry is <code>example.com</code> .
Product Code	The unique identifier for the product.
Product Title	The title of the product.
Gross Revenue	The amount that is billed for using the product or the monthly fees for using the product.

Column name	Description
AWS Revenue Share	The AWS fee amount that is deducted from the billed amount at settlement time. It appears in the the section called "Disbursement report" .
Gross Refunds	The total amount of any refunds for the transaction.
AWS Refunds Share	The portion of the AWS fee that is refunded for the transaction.
Net Revenue	The net amount that is billed for this transaction, minus AWS fees, refunds, and US sales and use tax.
Currency	The currency of the transaction. For example, if the transaction is in US dollars, the entry is USD.
AR Period	The month and year of the transaction, in the format of YYYY-MM.
Transaction Reference ID	A unique identifier that represents the transaction, which you can use to correlate transactions across AWS Marketplace reports.
Opportunity Name	The unique identifier for a registered opportunity.
Opportunity Description	Any metadata in the registered opportunity.
Solution Title	The name of the solution.
Solution ID	The unique identifier of the solution.

Column name	Description
Payer Reference ID	A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports.
Payer Address ID	A unique identifier that represents the customer's address.

US sales and use tax report

This monthly report provides sellers with information about US sales that AWS collects from sales and use transactions in AWS Marketplace. AWS doesn't collect or calculate US sales taxes for the seller.

You will see amounts that AWS has collected and remitted to the tax authority as AWS, based on our internal tax decisions. For more information, see [AWS Marketplace Sellers & Tax Collection](#) on AWS Marketplace Tax Help for Sellers.

To map transactions between the disbursement report and this report, use the Transaction Reference ID.

This report is available on the AWS Marketplace Management Portal under the **Reports** tab. If you're enrolled in the AWS Marketplace Commerce Analytics service, you can use API calls to pull down sections of this report. For more information, see [the section called "AWS Marketplace Commerce Analytics Service"](#).

Publication schedule

This report is published monthly on the fifteenth day of each month at 00:00 UTC. The report covers the previous calendar month from the first day of the month at 00:00 UTC through the last day of the month at 23:59 UTC. For example, the report that is published on May 15 covers from April 1 at 00:00 UTC through April 30 at 23:59 UTC.

US sales and use tax records

This section lists data for US sales tax amounts that result from software charges.

Column name	Description
Line Item ID	A unique identifier for a line item. Refund transactions have the same line item ID as their forward tax transactions.
Customer Bill ID	The unique identifier for a customer bill.
Product Name	The name of the product purchased.
Product Code	The unique identifier for the product.
Product Tax Code	A standard code to identify the tax properties for a product. You choose the properties when you create or modify the product.
Seller ID	A unique identifier for the seller of record of the transaction.
Seller Name	The legal name of the seller.
Transaction Date	The date of the transaction.
Total Adjusted Price	The final price for the transaction.
Total Tax	The total tax that is charged for the transaction.
Base Currency Code	The base currency code for all AWS Marketplace transactions. This entry is always USD.
Bill to City	The billing address city that is associated with the payer account that we bill software charges to.
Bill to State	The billing address zip code that is associated with the payer account that the software charges are billed to.

Column name	Description
Bill to Postal Code	The billing address postal code that is associated with the payer account that the software charges are billed to.
Bill to Country	The two-character country code that is associated with the payer account that the software charges are billed to. This report uses ISO 3166-1 alpha-2 standard.
Transaction Type Code	<p>The type code of the transaction. Valid values:</p> <ul style="list-style-type: none"> • AWS: A forward tax transaction • REFUND: A full or partial refund • TAXONLYREFUND : A tax-only refund <p>Refund transactions share the line item ID with their original forward transactions.</p>
Display Price Taxability Type	The taxability type for the price that appears to customers. All AWS Marketplace offerings are exclusive.
Tax Location Code Taxed Jurisdiction	The vertex geocode that is associated with the taxed location.
Tax Type Code	The type of tax that is applied to the transaction. The possible values are None, Sales, and SellerUse .
Jurisdiction Level	The jurisdiction level of the address that is used for tax location. The possible values are State, County, City, and District.
Taxed Jurisdiction	The name of the taxed jurisdiction.

Column name	Description
Taxable Sale Amount	The amount of the transaction that is taxable, by jurisdiction level.
Nontaxable Sale Amount	The amount of the transaction that is nontaxable, by jurisdiction level.
Tax Amount	The tax that is charged at the jurisdiction level.
Tax Jurisdiction Tax Rate	The tax rate that is applied at the jurisdiction level.
Tax Calculation Reason Code	Whether the transaction is taxable, not taxable, exempt, or zero-rated, organized by the jurisdiction level.
Date Used For Tax Calculation	The date that is used for calculating tax on the transaction.
Customer Exemption Certificate ID	The certificate ID of the exemption certificate.
Customer Exemption Certificate ID Domain	Where the certificate is being stored in Amazon systems.
Customer Exemption Certificate Level	The jurisdiction level that supplied the exemption.
Customer Exemption Code	The code that specifies the exemption: for example, RESALE.
Customer Exemption Domain	The Amazon system that is used to capture the customer exemption information, if information is available.
Customer Reference ID	A unique identifier that isn't the account ID. It helps track usage, revenue and subscriptions by customers.

Column name	Description
Transaction Reference ID	A unique identifier for the transaction that helps you correlate transactions across AWS Marketplace reports.
Payer Reference ID	A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports.
Tax Liable Party	This field will either be populated with Seller or AWS. If the seller is the tax liable party, they are responsible for their own collection and remittance obligations based on their tax decision. If AWS is the tax liable party, sales tax will be collected and remitted to the tax authority by AWS. For more information, see AWS Marketplace Sellers & Tax Collection on Amazon Web Services Tax Help.

Supplementary reports

AWS Marketplace delivers supplementary reports through the [Seller delivery data feeds service](#) to seller-owned Amazon S3 accounts that are connected to the AWS Marketplace Seller Account ID associated with the AWS Marketplace listings for sellers. For more information, refer to [Create a destination Amazon S3 bucket](#).

The supplementary reports are published daily at 16:00 UTC if there were new subscribers in the day prior. These reports cover the previous day from 13:59 UTC through 16:01 UTC of the following day.

Agreement details report

The agreement details report helps you support the customers that are on a software as a service (SaaS) contract free trial. The report includes agreement details such as the subscriber name, subscriber ID, offer ID, agreement start, and agreement end date.

You only receive this report if relevant information is available. If you don't receive this report on an occasion when you think that you should, contact the [AWS Marketplace Seller Operations](#) team.

You can access this report through Amazon S3 bucket associated with the AWS Marketplace Seller Account ID.

The following table lists the column names and descriptions for the agreement details report.

SaaS contract free trial report data

Name	Description
vendor_display_name	The name of the vendor that sold the product.
vendor_aws_account_id	The identification associated with the vendor that sold the product.
subscriber_aws_account_id	The identification associated with the AWS account that is subscribed to the product.
customer_id	The unique identifier for the software product.
product_title	The title of the product.
offer_id	The identifier for the offer that the buyer signed.
offer_visibility	Indication of whether the offer is a public, private, or enterprise contract offer.
reseller_name	The name of the channel partner reseller.
reseller_aws_account_id	The unique identifier for the channel partner reseller.

Name	Description
agreement_id	A unique agreement data feed reference for the agreement signed between a proposer and an acceptor to start using a product.
agreement_acceptance_date	The date the agreement was accepted.
agreement_start_date	The start date of the agreement.
agreement_end_date	The end date of the agreement. For metered/ pay as go/subscriptions, this is set to 1-JAN-9999.
is_free_trial_offer	A flag that indicates if the offer or agreement is a free trial offer.
is_upgraded_after_free_trial	A flag that indicates if the agreement was upgraded to a paid contract.
total_contract_value	The total value of the contract.

Seller dashboards

AWS Marketplace provides dashboards powered by [Amazon QuickSight](#) with charts, graphs, and insights that help you access and analyze financial and sales data. The seller dashboards include:

[the section called “Dashboards for finance operations”](#)

- [the section called “Billed revenue dashboard”](#) – Provides information about billed revenue for accounting and other financial reporting purposes.
- [the section called “Collections and disbursement dashboard”](#) – Provides information about funds that AWS collected and disbursed to your bank accounts since the previous disbursement.
- [the section called “Taxation dashboard”](#) – Provides information about taxes for seller transactions.

the section called “Dashboards for sales operations”

- [the section called “Agreements and renewals dashboard”](#) – Provides information about agreements and renewals within 24 hours of signing an agreement in AWS Marketplace.
- [the section called “Usage dashboard”](#) – Provides visualizations and fine-grained data for customers using SaaS and server usage-based products.

Dashboards are available to AWS Marketplace sellers who have the appropriate permissions.

Accessing dashboards

By default, AWS Marketplace system administrators for seller accounts have access to all dashboards on the Insights tab in the AWS Marketplace Management Portal. System administrators can create an AWS Identity and Access Management (IAM) policy to provide access for specific dashboards to other users in the seller company.

Note

In September 2023, we will no longer support access to seller dashboards enabled by legacy IAM permissions. Update your IAM permissions using the new Amazon Resource Name (ARN) format in the code examples below.

For information about creating policies, see [Creating IAM policies](#).

Dashboard policy

Use one of the following policy to provide access to the billed revenue dashboard and the collections and disbursements dashboard.

You can provide access to current and future AWS Marketplace resources (including dashboards and reports) based on current and future data feeds, using the following code example:

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

```

    "aws-marketplace:GetSellerDashboard"
  ],
  "Resource": [
    "arn:aws:aws-marketplace::<awsAccountID>:AWSMarketplace/*",
  ]
}]
}

```

Alternatively, you can provide access to one or more dashboards by including the specific ARN section, as shown in the following code example. For example, to provide access to only the billed revenue dashboard, agreements and renewals dashboard, and usage dashboard, remove this line from the following code example: `arn:aws:aws-marketplace::<awsAccountID>:AWSMarketplace/ReportingData/BillingEvent_V1/Dashboard/CollectionsAndDisbursements_V1`

```

{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "aws-marketplace:GetSellerDashboard"
    ],
    "Resource": [
      "arn:aws:aws-marketplace::<awsAccountID>:AWSMarketplace/ReportingData/BillingEvent_V1/Dashboard/BilledRevenue_V1",
      "arn:aws:aws-marketplace::<awsAccountID>:AWSMarketplace/ReportingData/BillingEvent_V1/Dashboard/CollectionsAndDisbursements_V1",
      "arn:aws:aws-marketplace::<awsAccountID>:AWSMarketplace/ReportingData/Agreement_V1/Dashboard/AgreementsAndRenewals_V1",
      "arn:aws:aws-marketplace::<awsAccountID>:AWSMarketplace/ReportingData/Usage_V1/Dashboard/Usage_V1",
      "arn:aws:aws-marketplace::123456789012:AWSMarketplace/ReportingData/TaxItem_V1/Dashboard/Tax_V1"
    ]
  }]
}

```

Note

For information about creating AWS Identity and Access Management (IAM) policies, see [Creating IAM policies](#) in the *AWS Identity and Access Management User Guide*.

Dashboards for finance operations

AWS Marketplace provides multiple dashboards to help you track your financial data.

Topics

- [Billed revenue dashboard](#)
- [Collections and disbursement dashboard](#)
- [Taxation dashboard](#)

Billed revenue dashboard

The billed revenue dashboard provides information about all billed sales in AWS Marketplace. This report is expected to save time and is available approximately 45 days earlier than the legacy [monthly billed revenue report](#). The legacy report delivers on the 15th day of each month, which delays visibility on billings of the prior month.

You can export and download data (as .csv or Microsoft Excel files) from any chart, graph, and table on the Amazon QuickSight dashboard. For more information, see [Exporting data from visuals](#) in the *Amazon QuickSight User Guide*.

For operational and financial processes, review the following topics.

Topics

- [Refresh frequency of the billed revenue dashboard](#)
- [Section 1: Controls](#)
- [Section 2: Invoice date range](#)
- [Section 3: Metrics](#)
- [Section 4: Trends](#)
- [Section 5: Breakdowns](#)
- [Section 6: Granular data](#)

Refresh frequency of the billed revenue dashboard

Dashboards are updated daily at 4 PM PST (midnight UTC). If an invoice is created on May 10 *before* 4 PM PST (midnight UTC), then the update on May 11 will display the invoice. If an invoice is created on May 10 *after* 4 PM PST (midnight UTC), then the update on May 12 will display the

invoice. If the latest invoicing or disbursement data received from upstream systems is delayed, there might be a 1–2 day delay for the latest data to reflect on the dashboards.

If you don't see an expected customer invoice, use the following procedure before contacting your AWS Marketplace business development contact.

To troubleshoot missing customer invoices for the billed revenue dashboard

1. Confirm that the offer was accepted by the customer using one or more of the following tools:
 - [Daily customer subscriber report](#)
 - [Notification for AWS Marketplace events](#)
 - [AWS Marketplace Management Portal agreements tab](#)
2. (For private offers) After you confirm that the customer accepted the offer, review the offer invoice schedule and amounts:
 - For private offers, check the AWS Marketplace Management Portal **Offers** tab.
 - For channel partner private offers, check the AWS Marketplace Management Portal **Partners** tab to view a custom payment schedule within the partner opportunity.
3. Consider if the transaction uses a [usage pricing model](#), where customers are billed on the second and third day of each month for the previous month of usage.

Section 1: Controls

This section of the dashboard provides filters to refine your billed revenue dashboard data. For example, you can select a filter on a field from the [notifications for AWS Marketplace events](#) to confirm billing for a specific customer account ID, subscriber company name, or offer ID. You can also add a filter to an analysis, such as the range of dates that you want to include in visuals. The filters selected within the controls update the data that is displayed in the metrics, trends, breakdowns, and granular data sections.

For more information about filtering, see [Filtering data on Amazon QuickSight](#) in the *Amazon QuickSight User Guide*.

Control descriptions

Control name	Description
Subscriber AWS account ID	The ID of the account that subscribed to the product.
Subscriber company name	The name of the account that subscribed to the product.
Product title	The title of the product.
Offer ID	The identifier for the offer that the buyer signed.
Offer visibility	Whether the offer is a public, private, or enterprise contract offer.
Agreement ID	A unique agreement data feed reference for the agreement signed between a proposer and an acceptor to start using a product.
AWS seller of record	<p>An identifier of the business entity which facilitated the transaction. Possible values include:</p> <ul style="list-style-type: none"> • AWS_INC: The identifier for AWS, Inc. (based in the United States). • AWS_EUROPE: The identifier for Amazon Web Services EMEA SARL (based in Luxembourg). • AWS_AUSTRALIA: The identifier for Amazon Web Services Australia Pty Ltd • AWS_JAPAN: The identifier for Amazon Web Services Japan G.K.
Payer AWS account ID	The ID of the account that the charges are billed to.

Control name	Description
Payer company name	The business name of the account that the charges are billed to.
Reseller company	The business name of the reseller account authorized to sell a software manufacturer's product.
Reseller AWS account ID	The ID of the account that purchased a product or service at wholesale from an ISV to resell to a customer.
Resale authorization ID	The unique identifier for a registered opportunity.
Resale authorization name	The unique name for a registered opportunity.
Subscriber country	The two-character country code associated with the account subscribed to the product.
Subscriber state or region	The billing address state or region associated with the account subscribed to the product.
Transaction reference ID	A unique identifier for the transaction that helps you correlate transactions across AWS Marketplace legacy reports.

Section 2: Invoice date range

This section of the dashboard provides filters to refine records based on whether the billing date is before or after a specified date or within a date range. The filter selected within the invoice date range updates the data displayed in the metrics, trends, breakdowns, and granular data sections. The default invoice date range is set to pull data from the last six months.

Section 3: Metrics

This section of the dashboard displays a key performance indicator (KPI) to visualize a comparison of key billed revenue figures. A KPI is displayed for gross revenue, gross refund, listing fee,

wholesale cost (if applicable), and seller net revenue for the specified invoice date range. You can update the date range by updating the date criteria in the invoice date range field.

Section 4: Trends

This section of the dashboard provides billed revenue trends for a specified date range. You can view the trends by a specified date aggregation—such as day, month-over-month, quarter-over-quarter, or year-over-year—to gain insight into billed revenue. Billed revenue trends information is available at an aggregate level or by offer visibility type:

- **Billing trends** – Provides a snapshot of gross revenue, seller net revenue, wholesale cost (if applicable), and refunds over time for the selected date range from the date aggregation filter.
- **Billing trend by offer visibility** – Provides a snapshot of offer count and gross revenue by offer visibility type over time across [private offers](#), public offers (or self-service), and enterprise programs.

Section 5: Breakdowns

This section of the dashboard provides you with key metrics about your business across subscribers, seller of records, subscriber geography, and product title. You can filter by gross revenue, payer count, subscriber count, gross refund, listing fee, seller net revenue, and wholesale cost.

Section 6: Granular data

This section of the dashboard shows all billed sales, including the total amount that AWS bills to customers for hourly, annual, or monthly usage of your products. AWS bills customers using the following three frequencies:

- Date of subscription acceptance (upfront billing)
- Custom payment schedule (private offers built by flexible payment scheduler)
- Metered usage on the second and third day of the month for the prior month's usage

Note

Invoices created before April 1, 2021 might not have an associated agreement ID, offer ID, subscriber AWS account ID, or subscriber company name.

Customer charges display in this granular data section 24 hours after the customer has been billed. For example, if a customer was charged on November 3, then the dashboard reports this invoice on November 4. For more information about how to export and download data from a QuickSight table, see [Exporting data from visuals](#) in the *Amazon QuickSight User Guide*.

Granular data descriptions

Column	Description
Invoice date	The date the customer was billed for the product subscription.
Payment due date	The payment due date in the format of YYYY-MM-DD.
Payment terms	The customer's AWSinvoice payment terms.
Invoice ID	The AWS ID assigned to the charges billed to the customer.
Listing fee invoice ID	When an AWS Marketplace subscription is transacted through AWS EMEA SARL, Japan or Australia legal entities (seller of record), the operator for the sale is required to charge the seller a value-added tax (VAT) on the seller listing fee and given a tax compliant invoice. For applicable transactions, the invoice ID for the VAT assessed on the listing fee is different than the software or product subscription invoice ID. Transactions from AWS, Inc. have a value of "Missing_listing_fee_invoice_id" because the listing fee invoice ID is not applicable.
Subscriber company name	The name of the account that subscribed to the product.
Subscriber AWS account ID	The ID of the account that subscribed to the product.

Column	Description
Subscriber email domain	The email domain associated with the account that subscribed to the product. For example, if the email address is liu-jie@example.com, the entry is example.com.
Subscriber city	The billing address city associated with the account that subscribed to the product.
Subscriber state or region	The billing address state associated with the account subscribed to the product.
Subscriber country	The billing address country associated with the account that subscribed to the product.
Subscriber postal code	The billing address postal code associated with the account that subscribed to the product.
Product title	The title of the product.
Offer name	The seller-defined name of the offer.
Offer ID	The identifier for the offer that the buyer signed.
Offer visibility	Whether the offer is a public, private, or enterprise contract offer.
Agreement ID	A unique agreement data feed reference for the agreement signed between a proposer and an accepter to start using a product.
Agreement start date	The date that the customer's product subscription starts, formatted as MM-DD-YYYY. This date could be different than acceptance date if this is a future dated agreement.

Column	Description
Agreement acceptance date	The date when the customer subscribed to the product, formatted as MM-DD-YYYY.
Agreement end date	The date when the contract expires, formatted as MM-DD-YYYY. For metered/pay-as-you-go subscriptions, this date is set to JAN-1-9999.
Usage period end date	The end date of the product usage period.
Usage period start date	The start date of the product usage period.
Disbursement status	A status associated with an invoice to confirm that AWS has collected and disbursed funds to your bank accounts since the previous disbursement. Disbursed funds for the associated invoice have been collected and disbursed. Not Disbursed funds for the associated invoice have not been collected and disbursed.
Disbursement date	The date AWS initiated disbursement to the seller's bank.
Disburse bank trace ID	For disbursements, the trace ID is assigned by the bank. The disburse bank trace ID can be used to correlate seller bank-provided deposit notifications and reports to invoices in AWS Marketplace reports.
Gross revenue	The amount that is billed to the customer for the usage or monthly fees of the product.
Gross refund	The total amount of the subscription cost refunded to customers if any refunds were processed during the data coverage period.

Column	Description
Listing fee	The AWS Marketplace fee amount to be deducted from the billed amount.
Listing fee refund	The portion of the AWS Marketplace fee refunded if any refunds were processed during the data coverage period.
Listing fee percentage	The AWS Marketplace fee percentage to be deducted from the billed amount.
Seller tax share	The total amount of US sales and use tax billed for this transaction.
Seller tax share refund	The total amount of US sales and use tax refunded for this transaction if a refund was processed.
AWS tax share	The total amount of US sales and use tax billed for this transaction on behalf of the seller.
AWS tax share refund	The total amount of US sales and use tax refunded for this transaction if a refund was processed, when such taxes were collected on behalf of the seller.
Wholesale cost	For channel partner private offers only. The cost of goods to a reseller. For example, what a reseller pays a manufacturer when they sell a manufacturer's product. The wholesale cost is the list price multiplied by the discount percentage.
Wholesale cost refund	For channel partner private offers only. The refunded cost of goods from a reseller.

Column	Description
Seller net revenue	The total amount billed for the transaction, net of AWS Marketplace fees, refunds, and US sales and use tax.
Currency	The currency of the transaction. For example, if the transaction is in US dollars, the entry is USD.
Transaction reference ID	A unique identifier that represents the transaction, which you can use to correlate transactions across AWS Marketplace reports.
AWS seller of record	<p>An identifier of the business entity which facilitated the transaction. Possible values are as follows:</p> <ul style="list-style-type: none">• AWS_INC: The identifier for AWS, Inc. (based in the United States)• AWS_EUROPE: The identifier for Amazon Web Services EMEA SARL (based in Luxembourg)• AWS_AUSTRALIA: The identifier for Amazon Web Services Australia Pty Ltd• AWS_JAPAN: The identifier for Amazon Web Services Japan G.K.
Resale authorization ID	The unique identifier for a registered opportunity.
Resale authorization name	The unique name for a registered opportunity.
Resale authorization description	The ISV-defined description for a registered opportunity.

Column	Description
Resale company name	The name of the account that purchased a product or service at wholesale cost from an ISV to resell to a customer.
Payer AWS account ID	The ID of the account that purchased a product or service at wholesale cost from an ISV to resell to a customer.
Payer email domain	The email domain that is associated with the account that the charges are billed to. For example, if the email address is liu-jie@example.com, the entry is example.com.
Payer city	The billing address city associated with the account that charges are billed to.
Payer state or region	The billing address state associated with the account that the charges are billed to.
Payer country	The two-character country code associated with the account that the charges are billed to.
Payer postal code	The billing address postal code associated with the account that the charges are billed to.
ISV account ID	The identifier of the product or service owner.
ISV company name	The business name of the product or service owner.
Product ID	The friendly unique identifier for the software product.

Collections and disbursement dashboard

The collections and disbursements dashboard provides data about funds that AWS collected and disbursed to your bank accounts since the previous disbursement. It also provides a list of all invoices that are open and unpaid.

Disbursements include customer payments or refunds for a subscription to your product and some taxes collected or refunded to the customer. You don't receive disbursement of funds until the full amount of funds on an invoice is collected from the customer. Refunds on the dashboard appear as negative amounts because the money is returned to your customer after you authorize a refund.

By using the collections and disbursements dashboard, you'll receive more timely access to customer disbursements. Expect to save approximately 4 days compared to the legacy [disbursement report](#), which is created 5 days after the disbursement is sent.

Note

Customers have different payment terms with AWS, so some funds in the uncollected age categories might not be due from the customer.

The collections and disbursements dashboard provides information for operational and financial processes. The dashboard refreshes daily. For more information, see the following topics.

Topics

- [Refresh frequency of the collections and disbursements dashboard](#)
- [Section 1: Controls](#)
- [Section 2: Select date category](#)
- [Section 3: Metrics](#)
- [Section 4: Trends](#)
- [Section 5: Breakdowns](#)
- [Section 6: Granular data](#)

Refresh frequency of the collections and disbursements dashboard

The collections and disbursements dashboard is updated on North American business days only. You can expect to see disbursed invoices within 1 day of receiving a deposit to your bank.

Section 1: Controls

This section of the dashboard provides filters to refine your dashboard data. For example, you can select a filter on a field from the [notifications for AWS Marketplace events](#) to confirm disbursement for a specific customer account ID, subscriber company name, or offer ID. You can also filter by disbursement status to understand all invoices paid to you or invoices open and unpaid. You can add a filter to an analysis, such as the range of dates that you want to include in visuals. The filters selected within the controls update the data displayed in the metrics, trends, breakdowns, and granular data sections.

For more information about filtering, see [Filtering data on Amazon QuickSight](#) in the *Amazon QuickSight User Guide*.

Control descriptions

Control name	Description
Subscriber AWS account ID	The ID of the account that subscribed to the product.
Subscriber company name	The name of the account that subscribed to the product.
Product title	The title of the product.
Offer ID	The identifier for the offer that the buyer signed.
Offer visibility	Whether the offer is a public, private, or enterprise contract offer.
Agreement ID	A unique agreement data feed reference for the agreement signed between a proposer and an accepter to start using a product.
AWS seller of record	An identifier of the business entity that facilitated the transaction. Possible values include:

Control name	Description
	<ul style="list-style-type: none"> • AWS_INC: The identifier for AWS, Inc. (based in the United States). • AWS_EUROPE: The identifier for Amazon Web Services EMEA SARL (based in Luxembourg). • AWS_AUSTRALIA: The identifier for Amazon Web Services Australia Pty Ltd (AWS Australia) • AWS_JAPAN: The identifier for Amazon Web Services Japan G.K.
Disbursement status	A status associated with an invoice to confirm that AWS has collected and disbursed funds to your bank accounts since the previous disbursement. Disbursed funds for the associated invoice have been collected and disbursed. Not Disbursed funds for the associated invoice have not been collected and disbursed.
Payer AWS account ID	The ID of the account that the charges are billed to.
Payer company name	The business name of the account that the charges are billed to.
Reseller company	The business name of the reseller account authorized to sell a software manufacturer's product.
Reseller AWS account ID	The ID of the account that purchased a product or service at wholesale from an ISV to resell to a customer.
Resale authorization ID	The unique identifier for a registered opportunity.

Control name	Description
Resale authorization name	The unique name for a registered opportunity.
Subscriber country	The two-character country code associated with the account subscribed to the product.
Subscriber state or region	The billing address state or region associated with the account subscribed to the product.
Transaction reference ID	A unique identifier for the transaction that helps you correlate transactions across AWS Marketplace legacy reports.
Disburse bank trace ID	For disbursements, the trace ID is assigned by the bank. The bank trace ID can correlate seller bank-provided deposit notifications and reports to invoices in AWS Marketplace reports.

Section 2: Select date category

This section of the dashboard provides filters to refine records based on two different date dimensions and whether the date field value is before or after a specified date or within a date range. The date dimensions are payment due date or last disbursement date. The date category filter updates the data displayed in the metrics, trends, breakdowns, and granular data sections. The default date category is the last disbursement date and pulls data from the last six months.

Section 3: Metrics

This section of the dashboard displays a key performance indicator (KPI) to visualize a comparison between disbursed and undisbursed revenue figures. A KPI is displayed for gross revenue, net revenue, wholesale cost (if applicable), amount disbursed, and amount undisbursed for a specified date category and date range.

Section 4: Trends

This section of the dashboard provides a view of disbursement and past due trends for the specified date range. You can view the trends by a specified date aggregation, such as by day,

month, quarter, or year, to gain insight into your AWS Marketplace collection health. Trend views include the following:

- **Disbursement trends** – Provides a snapshot of the average number of days to disburse and associated net revenue. The trend measures the number of days between invoice date and disbursement date to report collection efficiency. You can select a date range from the date aggregation filter.
- **Age of disbursed payments** – Provides a snapshot of net revenue and a count of disbursed invoices that is categorized by standard aging receivable buckets (such as not due, 1 to 30 days, and 31 to 60 days). The trend measures the days between payment due date and disbursement date to report if the disbursement was within the customer's payment terms.
- **Age of undisbursed payments** – Provides a snapshot of net revenue and count of open and unpaid invoices, organized by past due buckets (such as not due, 1 to 30 days, and 31 to 60 days). Undisbursed funds might include amounts that aren't due yet. The trend measures days between today's date and the payment due date to display incoming receivables.

Section 5: Breakdowns

This section of the dashboard provides you with a view of receivables by offer ID, product title, payer company name, subscriber company name, reseller name (if they participate in channel partner private offers), payer geography, and subscriber geography. Use the breakdowns to measure disbursed receivables against undisbursed receivables for each category.

Section 6: Granular data

This section of the dashboard shows all disbursements and uncollected funds by product, customer, and offer details.

Note

Invoices created before April 1, 2021 might not have an associated agreement ID, offer ID, subscriber AWS account ID, or subscriber company name.

For information about how to export and download data from a QuickSight table, see [Exporting data from visuals](#) in the *Amazon QuickSight User Guide*.

Granular data descriptions

Column	Description
Invoice date	The date the customer was billed for the product subscription.
Payment due date	The payment due date in the format of YYYY-MM-DD.
Payment terms	The customer's AWS invoice payment terms.
Invoice ID	The AWS ID assigned to the charges billed to the customer.
Listing fee invoice ID	When an AWS Marketplace subscription is transacted through AWS EMEA SARL, Japan, or Australia legal entities (seller of record), the marketplace operator for the sale (for example, AWS EMEA SARL) is required to charge the seller a VAT on the seller listing fee. For applicable transactions, the invoice ID for the VAT assessed on the listing fee is different than the software or product subscription invoice ID.
Subscriber company name	The name of the account that subscribed to the product.
Subscriber AWS account ID	The ID of the account that subscribed to the product.
Subscriber email domain	The email domain associated with the account that subscribed to the product. For example, if the email address is liu-jie@example.com, the entry is example.com.
Subscriber city	The billing address city associated with the account that subscribed to the product.

Column	Description
Subscriber state or region	The billing address state associated with the account subscribed to the product.
Subscriber country	The billing address country associated with the account that subscribed to the product.
Subscriber postal code	The billing address postal code associated with the account that subscribed to the product.
Product title	The title of the product.
Offer name	The seller-defined name of the offer.
Offer ID	The identifier for the offer that the buyer signed.
Offer visibility	Whether the offer is a public, private, or enterprise contract offer.
Agreement ID	A unique agreement data feed reference for the agreement signed between a proposer and an accepter to start using a product.
Agreement start date	The date the customer's product subscription starts, formatted as MM-DD-YYYY. This date could be different than acceptance date if this is a future dated agreement.
Agreement end date	The date when the contract expires, formatted as MM-DD-YYYY. For metered/pay-as-you-go subscriptions, this date is set to JAN-1-9999.
Agreement acceptance date	The date when the customer subscribed to the product, formatted as MM-DD-YYYY.

Column	Description
Usage period end date	The end date of the product usage period.
Usage period start date	The start date of the product usage period.
Disbursement status	A status associated with an invoice to confirm that AWS has collected and disbursed funds to your bank accounts since the previous disbursement. Disbursed funds for the associated invoice have been collected and disbursed. Not Disbursed funds for the associated invoice have not been collected and disbursed.
Disbursement date	The date AWS initiated disbursement to the seller's bank.
Disburse bank trace ID	For disbursements, the trace ID is assigned by the bank. The bank trace ID can be used to correlate seller bank-provided deposit notifications and reports to invoices in AWS Marketplace reports.
Gross revenue	The amount billed to the customer for the usage or monthly fees of the product.
Gross refund	The total amount of the subscription cost refunded to customers if any refunds were processed during the data coverage period.
Listing fee	The AWS Marketplace fee amount to be deducted from the billed amount.
Listing fee refund	The portion of the AWS Marketplace fee refunded if any refunds were processed during the data coverage period.

Column	Description
Listing fee percentage	The AWS Marketplace fee percentage to be deducted from the billed amount.
Seller tax share	The total amount of US sales and use tax billed for this transaction.
Seller tax share refund	The total amount of US sales and use tax refunded for this transaction if a refund was processed.
Wholesale cost	For channel partner private offers only. The cost of goods to a reseller. For example, what a reseller pays a manufacturer when they sell a manufacturer's product. The wholesale cost is the list price multiplied by the discount percentage.
Wholesale cost refund	For channel partner private offers only. The refunded cost of goods from a reseller.
Seller net revenue	The total amount billed for the transaction, net of AWS Marketplace fees, refunds, and US sales and use tax.
Currency	The currency of the transaction. For example, if the transaction is in US dollars, the entry is USD.
Transaction reference ID	A unique identifier that represents the transaction, which you can use to correlate transactions across AWS Marketplace reports.

Column	Description
AWS seller of record	<p>An identifier of the business entity which facilitated the transaction. Possible values are as follows:</p> <ul style="list-style-type: none">• AWS_INC: The identifier for AWS, Inc. (based in the United States)• AWS_EUROPE: The identifier for Amazon Web Services EMEA SARL (based in Luxembourg)• Amazon Web Services Australia Pty Ltd (AWS Australia)• Amazon Web Services Japan G.K.
Resale authorization ID	The unique identifier for a registered opportunity.
Resale authorization name	The unique name for a registered opportunity.
Resale authorization description	The ISV-defined description for a registered opportunity.
Reseller company name	The name of the account that purchased a product or service at wholesale cost from an ISV to resell to a customer.
Reseller AWS account ID	The ID of the account that purchased a product or service at wholesale cost from an ISV to resell to a customer.
Payer company name	The name of the account that the charges are billed to.
Payer AWS account ID	The ID of the account that the charges are billed to.

Column	Description
Payer email domain	The email domain that is associated with the account that the charges are billed to. For example, if the email address is liu-jie@example.com, the entry is example.com.
Payer city	The billing address city associated with the account that charges are billed to.
Payer state or region	The billing address state associated with the account that the charges are billed to.
Payer country	The two-character country code associated with the account that the charges are billed to.
Payer postal code	The billing address postal code associated with the account that the charges are billed to.
ISV account ID	The identifier of the product or service owner.
ISV company name	The business name of the product or service owner.
Product ID	The friendly unique identifier for the software product.
Disbursed net revenue	The total amount for the transaction disbursed to seller.
Undisbursed net revenue	The total amount for the transaction not disbursed to seller
Disbursement period	The categories describing the receivables range in which the funds were collected (such as, not due, 1 to 30 days, and 31 to 60 days).

Taxation dashboard

The taxation dashboard provides visualization and fine-grained data on US and international tax for transactions in AWS Marketplace. This dashboard can be accessed from the AWS Marketplace Management Portal **Insights** tab under **Finance operations**.

You can export and download data (as .csv or Microsoft Excel files) from any chart, graph, and table on the Amazon QuickSight dashboard. For more information, see [Exporting data from visuals](#) in the *Amazon QuickSight User Guide*.

For operational and financial processes, review the following topics.

Topics

- [Refresh frequency of the taxation dashboard](#)
- [Section 1: Controls](#)
- [Section 2: Filters date](#)
- [Section 3: Metrics](#)
- [Section 4: Tax trends](#)
- [Section 5: Breakdowns](#)
- [Section 6: Granular data](#)
- [US exemption codes](#)
- [EU exemption codes](#)

Refresh frequency of the taxation dashboard

Dashboards are updated daily. If the latest transaction data received from upstream systems is delayed, there might be a 1–2 day delay for the latest data to reflect on the dashboards.

Section 1: Controls

This section of the dashboard provides filters to refine your taxation data. You can also add a filter to an analysis, such as the range of dates that you want to include in visuals. The filters selected within the controls update the data that is displayed in the metrics, trends, breakdowns, and granular data sections.

For more information about filtering, see [Filtering data on Amazon QuickSight](#) in the *Amazon QuickSight User Guide*.

Control descriptions

Control name	Description
Invoice ID	The AWS ID assigned to the charges billed to the customer.
Taxed customer AWS account ID	The AWS ID of the account that was taxed for the product subscription.
AWS seller of record	An identifier of the business entity that facilitated the transaction.
Taxed customer country	The two-character country code associated with the taxed customer
Taxed customer state or region	The billing address state or region associated with the taxed customer.
Taxed customer city	The billing address city associated with the taxed customer
Taxable	Source transactions that are taxable, non-taxable, and tax refunds.
Tax liable party	<p>Either AWS or Seller. If the seller is the tax liable party, taxes are collected. If AWS is the tax liable party, sales tax is collected and remitted by AWS. For more information, see AWS Marketplace Tax Help for Sellers.</p> <p>If no taxes are collected, there's no value shown. The seller must determine whether some taxes were collected for each invoice, as the seller is liable for tax collection.</p>
Tax type	The type of tax that is applied to the transaction. The possible values are None, Sales, and

Control name	Description
	SellerUse , Tax registration type , VAT (Value-Added Tax), CNPJ , IGST, CT.

Section 2: Filters date

This section of the dashboard provides filters to refine records based on invoice dates for a specified date range. The filter selected within invoice date updates the data displayed in the metrics, trends, breakdowns, and granular data sections. For example, if you select the date filter Past 30 days, then all sections will reflect data associated to invoices taxed within the last 30 days.

Section 3: Metrics

This section of the dashboard displays key performance indicators (KPI) to visualize metrics related to tax, including taxable amount, non-taxable amount, tax amount, number of invoices taxed, and total number of transactions. You can update the date range by updating the date filter in the **Filters** section.

Section 4: Tax trends

This section of the dashboard provides taxation trends for a specified date range. You can view the trends by a specified date aggregation—such as daily, month-over-month, quarter-over-quarter, or year-over-year—to gain insight into taxation. The following taxation trends information is available:

- **Tax trends - amounts** – Provides a snapshot of both the taxable and tax amounts for the selected period of time by invoice date.
- **Tax trends - number of invoices** – Provides a snapshot of the number of invoices for the selected period of time by invoice date.

Section 5: Breakdowns

This section of the dashboard provides you with tax metrics for your business by invoice ID, tax type, product title, and customer geography.

Section 6: Granular data

This section of the dashboard displays the granular tax data for your organization's transactions in AWS Marketplace.

Granular data descriptions

Column	Description
Invoice ID	The AWS ID assigned to the charges billed to the customer.
Line item ID	A unique identifier for a line item. Refund transactions have the same line item ID as their forward tax transactions.
Customer bill ID	Customer bill ID
Tax liable party	<p>Either AWS or Seller. If the seller is the tax liable party, taxes are collected. If AWS is the tax liable party, sales tax is collected and remitted by AWS. For more information, see AWS Marketplace Tax Help for Sellers.</p> <p>If no taxes are collected, there's no value shown. The seller must determine whether some taxes were collected for each invoice, as the seller is liable for tax collection.</p>
Transaction type code	<p>The type code of the transaction. Values include:</p> <ul style="list-style-type: none"> • AWS – A forward tax transaction. • REFUND – A full or partial refund. • TAXONLYREFUND – A tax-only refund. <p>Refund transactions share the line item ID with their original forward transactions.</p>

Column	Description
Product ID	The friendly unique identifier for the software product.
Product title	The name of the product purchased.
Product tax code	A standard code to identify the tax properties for a product. You choose the properties when you create or modify the product.
Invoice date	The date the customer was billed and taxed for the product subscription.
Taxed customer AWS account ID	The AWS account ID of the taxed customer.
Taxed customer country	The two-character country code associated with the taxed customer.
Taxed customer state or region	The billing address state or region associated with the taxed customer.
Taxed customer city	The billing address city associated with the taxed customer.
Taxed customer postal code	The postal code used for tax calculations.
Tax type	The type of tax that is applied to the transaction. The possible values are None, Sales, and SellerUse , Tax registration type , VAT, CNPJ, IGST, CT.
Jurisdiction level	The jurisdiction level of the address that is used for tax location. The possible values are State, County, City, and District.
Taxed jurisdiction	The name of the taxed jurisdiction.
Invoice charge	Summary of the taxable amount and the non-taxable amount.

Column	Description
Display price taxability type	The taxability type for the price that appears to customers. All AWS Marketplace offerings are exclusive.
Taxable amount	The amount of the transaction that is taxable, at this jurisdiction level.
Non-taxable amount	The amount of the transaction that is nontaxable, by jurisdiction level.
Tax jurisdiction rate	The tax rate that is applied at the jurisdiction level.
Tax amount	The tax that is charged at the jurisdiction level.
Tax currency	The currency of the charged taxed. For example, if the transaction is in US dollars, the entry is USD.
Tax calculation reason code	Whether the transaction is taxable, not taxable, exempt, or zero-rated, organized by the jurisdiction level.
Date used for tax calculation	The date that is used for calculating tax on the transaction.
Customer exemption certificate ID	The certificate ID of the exemption certificate.
Customer exemption certificate ID domain	Where the certificate is being stored in Amazon systems.
Customer exemption certificate level	The jurisdiction level that supplied the exemption.
Customer exemption code	The code that specifies the exemption. For example, RESALE.

Column	Description
Transaction reference ID	A unique identifier for the transaction that helps you correlate transactions across AWS Marketplace reports.
AWS seller of record	An identifier of the business entity that facilitated the transaction.

US exemption codes

Exemption code	Description
501C	501C Organization
AP	Agriculture Production
CO	Charitable Organization
DIRECT	Direct Pay Permit
DISTSBT	Distributor
DP	United Nations/Diplomat
DV	Disabled Veteran
EDI	Education Institution
FG	Federal Government
HCP	Health Care Provider
HO	Hospital (Nonprofit or State)
INSUR	Insurance
IPM	IPM
LB	Library

Exemption code	Description
MPU	Multiple Point of Use Exemption; Digital Products/Software Exemption
NA	Native American
NAI	Native American Individual
NP	Nonprofit Organization
NPR	Nonprofit Religious Organization
OT	Other
RESALE	Reseller
SBE	Small Business Exemption
SLG	State/Local Government

EU exemption codes

Exemption code	Description
SpecialOrg	Organizations that are exempt from being charged VAT.
SpecialZone	An area that's tax exempt.
SplitPayment	Buyer makes payment of any VAT stated on an invoice directly to the tax authority.

Dashboards for sales operations

AWS Marketplace provides multiple dashboards to help you track your sales data.

Topics

- [Agreements and renewals dashboard](#)

- [Usage dashboard](#)

Agreements and renewals dashboard

The agreements and renewals dashboard provides information about agreements and renewals within 24 hours of signing an agreement in AWS Marketplace. It helps to track expiring subscriptions to enable renewals. This dashboard can be accessed in the AWS Marketplace Management Portal **Insights** tab under **Sales operations**.

You can export and download data (as .csv or Microsoft Excel files) from any chart, graph, and table on the Amazon QuickSight dashboard. For more information, see [Exporting data from visuals](#) in the *Amazon QuickSight User Guide*.

Topics

- [Section 1: Controls](#)
- [Section 2: Select date category](#)
- [Section 3: Metrics](#)
- [Section 4: Subscription trends](#)
- [Section 5: Breakdowns](#)
- [Section 6: Granular data](#)

Section 1: Controls

This section of the dashboard provides filters to refine your agreement and renewal dashboard data. You can select from the following filters.

Note

For more information about filtering, see [Filtering data on Amazon QuickSight](#) in the *Amazon QuickSight User Guide*.

Control descriptions

Control name	Description
Subscriber AWS account ID	The ID of the account that subscribed to the product.
Subscriber company name	The name of the account that subscribed to the product.
Offer ID	The identifier for the offer that the buyer signed.
Offer visibility	Whether the offer is a public, private, or an enterprise contract offer.
Agreement ID	A unique agreement data feed reference for the agreement signed between a proposer and an acceptor to start using a product.
CPPO flag	A yes/no flag indicating whether an agreement was made using a channel partner private offer. If yes, the seller of record is the channel partner. If no, the seller of record is the software manufacturer (independent software vendor).
ISV AWS account ID	The account of the product or service owner.
ISV company name	The business name of the product or service owner.
Legacy product ID	The legacy unique identifier for the software product.
Product title	The title of the product.
Resale authorization ID	The ID of the resale authorization provided by the ISV to the channel partner.

Control name	Description
Resale authorization name	The name of the resale authorization provided by the ISV to the channel partner.
Reseller AWS account ID	The ID of the account that purchased a product or service at wholesale from an ISV to resell to a customer.
Reseller company name	The business name of the reseller account authorized to sell a software manufacturer's product.
Subscriber country	The two-character country code associated with the account subscribed to the product.
Subscriber state or region	The billing address state or region associated with the account subscribed to the product.
Subscriber city	The billing address city associated with the account subscribed to the product.

Section 2: Select date category

This section of the dashboard provides filters to refine records based on the agreement end date, agreement ending period, or agreement start date for a specified date range. The filter selected within the select date category updates the data displayed in the metrics, trends, breakdowns, and granular data sections.

For information about recently signed agreements, you can filter by agreement start date. If you're interested in learning about agreement renewals, you can filter using the agreement end date.

Section 3: Metrics

This section of the dashboard displays a key performance indicator (KPI) to visualize a comparison of key agreement data. A KPI is displayed for the number of active agreements and number of ended agreements. You can update the date range by updating the date criteria in the Select date category field.

Section 4: Subscription trends

This section of the dashboard provides subscription trends for a specified date range. You can view the trends by a specified date aggregation—such as daily, month-over-month, quarter-over-quarter, or year-over-year—to gain insight into agreements. The agreement and renewal trends information is available at an aggregate level.

- **Subscription trends** – Provides a snapshot of the number of active agreements and number of ended agreements based on the selected date aggregation filter and selected date category filter. To view subscription trends for agreement start dates, you can select **Agreement start date** in the **Select date category** filter. To view subscription trends for agreement end dates, select **Agreement end date** in the **Select date category** filter.
- **Renewal breakdowns** – Provides a snapshot of renewals that have either ended or are about to end in a specific number of days. You can view renewal breakdowns for several time periods, such as agreements that ended one to 30 days ago, 31 to 60 days ago, or more than 60 days ago. You can also filter on agreements that are about to end in zero to 30 days, 31 to 60 days, and so on.

Section 5: Breakdowns

This section of the dashboard provides you with metrics for your business across company names for active subscribers, ISVs, and resellers. You can view the number of agreements IDs, ended agreement IDs, offer IDs, renewed agreement IDs, and subscribers.

Section 6: Granular data

This section of the dashboard shows granular data for agreements, offers, products, proposers of the agreement, subscribers, resale authorizations, resellers, and ISVs.

Note

For more information about how to export and download data from a QuickSight table, see [Exporting data from visuals](#) in the *Amazon QuickSight User Guide*.

Granular data descriptions

Column	Description
Subscriber company name	The name of the subscriber's company.
Subscriber AWS account ID	The ID of the account that subscribed to the product.
Subscriber email domain	The email domain associated with the account that subscribed to the product. For example, if the email address is liu-jie@example.com, the entry is example.com.
Subscriber country	The billing address country associated with the account that subscribed to the product.
Subscriber state or region	The billing address state associated with the account subscribed to the product.
Subscriber city	The billing address city associated with the account that subscribed to the product.
Subscriber postal code	The billing address postal code associated with the account that subscribed to the product.
Product title	The title of the product.
Product ID	The friendly unique identifier for the software product.
Legacy product ID	The legacy unique identifier for the software product.
Offer name	The seller-defined name of the offer.
Offer ID	The identifier for the offer that the buyer signed.

Column	Description
Offer visibility	Whether the offer is a public, private, or an enterprise contract offer.
Agreement ID	A unique agreement data feed reference for the agreement signed between a proposer and an accepter to start using a product.
Agreement start date	The date that the customer's product subscription starts, formatted as MM-DD-YYYY Y. This date could be different than acceptance date if this is a future dated agreement.
Agreement acceptance date	The date when the customer subscribed to the product, formatted as MM-DD-YYYY.
Agreement end date	The date when the contract expires, formatted as MM-DD-YYYY. For metered/pay-as-you-go subscriptions, this date is set to JAN-1-9999.
Reseller company name	The name of the account that purchased a product or service at wholesale cost from an ISV to resell to a customer.
Reseller AWS account ID	The ID of the account that purchased a product or service at wholesale cost from an ISV to resell to a customer.
Resale authorization ID	The unique identifier for a registered opportunity.
Resale authorization name	The unique name for a registered opportunity.
Resale authorization description	The ISV-defined description for a registered opportunity.
CPPO flag	Calculated field in Amazon QuickSight.

Column	Description
Agreement intent	<p>Populates if the current agreement was a renewal or upgrade of a previous agreement that's provided in the previous agreement_id column. Possible values include:</p> <ul style="list-style-type: none"> • Upgrade: For private offers, upgrade is a broad term for any modification (upgrade, downgrade, amend, expand, renew). For public offers, an upgrade is a specific agreement to change terms (use a different offer) resulting in a new agreement, and the specific agreement is archived. • Renew: For public offers only. When a renewal agreement has start date as the specific agreement's end date and the offer ID hasn't changed but the agreement ID changed. For example, agreement A has become agreement B. • Auto-renew: For public offers only. When a new agreement is created when current agreement ends. The buyer has auto-renew turned on. • New: The buyer has accepted the terms (using the offer) and subscribed to a new agreement.
Previous agreement ID	Agreement reference if this agreement was renewed or upgraded.
Next agreement ID	Agreement ID of the next agreement if the current agreement was renewed or upgraded.
Previous offer ID	Offer ID associated with the previous agreements if current agreement was renewed or upgraded.

Column	Description
Next offer ID	Offer ID associated with the next agreement if current agreement was renewed or upgraded.
Next agreement ID acceptance date(s)	The date when the next agreement was accepted by the buyer.
Proposer AWS account ID	The identifier of the product or service owner.
Proposer company name	The business name of the product or service owner.
ISV AWS account ID	The AWS identifier of the seller.
ISV company name	The business name of the seller.
Agreement ends (in days)	Calculated field in Quicksight. The number of days left for agreement to end from today.
Agreement ending period	Calculated field in QuickSight. The time period within which the agreement will end.

Usage dashboard

The Usage dashboard provides visualizations and fine-grained data for customers using SaaS and server usage-based products. AWS Marketplace sellers can use this dashboard to track customer consumption across usage-based products to make decisions on product support, pricing, conversion from public to private offers, and product discontinuation. This dashboard can be accessed in AWS Marketplace Management Portal on the **Insights** tab under **Sales operations**. It provides data from the last 6 months, which is a rolling window.

You can export and download data (as .csv or Microsoft Excel files) from any chart, graph, and table on the Amazon QuickSight dashboard. For more information, see [Exporting data from visuals](#) in the *Amazon QuickSight User Guide*.

Note

This dashboard displays usage for all the dimension keys specified at the time of the offer creation. For example, for Amazon Machine Image (AMI) products, dimension keys are instance types, and all instance types specified in the offer will be shown in this dashboard, even if they're priced at \$0. To see product usage for a dimension that wasn't specified at the time of offer creation, consider republishing the product to include the dimension you need.

Topics

- [Refresh frequency of the usage dashboard](#)
- [Section 1: Controls](#)
- [Section 2: Filters](#)
- [Section 3: Metrics](#)
- [Section 4: Trends](#)
- [Section 5: Breakdowns](#)
- [Section 6: Granular data](#)

Refresh frequency of the usage dashboard

Dashboards are updated daily at 4 PM PST (midnight UTC). Note that the usage data is received from upstream data sources and may encounter delays, you can refer to the usage date and the usage reported date for clarity on when the usage occurred compared to when it was reported on the dashboard.

You can export and download data (as .csv or Microsoft Excel files) from any chart, graph, and table on the Amazon QuickSight dashboard. For more information, see [Exporting data from visuals](#) in the *Amazon QuickSight User Guide*.

For operational and financial processes, review the following topics.

Section 1: Controls

This section of the dashboard provides filters to refine your usage data. For example, you can select from the following filters.

Note

For more information about filtering, see [Filtering data in Amazon QuickSight](#) in the *Amazon QuickSight User Guide*.

Control descriptions

Control name	Description
End user company name	The name of the account that used the product.
End user AWS account ID	The ID of the account that used the product.
End user country	The two-character country code associated with the account that used the product.
Product title	The title of the product.
Product code	The existing entitlement product code used to meter the product. This value is also used to join data with a report, or to reference what's provided in the AWS Marketplace Metering Service.
Offer ID	The identifier for the offer that the buyer signed.
Offer visibility	Whether the offer is a public, private, or enterprise contract offer.
Agreement ID	A unique agreement data feed reference for the agreement signed between a proposer and an accepter to start using a product.
Dimension key	The resource type associated with the product usage. Dimension keys apply to SaaS and server usage-based products.

Control name	Description
Subscriber company name	The name of the account that subscribed to the product.
Subscriber AWS account ID	The ID of the account that subscribed to the product.
Subscriber country	The two-character country code associated with the account subscribed to the product.
Reseller company name	The name of the reseller account authorized to sell a product manufacturer's product.
Reseller AWS account ID	The ID of the account that purchased a product or service at wholesale from an ISV to resell to a customer.
Resale authorization ID	The ID of the account that purchased a product or service at wholesale from an ISV to resell to a customer.
CPPO flag	A yes/no flag indicating whether an agreement was made using a channel partner private offer. If yes, the seller of record is the channel partner. If no, the seller of record is the product manufacturer (independent software vendor).

Section 2: Filters

This section of the dashboard provides filters to refine records based on the usage date. The values selected in these filters update the data displayed in the metrics, trends, breakdowns, and granular data sections. The default selection is to pull data for last 6 months usage.

Section 3: Metrics

This section of the dashboard displays a key performance indicator (KPI) to visualize metrics related to consumption: estimated usage units, customers with usage, and products with usage. You can

update the date range by updating the usage date criteria in the filters section. Note that the key metrics display data for all unit types.

Section 4: Trends

This section of the dashboard provides usage trends for a specified date range. You can view the trends by a specified date aggregation, such as daily, month-over-month, quarter-over-quarter, or year-over-year to gain insight into usage. You can also select a usage unit type to view its usage trends graphically.

Section 5: Breakdowns

This section of the dashboard provides you with estimated usage metrics for your business across company names, product titles, dimension key and offer IDs for the unit type selected. You may also select the number of entries to view.

Section 6: Granular data

This section of the dashboard shows granular data for usage, offers, product, subscriber, payer, end user, resale authorizations, resellers, and independent software vendors (ISVs). Note that the granular data table displays data for all unit types.

Revenue should be considered estimated until billing is finalized at the end of the month. Usage-based invoices are presented to buyers on the second or third day of the following month for the previous month's usage (for example, customers metered for usage between 11/1 and 11/30 will be presented an invoice for the usage on 12/2 or 12/3). Metered usage may arrive to this dashboard several days after the actual usage date, so the usage date and usage reported dates may be different. This means you may need to visit the dashboard daily throughout the month for up-to-date tracking in the current month. For authoritative customer billing information, see the [Billed revenue dashboard](#) in the **Financial operations** tab.

Note

For more information about filtering, see [Exporting data from visuals](#) in the *Amazon QuickSight User Guide*.

Granular data descriptions

Column	Description
Usage date	The date of the customer's product consumption.
Usage reported date	The date the customer's product consumption is surfaced in the insights dashboard.
End user company name	The name of the account that used the product.
End user AWS account ID	The ID of the account that used the product.
End user email domain	The email domain associated with the account that used the product. For example, if the email address is abc@example.com, the entry is example.com.
End user city	The city associated with the account that used the product.
End user state or region	The state or region associated with the account that used the product.
End user country	The two-character country code associated with the account that used the product.
End user postal code	The billing address postal code associated with the account that used the product.
Product title	The title of the product.
Legacy product ID	The legacy unique identifier for the product.
Product ID	The friendly unique identifier for the product.
Product code	The existing entitlement product code used to meter the product. This value is also used

Column	Description
	to join data with a report, or to reference what's provided in AWS Marketplace Metering Service.
Offer ID	The identifier for the offer that the buyer signed.
Offer name	The seller-defined name of the offer.
Offer visibility	Whether the offer is a public, private, or an enterprise contract offer.
Agreement ID	A unique agreement data feed reference for the agreement signed between a proposer and an acceptor to start using a product.
Agreement acceptance date	The date time stamp in UTC when the customer subscribed to the product.
Agreement start date	The date timestamps in UTC when the customer's product subscription starts. This date could be different than acceptance date if this is a future dated agreement.
Agreement end date	The date in UTC when the contract expires. For metered/pay-as-you-go subscriptions, this date is set to Jan 1, 9999 12:00 AM.
Dimension key	The resource type associated with the product usage. Dimension keys apply for SaaS and server usage-based products.
Region	The region where the buyer deployed Amazon EC2 instances.
Estimated usage	The quantity of the usage recorded for the product.

Column	Description
Usage unit types	The unit type for which the usage is recorded.
Estimated revenue	The revenue from the product usage. Revenue should be considered estimated until billing is finalized at the end of the month. Usage-based invoices are presented to buyers on the second or third of the following month for the previous month's usage.
Currency	The currency of the transaction. For example, if the transaction is in U.S. dollars, the entry is USD.
Subscriber company name	The name of the account that subscribed to the product.
Subscriber AWS account ID	The ID of the account that subscribed to the product.
Subscriber email domain	The email domain associated with the account that subscribed to the product. For example, if the email address is abc@example.com, the entry is example.com.
Subscriber city	The billing address city associated with the account that subscribed to the product.
Subscriber state or region	The billing address state associated with the account subscribed to the product.
Subscriber country	The billing address country associated with the account that subscribed to the product.
Subscriber postal code	The billing address postal code associated with the account that subscribed to the product.

Column	Description
Payer company name	The name of the account that paid for the product.
Payer AWS account ID	The ID of the account that paid for the product.
Payer email domain	The email domain associated with the account that paid for the product. For example, if the email address is abc@example.com, the entry is example.com.
Payer city	The billing address city associated with the account that paid for the product.
Payer state or region	The billing address state associated with the account that paid for the product.
Payer country	The billing address country associated with the account that paid for the product.
Payer postal code	The billing address postal code associated with the account that paid for the product.
Reseller company name	The name of the account that purchased a product or service at wholesale cost from an ISV to resell to a customer.
Reseller AWS account ID	The ID of the account that purchased a product or service at wholesale cost from an ISV to resell to a customer.
Resale authorization ID	The unique identifier for a registered resale opportunity.
Resale authorization name	The unique name for a registered resale opportunity.

Column	Description
Resale authorization description	The description for a registered resale opportunity.
CPPO flag	A yes/no flag indicating whether an agreement was made using a channel partner private offer. If yes, the seller of record is the channel partner. If no, the seller of record is the product manufacturer (ISV).
ISV company name	The name of the product or service owner.
ISV AWS account ID	The identifier of the product or service owner.

AWS Marketplace Vendor Insights

AWS Marketplace Vendor Insights is a feature that simplifies software risk assessments performed by organizations to safeguard procuring software they trust and meets their standards. With AWS Marketplace Vendor Insights, buyers can monitor the security profile of a product in near real-time from a single console. AWS Marketplace Vendor Insights can ease the procurement process for buyers and potentially increase sales for sellers. It reduces a buyer's assessment effort by providing a dashboard of the software product's security and compliance information.

All security and compliance information in the AWS Marketplace Vendor Insights dashboard is based on evidence gathered from the following sources:

- Seller's self-attestation, including the AWS Marketplace Vendor Insights security self-assessment and Consensus Assessment Initiative Questionnaire (CAIQ)
- Industry standard audit reports (for example, International Organization for Standardization ISO 27001)
- AWS Audit Manager, which automates evidence collection from the seller's production environment

AWS Marketplace Vendor Insights gathers compliance artifacts and security control information about the product and presents it in a dashboard. The dashboard takes data from the seller's self-assessment, evidence from audit reports, and live evidence from AWS accounts. This data feeds into the security controls and then to the dashboard for buyers to review. Live evidence is the method of consistently updating data from multiple sources to present the most current information. AWS Config is enabled in the seller's environment. Data about configurations, backups enabled, and other information is updated automatically. For example, assume that the **Access Control** for a product is **Compliant** and an Amazon Simple Storage Service (Amazon S3) bucket becomes public. The dashboard would display that the control's status changed from **Compliant** to **Undetermined**.

You must set up the baseline resources and infrastructure in your AWS accounts before using AWS Marketplace Vendor Insights. After setup is completed, AWS Marketplace Vendor Insights can gather information and generate security profiles for your software as a service (SaaS) products in AWS Marketplace.

Contents

- [Understanding AWS Marketplace Vendor Insights](#)
- [Setting up AWS Marketplace Vendor Insights](#)
- [Viewing your AWS Marketplace Vendor Insights profile](#)
- [Managing snapshots in AWS Marketplace Vendor Insights](#)
- [Controlling access in AWS Marketplace Vendor Insights](#)

Understanding AWS Marketplace Vendor Insights

AWS Marketplace Vendor Insights gathers compliance artifacts and security control information for your product and presents it in a dashboard. The dashboard takes data from the product owner's self-assessment, evidence from audit reports, and live evidence from AWS accounts. This data feeds into the security controls and then to the dashboard for buyers to review.

The dashboard presents the evidence-based information gathered by AWS Marketplace Vendor Insights from multiple security control categories. This provides insight with a near real-time view of the security profile and reduces discussions between the buyer and seller. Buyers can validate a seller's information completing assessments within a few hours. AWS Marketplace Vendor Insights provides a mechanism for sellers to keep security and compliance posture information up-to-date automatically. They can share it with buyers on-demand which eliminates the need to respond to questionnaires on a random basis.

AWS Marketplace Vendor Insights gathers the evidence-based information from three sources:

- **Your vendor self-assessment** – Supported self-assessments include the AWS Marketplace Vendor Insights security self-assessment and Consensus Assessment Initiative Questionnaire (CAIQ).
- **Your production accounts** – Of the multiple controls, 25 controls support live evidence gathering from your production accounts. Live evidence for each control is generated by evaluating the configuration settings of your AWS resources using one or more AWS Config rules. AWS Audit Manager captures the evidence and prepares it for AWS Marketplace Vendor Insights to consume. The onboarding AWS CloudFormation template automates the prerequisite steps required for enabling live evidence gathering. AWS Config is enabled in the seller's environment. Data about configurations, backups enabled, and other information is updated automatically. For example, assume that the **Access Control** for a product is **Compliant** and an Amazon S3 bucket becomes public. The dashboard would display that the control's status changed from **Compliant** to **Undetermined**.

- Turning on AWS Config and the AWS Audit Manager service.
- Creating AWS Config rules and the AWS Audit Manager automated assessment.
- Provisioning the AWS Identity and Access Management (IAM) role so that AWS Marketplace Vendor Insights can pull assessment results.
- **Your ISO 27001 and SOC2 Type II report** – The control categories are mapped to controls in the International Organization for Standardization (ISO) and System and Organization Controls (SOC2) reports. When you share these reports with AWS Marketplace Vendor Insights, it can extract relevant evidence from these reports and present it on the dashboard.

Setting up AWS Marketplace Vendor Insights

The following procedure describes the high-level steps for setting up AWS Marketplace Vendor Insights on your AWS Marketplace software as a service (SaaS) listing.

To set up AWS Marketplace Vendor Insights on your SaaS listing

1. [the section called “Create a security profile”](#).
2. (Optional) [the section called “Upload a certification”](#).
3. [the section called “Upload a self-assessment”](#).
4. (Optional) [the section called “Enable AWS Audit Manager automated assessments”](#).

Create a security profile

A security profile provide your buyers with detailed insight into the security posture of your software product. A security profile uses associated data sources, including self-assessments, certifications, and AWS Audit Manager automated assessments.

Note

You can create a limited number of security profiles. To create more security profiles, request a quota increase. For more information, see [AWS service quotas](#) in the *AWS General Reference*.

To create a security profile

1. Sign in using an IAM user or role with access to the AWS Marketplace seller account.
2. Choose **Products** and select **SaaS** to navigate to the **SaaS products** page.
3. Choose a **product**.
4. Choose the **Vendor Insights** tab, and then choose **Contact Support for adding security profile**.
5. Complete the form, and then choose **Submit**.

The AWS Marketplace Seller Operations team will create the security profile. When the security profile is ready, they will send a notification email message to the recipients identified on the form.

Upload a certification

A certification is a data source that provides evidence of your product's security posture across multiple dimensions. AWS Marketplace Vendor Insights supports the following certifications:

- FedRAMP certification – Validates compliance with U.S. government cloud security standards
- GDPR compliance report – Demonstrates adherence to General Data Protection Regulation (GDPR) requirements, protecting personal data and individuals' rights to privacy
- HIPAA compliance report – Demonstrates adherence to Health Insurance Portability and Accountability Act (HIPAA) regulations, safeguarding protected health information
- ISO/IEC 27001 audit report – Confirms compliance with International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 27001, emphasizing information security standards
- PCI DSS audit report – Demonstrates compliance with security standards set by the PCI Security Standards Council
- SOC 2 Type 2 audit report – Confirms compliance with Service Organizational Control (SOC) data privacy and security controls

To upload a certification

1. On the **Vendor Insights** tab, navigate to the **Data sources** section.
2. Under **Certifications**, choose **Upload certification**.

3. Under **Certification details**, provide the requested information and upload the certification.
4. (Optional) Under **Tags**, add new tags.

Note

For information about tags, see [Tagging your AWS resources](#) in the *Tagging AWS Resources User Guide*.

5. Choose **Upload certification**.

Note

The certification is automatically associated with the current security profile. You can also associate certifications that you've already uploaded. On the product detail page, choose **Associate certification** under **Certifications**, select a certification from the list, and choose **Associate certification**.

After you upload the certification, you can download it using the **Download certification** button on the product detail page. You can also update the certification details using the **Update certification** button.

The certification status changes to **ValidationPending** until the certification details are validated. An alternate status displays during and after the data source is processed:

- **Available** – The data source was uploaded and system validations completed successfully.
- **AccessDenied** – The data source's external source reference is no longer accessible for AWS Marketplace Vendor Insights to read.
- **ResourceNotFound** – The data source's external source reference is no longer available for VendorInsights to read.
- **ResourceNotSupported** – The data source was uploaded but the provided source isn't supported, yet. For details about the validation error, refer to the status message.
- **ValidationPending** – The data source was uploaded but system validations are still running. There's no action item for you at this stage. The status is updated to **Available**, **ResourceNotSupported**, or **ValidationFailed**.
- **ValidationFailed** – The data source was uploaded, but the system validation failed for one or more reasons. For details about the validation error, refer to the status message.

Upload a self-assessment

A self-assessment is a type of data source that provide evidence of your product's security posture. AWS Marketplace Vendor Insights supports the following self-assessments:

- AWS Marketplace Vendor Insights self-assessment
- Consensus Assessment Initiative Questionnaire (CAIQ)

To upload a self-assessment

1. On the **Vendor Insights** tab, navigate to the **Data sources** section.
2. Under **Self-assessments**, choose **Upload self-assessment**.
3. Under **Self-assessment details**, complete the following information:
 - a. **Name** – Enter a name for the self-assessment.
 - b. **Type** – Choose an assessment type from the dropdown list.

Note

If you chose **Vendor Insights Security Self-Assessment**, then choose **Download template** to download the self-assessment. Choose **Yes**, **No**, or **N/A** for each answer in the spreadsheet.

4. To upload the completed assessment, choose **Upload self-assessment**.
5. (Optional) Under **Tags**, add new tags.

Note

For information about tags, see [Tagging your AWS resources](#) in the *Tagging AWS Resources User Guide*.

6. Choose **Upload self-assessment**.

Note

The self-assessment is automatically associated with the current security profile. You can also associate self-assessments that you've already uploaded. On the product

detail page, choose **Associate self-assessment** under **Self-assessments**, select a self-assessment from the list, and choose **Associate self-assessment**.

After you upload a self-assessment, you can download it using the **Download self-assessment** button on the product detail page. You can also update the self-assessment details using the **Update self-assessment** button.

The status is updated to one of the following:

- **Available** – The data source was uploaded and system validations completed successfully.
- **AccessDenied** – The data source's external source reference is no longer available for VendorInsights to read.
- **ResourceNotFound** – The data source's external source reference is no longer available for VendorInsights to read.
- **ResourceNotSupported** – The data source was uploaded but the provided source isn't supported, yet. For details about the validation error, refer to the status message.
- **ValidationPending** – The data source was uploaded, but system validations are still running. There's no action item for you at this stage. The status is updated to **Available**, **ResourceNotSupported**, or **ValidationFailed**.
- **ValidationFailed** – The data source was uploaded, but the system validation failed for one or more reasons. For details about the validation error, refer to the status message.

Enable AWS Audit Manager automated assessments


AWS Marketplace Vendor Insights uses multiple AWS services to automatically gather evidence for your security profile.

You need the following AWS services and resources for automated assessments:

- **AWS Audit Manager** – To simplify AWS Marketplace Vendor Insights setup, we use AWS CloudFormation Stacks and StackSets, which take care of provisioning and configuring the necessary resources. The stack set creates an automated assessment containing controls that are automatically populated by AWS Config.

For more information about AWS Audit Manager, see the [AWS Audit Manager User Guide](#).

- **AWS Config** – The stack set deploys an AWS Config conformance pack to set up the necessary AWS Config rules. These rules allow the Audit Manager automated assessment to gather live evidence for other AWS services deployed in that AWS account. For more information about AWS Config features, see the [AWS Config Developer Guide](#).

 **Note**

You might notice increased activity in your account during your initial month of recording with AWS Config when compared to subsequent months. During the initial bootstrapping process, AWS Config reviews all the resources in your account that you have selected for AWS Config to record.

If you're running ephemeral workloads, you might see increased activity from AWS Config as it records configuration changes associated with creating and deleting these temporary resources. An *ephemeral workload* is a temporary use of computing resources that are loaded and run when needed. Examples of ephemeral workloads include Amazon Elastic Compute Cloud (Amazon EC2) spot instances, Amazon EMR jobs, AWS Auto Scaling, and AWS Lambda. To avoid the increased activity from running ephemeral workloads, you can run these types of workloads in a separate account with AWS Config turned off. This approach avoids increased configuration recording and rule evaluations.

- **Amazon S3** – The stack set creates the following two Amazon Simple Storage Service (Amazon S3) buckets:
 - **vendor-insights-stack-set-output-bucket-{account number}** – This bucket contains outputs from the stack set run. The AWS Marketplace Seller Operations team uses the outputs to complete your automated data source creation process.
 - **vendor-insights-assessment-reports-bucket-{account number}** – AWS Audit Manager publishes assessment reports to this Amazon S3 bucket. For more information about publishing assessment reports, see [Assessment reports](#) in the *AWS Audit Manager User Guide*.

For more information about Amazon S3 features, see the [Amazon S3 User Guide](#).

- **IAM** – The onboarding stack set provisions the following AWS Identity and Access Management (IAM) roles in your account:
 - When the `VendorInsightsPrerequisiteCFT.yml` template is deployed, it creates the administrator role `AWSVendorInsightsOnboardingStackSetsAdmin` and the run role `AWSVendorInsightsOnboardingStackSetsExecution`. The stack set uses the administrator role to deploy the required stacks into multiple AWS Regions simultaneously.

The administrator role assumes the execution role to deploy the necessary parent and nested stacks as part of the AWS Marketplace Vendor Insights setup process. For more information about self-managed permissions, see [Grant self-managed permissions](#) in the *AWS CloudFormation User Guide*.

- The `AWSVendorInsightsRole` role provides AWS Marketplace Vendor Insights with access to read the assessments in AWS Audit Manager resources. AWS Marketplace Vendor Insights displays the evidence found on the assessments on your AWS Marketplace Vendor Insights profile.
- The `AWSVendorInsightsOnboardingDelegationRole` provides AWS Marketplace Vendor Insights with access to list and read objects in the `vendor-insights-stack-set-output-bucket` bucket. This capability allows the AWS Marketplace Catalog Operations team to assist you with setting up an AWS Marketplace Vendor Insights profile.
- The `AWSAuditManagerAdministratorAccess` role provides administrative access to enable or disable AWS Audit Manager, update settings, and manage assessments, controls, and frameworks. You or your team can assume this role to take actions for automated assessments in AWS Audit Manager.

To enable AWS Audit Manager automated assessments, you must deploy the onboarding stacks.

Deploy the onboarding stacks

To simplify AWS Marketplace Vendor Insights setup, we use AWS CloudFormation Stacks and StackSets, which take care of provisioning and configuring the necessary resources. If you have a multiple account or multiple AWS Region SaaS solution, StackSets allow you to deploy the onboarding stacks from a central management account.

For more information about CloudFormation StackSets, see [Working with AWS CloudFormation StackSets](#) in the *AWS CloudFormation User Guide*.

AWS Marketplace Vendor Insights setup requires that you use the following CloudFormation templates:

- `VendorInsightsPrerequisiteCFT` – Sets up the necessary administrator role and permissions to run CloudFormation StackSets in your account. Create this stack in your seller account.
- `VendorInsightsOnboardingCFT` – Sets up the required AWS services and configures the appropriate IAM permissions. These permissions allow AWS Marketplace Vendor Insights to

gather data for the SaaS product running in your AWS accounts and display the data on your AWS Marketplace Vendor Insights profile. Create this stack in both your seller account and production accounts that are hosting your SaaS solution through StackSets.

Create the VendorInsightsPrerequisiteCFT stack

By running the VendorInsightsPrerequisiteCFT CloudFormation stack, you set up IAM permissions to start onboarding stack sets.

To create the VendorInsightsPrerequisiteCFT stack

1. Review and download the latest VendorInsightsPrerequisiteCFT.yml file from the [AWS Samples Repo for Vendor Insights templates folder](#) on the GitHub website.
2. Sign in to the AWS Management Console using your AWS Marketplace seller account, and then open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation>.
3. In the CloudFormation console navigation pane, choose **Stacks**, and then choose **Create stack** and **With new resources (standard)** from the dropdown. (If the navigation pane is not visible, in the upper left corner, select and expand the navigation pane.)
4. Under **Specify template**, choose **Upload a template file**. To upload the VendorInsightsPrerequisiteCFT.yml file that you downloaded, use **Choose file**. Then choose **Next**.
5. Enter a name for the stack, and then choose **Next**.
6. (Optional) Configure the stack options as you want.

Choose **Next**.

7. On the **Review** page, review your choices. To make changes, choose **Edit** in the area in which you want to change. Before you can create the stack, you must select the acknowledgement check boxes in the **Capabilities** area.

Choose **Submit**.

8. After the stack is created, choose the **Resources** tab and make note of the following roles that are created:
 - AWSVendorInsightsOnboardingStackSetsAdmin
 - AWSVendorInsightsOnboardingStackSetsExecution

Create the VendorInsightsOnboardingCFT stack set

By running the VendorInsightsOnboardingCFT CloudFormation stack set, you set up the required AWS services and configure the appropriate IAM permissions. This allows AWS Marketplace Vendor Insights to gather data for the SaaS product running in your AWS account and display it in your AWS Marketplace Vendor Insights profile.

If you have a multiple account solution or if you have separate seller and production accounts, you must deploy this stack across multiple accounts. StackSets allow you to do this from the management account that you created the prerequisites stack on.

The stack set is deployed using self-managed permissions. For more information, see [Create a stack set with self-managed permissions](#) in the *AWS CloudFormation User Guide*.

To create the VendorInsightsOnboardingCFT stack set

1. Review and download the latest VendorInsightsOnboardingCFT .yaml file from the [AWS Samples Repo for Vendor Insights templates folder](#) on the GitHub website.
2. Sign in to the AWS Management Console using your AWS Marketplace seller account, and then open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation..>
3. In the CloudFormation console navigation pane, choose **Create StackSet**. (If the navigation pane is not visible, in the upper left corner, select and expand the navigation pane.)
4. Under **Permissions**, for the administrator role choose **IAM role name**, and then choose **AWSVendorInsightsOnboardingStackSetsAdmin** for the role name from the dropdown.
5. Enter **AWSVendorInsightsOnboardingStackSetsExecution** as the **IAM execution role name**.
6. Under **Specify template**, choose **Upload a template file**. To upload the VendorInsightsOnboardingCFT .yaml file that you downloaded, use **Choose file** and then choose **Next**.
7. Provide the following StackSet parameters, and then choose **Next**.
 - **CreateVendorInsightsAutomatedAssessment** – This parameter sets up the AWS Audit Manager automated assessment in your AWS account. If you have separate management and production accounts, this option should only be selected for production accounts and *not* for the management account.
 - **CreateVendorInsightsIAMRoles** – This parameter provisions an IAM role that allows AWS Marketplace Vendor Insights to read the assessment data in your AWS account.

- **PrimaryRegion** – This parameter sets the primary AWS Region for your SaaS deployment. This is the Region where the S3 bucket is created in your AWS account. If your SaaS product is deployed to only one Region, that Region is the primary Region.
8. Configure the StackSet options as you want. Keep the **Execution** configuration as **Inactive**, and then choose **Next**.
 9. Configure the deployment options. If you have a multiple account solution, you can configure the stack set to deploy across multiple accounts and Regions as a single operation. Choose **Next**.

Note

If you have a multiple account solution, we do *not* recommend deploying to all accounts as a single stack set. Pay close attention to the parameters defined in step 7. You might want to enable or disable some parameters, depending on the type of accounts that you're deploying to. StackSets apply the same parameters to all specified accounts in a single deployment. You can reduce deployment time by grouping accounts in a stack set, but you still need to deploy multiple times for a multiple account solution.

Important

If you're deploying to multiple Regions, the first Region that you list must be the **PrimaryRegion**. Leave the **Region Concurrency** option as the default setting of **Sequential**.

10. On the **Review** page, review your choices. To make changes, choose **Edit** in the area in which you want to change. Before you can create the stack set, you must select the acknowledgement check box in the **Capabilities** area.

Choose **Submit**.

The stack set takes about 5 minutes per Region to complete.

Viewing your AWS Marketplace Vendor Insights profile

Your profile in AWS Marketplace Vendor Insights provides important information for buyers to use as they assess your product. For data protection purposes, we recommend that you protect your AWS account credentials and set up individual users with AWS Identity and Access Management (IAM). With that approach, each user is given only the permissions necessary to fulfill their job duties. For more information about creating users with IAM, see [the section called “Creating or using groups”](#).

Buyers can assess your product by using the AWS Marketplace Vendor Insights dashboard. There, buyers can see a product overview that is defined by the data sources you add to your profile. The security profile is defined by multiple security controls in 10 categories.

The 10 security categories used to define data are as follows:

- Access management
- Application security
- Audit, compliance, and security policy
- Business resiliency
- Data security and privacy
- End user device and mobile security
- Human resources
- Infrastructure security
- Risk management and incident response
- Security and configuration policy

For more information see [Understanding control categories](#), in the *AWS Marketplace Buyer Guide*.

By setting up and using AWS Marketplace Vendor Insights, you agree to comply with AWS service terms and data privacy rules to keep user information private and secure. For more information about AWS data privacy terms, see [Data Privacy FAQ](#). For more information about service terms, see [AWS service terms](#).

View your security profile as a seller

After completing the self-assessment and adding other live evidence, it's important to view your profile as a seller. You will want to review the information added to your profile.

Note

This profile isn't visible to buyers until you request that the AWS Marketplace Vendor Insights support team update its visibility. After the support team completes the update, the security profile is accessible to buyers that signed your nondisclosure agreement (NDA). If you want to delete a subscriber's personally identifiable information (PII) data from your AWS Marketplace Vendor Insights profile, start a support case by contacting [AWS Support](#).

To view your security profile as buyers view it

1. Sign in to the AWS Management Console.
2. Go to the [SaaS Product](#) page in the portal.
3. Choose the product with an associated security profile.
4. Select the **Vendor Insights** tab, and then choose **View Latest Released Snapshot**.
5. On the **Overview** tab, all the certificate badges you uploaded are displayed.
6. Select the **Security and compliance** tab, where you can view data gathered from multiple controls. To view more details, choose each control set.

Managing snapshots in AWS Marketplace Vendor Insights

A *snapshot* is a point-in-time posture of a security profile. In AWS Marketplace Vendor Insights, you can use snapshots to assess a seller's product at any given time. As the seller, you can compare the security postures of your profile at different times or the latest snapshots of different security profiles to support your decision making. Snapshots provide necessary security information in addition to providing transparency about freshness and source of the data.

In the AWS Marketplace console, in the AWS Marketplace Vendor Insights **Snapshot summary** section, you can view the following snapshot details for the creation and release schedule:

- **Last created snapshot** – Snapshot last created for this profile.
- **Next scheduled creation** – Snapshot scheduled to be created next.
- **Creation frequency** – Length of time between snapshot creations or the frequency of creating snapshots.
- **Next scheduled release** – Snapshot scheduled to be released next.

- **Staging time** – Snapshot is staged for at least this length of time and then eligible to be released during a snapshot release event.
- **Release frequency** – Length of time between release events.

In the **Snapshot list** section, the snapshot statuses are as follows:

- **Released** – Snapshot is public and available to view for users with permission to this product.
- **Pending release** – Snapshot completed or is in the mandatory minimum staging period and scheduled for the next release.
- **Private** – Snapshot created before security profile activation or had validation errors and isn't visible to the public. Private snapshots remain only in seller visibility.

Topics

- [Create a snapshot](#)
- [View a snapshot](#)
- [Export a snapshot](#)
- [View latest released snapshot](#)
- [Postpone a snapshot release](#)
- [Change preferences for the snapshot list](#)

Create a snapshot

To create a snapshot for your profile, follow these steps. You can create a maximum of 20 snapshots per day.

1. Sign in to the AWS Management Console and open the [AWS Marketplace console](#).
2. Choose **Vendor Insights**.
3. From **Vendor Insights**, choose a product.
4. On the product profile page, go to the **Snapshot list**, and choose **Create new snapshot**.
5. A message notifies you that the snapshot schedule will change. Choose **Create**.

Note

The snapshot schedule changes when a new snapshot is created. New snapshots are scheduled for the same time as your manually created snapshot. This message includes the new schedule.

The new snapshot is created within 30 minutes and added to the snapshot list. New snapshots are created with a **Pending release** status. No one can view new snapshots until the status changes to **Released**.

View a snapshot

To view a snapshot for your profile, follow these steps.

1. Sign in to the AWS Management Console and open the [AWS Marketplace console](#).
2. Choose **Vendor Insights**.
3. From **Vendor Insights**, choose a product.
4. On the product profile page, go to the **Snapshot list**, and choose the **Snapshot ID** of the snapshot that you want to view.
5. When you're finished, choose **Back** to exit the snapshot view.

Export a snapshot

You can export to JSON or CSV formats. To export a snapshot, follow these steps.

1. Sign in to the AWS Management Console and open the [AWS Marketplace console](#).
2. Choose **Vendor Insights**.
3. From **Vendor Insights**, choose a product.
4. On the product profile page, go to the **Snapshot list**, and choose the **Snapshot ID** of the snapshot that you want to export.
5. Choose **Export**.
6. From the dropdown list, choose **Download (JSON)** or **Download (CSV)**.

View latest released snapshot

The latest released snapshot is what users use to view and assess your product's health. It's important to know what is in your latest released snapshot to ensure that you're portraying your product with accurate information. To view the latest snapshot for your profile, follow these steps.

1. Sign in to the AWS Management Console and open the [AWS Marketplace console](#).
2. Choose **Vendor Insights**.
3. From **Vendor Insights**, choose a product.
4. On the product profile page, go to the **Snapshot list**, and choose the **Snapshot ID** of the snapshot that you want to view.
5. Choose **View latest released snapshot**.
6. When you're finished, choose **Back** to exit the snapshot view.

Postpone a snapshot release

To delay the release of a snapshot to your profile, you can postpone a snapshot release for a specific **Snapshot ID**.

1. Sign in to the AWS Management Console and open the [AWS Marketplace console](#).
2. Choose **Vendor Insights**.
3. From **Vendor Insights**, choose a product.
4. On the product profile page, go to the **Snapshot list**, and choose the **Snapshot ID** of the snapshot for which you want to postpone the release.
5. From the **Snapshot summary**, choose **Postpone snapshot release**.
6. A message notifies you that the snapshot schedule will change. Choose **Postpone**.

A success message appears, indicating that you have successfully postponed the snapshot release for this product.

Change preferences for the snapshot list

After creating a snapshot, you can change the preferences of how a snapshot is viewed in the **Snapshot list**.

1. Sign in to the AWS Management Console and open the [AWS Marketplace console](#).
2. Choose **Vendor Insights**.
3. From **Vendor Insights**, choose a product.
4. On the product profile page, go to the **Snapshot list**, and choose the **Snapshot ID** of the snapshot that you want to change.
5. Choose the preferences icon. You can customize the following preferences for your snapshot:
 - **Page size** – Select how many snapshots you want listed on each page: **10 resources**, **20 resources**, or **50 resources** per page.
 - **Wrap lines** – Select an option to wrap lines to view the entire record.
 - **Time format** – Select whether you want **Absolute**, **Relative**, or **ISO**.
 - **Visible columns** – Select options that you want visible for the snapshot details: **Snapshot ID**, **Status**, and **Date created** .

Controlling access in AWS Marketplace Vendor Insights

AWS Identity and Access Management (IAM) is an AWS service that helps you control access to AWS resources. IAM is an AWS service that you can use with no additional charge. If you're an administrator, you control who can be *authenticated* (signed in) and *authorized* (have permissions) to use AWS Marketplace resources. AWS Marketplace Vendor Insights uses IAM to control access to seller data, assessments, seller self-attestation, and industry standard audit reports.

The recommended way to control who can do what in AWS Marketplace Management Portal is to use IAM to create users and groups. Then you add the users to the groups, and manage the groups. You can assign a policy or permissions to the group that provide read-only permissions. If you have other users that need read-only access, you can add them to the group you created rather than adding permissions for the user.

A *policy* is a document that defines the permissions that apply to a user, group, or role. The permissions determine what users can do in AWS. A policy typically allows access to specific actions, and can optionally grant that the actions are allowed for specific resources, like Amazon EC2 instances, Amazon S3 buckets, and so on. Policies can also explicitly deny access. A *permission* is a statement within a policy that allows or denies access to a particular resource.

⚠ Important

All of the users that you create authenticate by using their credentials. However, they use the same AWS account. Any change that a user makes can impact the whole account.

AWS Marketplace has permissions defined to control the actions that someone with those permissions can take in the AWS Marketplace Management Portal. There are also policies that AWS Marketplace created and manages that combine several permissions. The `AWSMarketplaceSellerProductsFullAccess` policy gives the user full access to products in the AWS Marketplace Management Portal.

For more information about the actions, resources, and condition keys that are available, see [Actions, resources, and condition keys for AWS Marketplace Vendor Insights](#) in the *Service Authorization Reference*.

Permissions for AWS Marketplace Vendor Insights sellers

You can use the following permissions in IAM policies for AWS Marketplace Vendor Insights. You can combine permissions into a single IAM policy to grant the permissions you want.

CreateDataSource

`CreateDataSource` allows the user to create a new data source resource. Supported data sources are:

- `SOC2Type2AuditReport`
- `ISO27001AuditReport`
- `AWSAuditManagerSecurityAutomatedAssessment`
- `FedRAMPCertification`
- `GDPRComplianceReport`
- `HIPAAComplianceReport`
- `PCIDSSAuditReport`
- `SecuritySelfAssessment`

Action groups: Read-write

Required resources: None

Creates resources: DataSource

DeleteDataSource

DeleteDataSource allows the user to delete a data source that they own. A data source must be disassociated from any profile to be deleted. For more information, see [the section called "AssociateDataSource"](#).

Action groups: Read-write

Required resources: DataSource

GetDataSource

GetDataSource allows the user to retrieve the details of a data source. Details of a data source include metadata information such as associated timestamps, original creation parameters, and processing information, if any.

Action groups: Read-only, read-write

Required resources: DataSource

UpdateDataSource

UpdateDataSource allows the user to update the details of a data source. Details include metadata information, such as the name and source information (for example, roles, source Amazon Resource Name (ARN), and source content).

Action groups: Read-only, read-write

Required resources: DataSource

ListDataSources

ListDataSources allows the user to list the data sources that they own.

Action groups: Read-only, read-write, list-only

Required resources: None

CreateSecurityProfile

CreateSecurityProfile allows the user to create a new security profile. A security profile is a resource to manage how and when a snapshot is generated. Users can also control how buyers can access snapshots by controlling the status and applicable terms of the profile.

Action groups: Read-only, read-write, list-only

Required resources: None

Creates resources: SecurityProfile

ListSecurityProfiles

ListSecurityProfiles allows the user to list the security profiles that they own.

Action groups: Read-only, read-write, list-only

Required resources: None

GetSecurityProfile

CreateSecurityProfile allows users to get the details of a security profile.

Action groups: Read-only and read-write

Required resources: SecurityProfile

AssociateDataSource

AssociateDataSource allows users to associate an existing DataSource with an AWS Marketplace Vendor Insights profile. Users can control the content of the snapshot by associating or disassociating a data source to a profile.

Action groups: Read-write

Required resources: SecurityProfile and DataSource

DisassociateDataSource

DisassociateDataSource allows users to disassociate an existing DataSource with an AWS Marketplace Vendor Insights profile. Users can control the content of the snapshot by associating or disassociating a data source to a profile.

Action groups: Read-write

Required resources: SecurityProfile and DataSource

UpdateSecurityProfile

UpdateSecurityProfile allows users to modify security profile attributes such as name and description.

Action groups: Read-write

Required resources: SecurityProfile

ActivateSecurityProfile

ActivateSecurityProfile allows users to set an Active status for a security profile. After a security profile is activated, new snapshots can be created in a Staged state which makes it possible to release them if other conditions are met. For more information, see [the section called "UpdateSecurityProfileSnapshotReleaseConfiguration"](#).

An Active security profile with at least one Released snapshot is eligible for AWS Marketplace Vendor Insights discovery for end users.

Action groups: Read-write

Required resources: SecurityProfile

DeactivateSecurityProfile

DeactivateSecurityProfile allows users to set an Inactive status for a security profile. This terminal state for a security profile is equivalent to taking down the profile from shared state. Users can only deactivate a security profile if there are no active subscribers to the profile.

Action groups: Read-write

Required resources: SecurityProfile

UpdateSecurityProfileSnapshotCreationConfiguration

UpdateSecurityProfileSnapshotCreationConfiguration allows users to define custom schedules for the snapshot creation configuration. The default creation configuration of weekly creation can be overridden with this action.

Users can use this action to change the schedule including to cancel a schedule, postpone the schedule to a future date, or initiate a new snapshot creation for an earlier time.

Action groups: Read-write

Required resources: SecurityProfile

UpdateSecurityProfileSnapshotReleaseConfiguration

UpdateSecurityProfileSnapshotReleaseConfiguration allows users to define custom schedules for the snapshot release configuration. The default creation configuration of weekly releases with a two-day staging period to review can be overridden with this action.

Users can use this action to change the schedule including to cancel a schedule or postpone the schedule to a future date.

Action groups: Read-write

Required resources: SecurityProfile

ListSecurityProfileSnapshots

ListSecurityProfileSnapshots allows users to list the snapshots for a security profile that they own.

Action groups: Read-only, list-only, and read-write

Required resources: SecurityProfile

GetSecurityProfileSnapshot

GetSecurityProfileSnapshot allows users to get the snapshots for a security profile that they own.

Action groups: Read-only and read-write

Required resources: SecurityProfile

TagResource

TagResource allows users to add new tags to a resource. Supported resources are SecurityProfile and DataSource.

Action groups: Tagging

Optional resources: SecurityProfile and DataSource

UntagResource

UntagResource allows users to remove tags from a resource. Supported resources are SecurityProfile and DataSource.

Action groups: Tagging

Optional resources: SecurityProfile and DataSource

ListTagsForResource

ListTagsForResource allows users to list resource tags for a resource. Supported resources are SecurityProfile and DataSource.

Action groups: Read-only

Optional resources: SecurityProfile and DataSource

Additional resources

The following resources in the *IAM User Guide* provide more information about getting started and using IAM:

- [Security best practices in IAM](#)
- [Managing IAM policies](#)
- [Attaching a policy to a user group](#)
- [IAM Identities \(users, user groups, and roles\)](#)
- [Creating your first user and user group](#)
- [Managing IAM policies](#)
- [Controlling access to AWS resources using policies](#)

AWS Marketplace security

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The [shared responsibility model](#) describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. The effectiveness of our security is regularly tested and verified by third-party auditors as part of the [AWS compliance programs](#). To learn about the compliance programs that apply to AWS Marketplace, see [AWS Services in Scope by Compliance Program](#).
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You're also responsible for other factors including the sensitivity of your data, your organization's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using AWS Marketplace. The following topics show you how to configure AWS Identity and Access Management to manage access to AWS Marketplace in order to meet your security and compliance objectives. You can also learn how to use other AWS services that can help you to monitor and secure your AWS Marketplace resources.

To learn more about security and other policies regarding the products that you offer in AWS Marketplace, see the following topics:

- [AMI-based product requirements](#)
- [Container-based product requirements](#)
- [SaaS product guidelines](#)
- [Requirements for professional services products](#)

Note

To learn about security on AWS Data Exchange for data products, see [Security](#) in the *AWS Data Exchange User Guide*.

To learn about security for buyers in AWS Marketplace, see [Security on AWS Marketplace in the AWS Marketplace Buyer Guide](#).

Topics

- [Controlling access to AWS Marketplace Management Portal](#)
- [Policies and permissions for AWS Marketplace sellers](#)
- [AWS managed policies for AWS Marketplace sellers](#)
- [AWS Marketplace Commerce Analytics Service account permissions](#)
- [Amazon SQS permissions](#)
- [AWS Marketplace metering and entitlement API permissions](#)
- [Using service-linked roles for AWS Marketplace](#)
- [Logging AWS Marketplace Metering API calls with AWS CloudTrail](#)

Controlling access to AWS Marketplace Management Portal

AWS Identity and Access Management (IAM) is an AWS service that helps you control access to AWS resources. If you are an administrator, you control who can be *authenticated* (signed in) and *authorized* (have permissions) to use AWS Marketplace resources. IAM is an AWS service that you can use with no additional charge.

The recommended way to control who can do what in AWS Marketplace Management Portal is to use IAM to create users and groups. Then you add the users to the groups, and manage the groups. For example, if John should be allowed to view your products, create a user for him and add his user to a group you create for read-only access. You can assign a policy or permissions to the group that provide read-only permissions. If you have other users that need read-only access, you can add them to the group you created rather than adding permissions to the user. If John's role changes and he no longer needs read-only access, you can remove John from the group.

A *policy* is a document that defines the permissions that apply to a user, group, or role. In turn, the permissions determine what users can do in AWS. A policy typically allows access to specific actions, and can optionally grant that the actions are allowed for specific resources, like Amazon EC2 instances, Amazon S3 buckets, and so on. Policies can also explicitly deny access. A *permission* is a statement within a policy that allows or denies access to a particular resource. You can state any permission like this: "A has permission to do B to C." For example, Jane (A) has permission to

read messages (B) from John's Amazon Simple Queue Service queue (C). Whenever Jane sends a request to Amazon SQS to use John's queue, the service checks to see if she has permission. It further checks to see if the request satisfies the conditions John specified in the permission.

Important

All of the users that you create authenticate by using their credentials. However, they use the same AWS account. Any change that a user makes can impact the whole account.

AWS Marketplace has permissions defined to control the actions that someone with those permissions can take in AWS Marketplace Management Portal. There are also policies that AWS Marketplace created and manage that combine several permissions.

The following resources provide more information about getting started and using IAM.

- [Create an administrative user](#)
- [Security best practices in IAM](#)
- [Managing IAM policies](#)
- [Attaching a policy to an IAM user group](#)
- [IAM Identities \(users, groups, and roles\)](#)
- [Controlling access to AWS resources using policies](#)

The following topics provide some high-level guidance for creating users and groups, and signing in as a user.

Topics

- [Creating users](#)
- [Creating or using groups](#)
- [Signing in as a user](#)

Creating users

To allow people in your company to sign in to the AWS Marketplace Management Portal, create a user for each person who needs access.

To create users

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Users** and then choose **Create New Users**.
3. In the numbered text boxes, enter a name for each user that you want to create.
4. Clear the **Generate an access key for each user** check box and then choose **Create**.

To assign a password to each user that you just created

1. In the list of users, choose the name of a new user.
2. Choose the **Security Credentials** tab and then choose **Manage Password**.
3. Choose an option for either an auto-generated password or a custom password. Optionally, to require the user to choose a new password at the next sign-in, select the box for **Require user to create a new password at next sign-in**. Choose **Apply**.
4. Choose **Download Credentials** to save the sign-in credentials and account-specific sign-in URL to a comma-separated values (CSV) file on your computer. Then choose **Close**.

Note

To sign in with the sign-in credentials that you just created, users must navigate to your account-specific sign-in URL. This URL is in the credentials file that you just downloaded and is also available on the IAM console. For more information, see [How IAM users sign in to your AWS account](#) in the *IAM User Guide*.

Tip

Create sign-in credentials for yourself as well, even though you're the AWS account owner. It's a recommended best practice for everyone to work in AWS Marketplace as a user, even the account owner. For instructions on how to create a user for yourself that has administrative permissions, see [Create an administrative user](#) in the *IAM User Guide*.

Creating or using groups

After you create users, create groups, create permissions to access the pages in the AWS Marketplace Management Portal, add those permissions to the groups, and then add users to the groups.

When you assign permissions to a group, you allow any member of that group to perform specific actions. When you add a new user to the group, that user automatically gains the permissions that are assigned to the group. A group can have permissions for more than one action. We recommend using an [AWS Marketplace managed policy](#) rather than creating your own policy.

To assign a managed policy for AWS Marketplace to a group

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Groups**, and then choose the group that you want to attach a policy to.
3. On the summary page for the group, under the **Permissions** tab, choose **Attach Policy**.
4. On the **Attach Policy** page, next to **Filter:** enter **awsmarketplace**.
5. Choose the policy or policies that you want to attach, and then choose **Attach Policy**.

To create a policy with AWS Marketplace Management Portal permissions

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Policies** and then choose **Create Policy**.
3. Next to **Policy Generator**, choose **Select**.
4. On the **Edit Permissions** page, do the following:
 - a. For **Effect**, choose **Allow**.
 - b. For **AWS Service**, choose **AWS Marketplace Management Portal**.
 - c. For **Actions**, select the permission or permissions to allow.
 - d. Choose **Add Statement**.
 - e. Choose **Next Step**.
5. On the **Review Policy** page, do the following:
 - a. For **Policy Name**, enter a name for this policy. Take note of the policy name because you need it for a later step.

- b. (Optional) For **Description**, enter a description for this policy.
- c. Choose **Create Policy**.

To create an IAM group with appropriate permissions and add users to the group

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Groups** and then choose **Create New Group**.
3. For **Group Name:**, type a name for the group. Then choose **Next Step**.
4. On the **Attach Policy** page, do the following:
 - a. For **Filter:**, choose **Customer Managed Policies**.
 - b. Select the check box next to the name of the policy that you want to attach to this group. This is typically the policy that you just created.
 - c. Choose **Next Step**.
5. Choose **Create Group**.
6. Find your new group in the **Groups** list and then select the check box next to it. Choose **Group Actions** and then **Add Users to Group**.
7. Select the check box next to each user to add to the group and then choose **Add Users**.

Signing in as a user

After you have created users in IAM, users can sign in with their own sign-in credentials. To do so, they need to use the unique URL that is associated with your AWS account. You can get and distribute the sign-in URL to your users.

To get your account's unique sign-in URL

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Dashboard**.
3. Near the top of the content pane, find **IAM users sign-in link:** and take note of the sign-in link, which has a format like this:

```
https://AWS_account_ID.signin.aws.amazon.com/console/
```

Note

If you want the URL for your sign-in page to contain your company name (or other friendly identifier) instead of your AWS account ID, you can create an alias for your account by choosing **Customize**. For more information, see [Your AWS Account ID and Its Alias](#) in the *IAM User Guide*.

4. Distribute this URL to the people at your company who can work with AWS Marketplace, along with the sign-in credentials that you created for each. Instruct them to use your account's unique sign-in URL to sign in before they access AWS Marketplace.

Policies and permissions for AWS Marketplace sellers

AWS Marketplace has three managed policies you can use with the AWS Marketplace Management Portal. In addition, you can use individual permissions to create your own AWS Identity and Access Management (IAM) policy.

AWS Marketplace has several managed policies that you can use with the AWS Marketplace Management Portal. In addition, you can use individual permissions to create your own AWS Identity and Access Management (IAM) policy.

You can also provide fine-grained access to the AWS Marketplace Management Portal for the **Settings**, **Contact Us**, **File Upload**, and **Insights** tabs. Using fine-grained access, you can do the following:

- Grant other people permission to administer and use resources in your AWS account without sharing your password or access key.
- Grant granular permissions to multiple people for various resources. For example, you might allow some users access to view the **Settings** tab in the AWS Marketplace Management Portal. For other users, you might allow access to edit in the **Settings** and **Contact Us** tabs.

Note

For more information about policies and permissions in AWS Data Exchange for data products, see [Identity and Access Management in AWS Data Exchange](#) in the *AWS Data Exchange User Guide*.

For more information about policies and permissions for AWS Marketplace buyers, see [Controlling access to AWS Marketplace subscriptions](#) in the *AWS Marketplace Buyer Guide*.

Policies for AWS Marketplace sellers

You can use the following managed policies to provide users with controlled access to the AWS Marketplace Management Portal:

AWSMarketplaceSellerFullAccess

Allows full access to all of the pages in the AWS Marketplace Management Portal and other AWS services, such as Amazon Machine Image (AMI) management.

AWSMarketplaceSellerProductsFullAccess

Allows full access to the [Products](#) pages in the AWS Marketplace Management Portal.

AWSMarketplaceSellerProductsReadOnly

Allows read-only access to the [Products](#) pages in the AWS Marketplace Management Portal.

Important

AWS Marketplace buyers can use managed policies to manage the subscriptions they purchase. The names of the managed policies that you use with AWS Marketplace Management Portal start with `AWSMarketplaceSeller`. When you search for policies in IAM, make sure to search for policy names that start with `AWSMarketplaceSeller`.

AWS Marketplace also provides specialized managed policies for specific scenarios. For a full list of AWS managed policies for AWS Marketplace sellers and descriptions of what permissions they provide, see [AWS managed policies for AWS Marketplace sellers](#).

Permissions for AWS Marketplace sellers

You can use the following permissions in IAM policies for the AWS Marketplace Management Portal:

aws-marketplace-management:PutSellerVerificationDetails

Allows access to start the Know Your Customer (KYC) process.

aws-marketplace-management:GetSellerVerificationDetails

Allows access to view the KYC status in the AWS Marketplace Management Portal.

aws-marketplace-management:PutBankAccountVerificationDetails

Allows access to start the [bank account verification](#) process.

aws-marketplace-management:GetBankAccountVerificationDetails

Allows access to view the bank account verification status in the AWS Marketplace Management Portal.

aws-marketplace-management:PutSecondaryUserVerificationDetails

Allows access to add secondary users in the AWS Marketplace Management Portal.

aws-marketplace-management:GetSecondaryUserVerificationDetails

Allows access to view the secondary user status in the AWS Marketplace Management Portal.

aws-marketplace-management:GetAdditionalSellerNotificationRecipients

Allows access to view email contacts for AWS Marketplace notifications.

aws-marketplace-management:PutAdditionalSellerNotificationRecipients

Allows access to update email contacts for AWS Marketplace notifications.

tax:PutTaxInterview

Allows access to take the [tax interview](#) in the AWS Marketplace Management Portal.

tax:GetTaxInterview

Allows access to view the tax interview status in the AWS Marketplace Management Portal.

tax:GetTaxInfoReportingDocument

Allows AWS Marketplace sellers to view and download tax documents (for example, 1099-K forms) from the Tax dashboard

payments:CreatePaymentInstrument

Allows access to add a bank account to the AWS Marketplace Management Portal.

payments:GetPaymentInstrument

Allows access to view existing bank accounts in the AWS Marketplace Management Portal.

aws-marketplace:ListTasks

Allows access to view a list of tasks pending seller action.

aws-marketplace:DescribeTask

Allows access to view the details of any tasks pending seller action.

aws-marketplace:UpdateTask

Allows access to edit a task pending seller action.

aws-marketplace:CompleteTask

Allows access to submit edits made to a task pending seller action.

support:CreateCase

Allows access to create an AWS Marketplace case within the AWS Marketplace Management Portal.

aws-marketplace-management:viewSupport

Allows access to the [Customer Support Eligibility](#) page in the AWS Marketplace Management Portal.

aws-marketplace-management:viewReports

Allows access to the [Reports](#) page in the AWS Marketplace Management Portal.

aws-marketplace:ListEntities

Allows access to list objects in AWS Marketplace Management Portal. Required to access the [File Upload](#), [Offers](#) and [Partners](#) pages in the AWS Marketplace Management Portal.

Note

To allow access to view the **Settings** tab, you can use this permission, the `ListEntity` permission, and the following Amazon Resource Name (ARN): `arn:{partition}:{aws-marketplace}:{region}:{account-id}:AWSMarketplace/Seller/{entity-id}`.

aws-marketplace:DescribeEntity

Allows access to view details of objects in AWS Marketplace Management Portal. Required to access the [File Upload](#), [Offers](#), [Partners](#), and [Agreements](#) pages in the AWS Marketplace Management Portal.

Note

To allow access to view the **Settings** tab, you can use this permission, the DescribeEntity permission, and the following ARN: `arn:{partition}:{aws-marketplace}:{region}:{account-id}:AWSMarketplace/Seller/*`.

aws-marketplace:StartChangeSet

Allows access to create product changes in AWS Marketplace Management Portal. Required to make changes in the [File Upload](#), [Offers](#), [Partners](#), and [Agreements](#) pages in the AWS Marketplace Management Portal.

Note

To allow access to register as a seller in AWS Marketplace, you can use this permission, the `catalog:ChangeType: "CreateSeller"` condition key, and the following ARN: `arn:{partition}:{aws-marketplace}:{region}:{account-id}:AWSMarketplace/Seller/{entity-id}`.

To allow access to update the seller profile in AWS Marketplace, you can use this permission, the `catalog:ChangeType: "UpdateInformation"` condition key, and the following ARN: `arn:{partition}:{aws-marketplace}:{region}:{account-id}:AWSMarketplace/Seller/{entity-id}`.

To allow access to update disbursement preferences for Amazon Web Services, you can use this permission, the `catalog:ChangeType: "UpdateDisbursementPreferences"` condition key, and the following ARN: `arn:{partition}:{aws-marketplace}:{region}:{account-id}:AWSMarketplace/Seller/{entity-id}`.

aws-marketplace:SearchAgreements

Allows viewing the high-level list of agreements on the [Agreements](#) page, and opportunities between ISVs and channel partners on the [Partners](#) page.

aws-marketplace:DescribeAgreement

Allows viewing of high-level agreement details on the **Agreements** page, and opportunities between ISVs and channel partners on the **Partners** page.

aws-marketplace:GetAgreementTerms

Allows viewing all agreement term details on the **Agreements** page, and opportunities between ISVs and channel partners on the **Partners** page.

aws-marketplace:GetSellerDashboard

Allows access to the dashboards on the **Insights** page in the AWS Marketplace Management Portal.

Note

To enable a user to access the [Manage Products](#) page, you must use either the `AWSMarketplaceSellerProductsFullAccess` or `AWSMarketplaceSellerProductsReadOnly` managed permissions.

You can combine the preceding permissions into a single IAM policy to grant the permissions that you want. See the following examples.

Example 1: Permissions to view the KYC status

To grant permissions to view KYC status in the AWS Marketplace Management Portal, use a policy similar to the following example.

To grant permissions to view the KYC status in the AWS Marketplace Management Portal, use a policy similar to the following example.

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



```

    "aws-marketplace-management:GetSellerVerificationDetails"
  ],
  "Resource": ["*"]
}]
}

```

Example 2: Permissions to create upgrades and renewals for private offers

To grant permissions to view and use the **Agreements** page to create upgrades and renewals for private offers, use a policy similar to the following example.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "aws-marketplace:SearchAgreements",
        "aws-marketplace:DescribeAgreement",
        "aws-marketplace:GetAgreementTerms",
        "aws-marketplace:DescribeEntity",
        "aws-marketplace:StartChangeSet"
      ],
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws-marketplace:PartyType": "Proposer"
        },
        "ForAllValues:StringEquals": {
          "aws-marketplace:AgreementType": [
            "PurchaseAgreement"
          ]
        }
      }
    }
  ]
}

```

Example 3: Permissions to access the Offers page and create new private offers

To grant permissions to view and use the **Offers** page to view existing private offers and create private offers, use a policy similar to the following example.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "aws-marketplace:ListEntities",
        "aws-marketplace:DescribeEntity",
        "aws-marketplace:StartChangeSet"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Example 4: Permissions to access the Settings page

To grant permissions to view and use the **Settings** page, use a policy similar to the following example.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "aws-marketplace:ListEntities",
        "aws-marketplace:DescribeEntity",
        "aws-marketplace:StartChangeSet"
      ],
      "Effect": "Allow",
      "Resource": "arn:{partition}:{aws-marketplace}:{region}:{account-id}:AWSMarketplace/Seller/*",
    }
  ]
}
```

Example 5: Permissions to access the File Upload page

To grant permissions to view and use the **File Upload** page, use a policy similar to the following example.

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Action": [
      "aws-marketplace:ListEntities",
      "aws-marketplace:DescribeEntity",
      "aws-marketplace:StartChangeSet"
    ],
    "Effect": "Allow",
    "Resource": "*"
  }
]
```

Using IAM groups

Alternatively, you can create separate IAM groups for granting access to each individual page in the AWS Marketplace Management Portal. Users can belong to more than one group. So, if a user needs access to more than one page, you can add the user to all of the appropriate groups. For example, create one IAM group and grant that group permission to access the **Insights** page, create another group and grant that group permission to access the **File Upload** page, and so on. If a user needs permission to access both the **Insights** page and the **File Upload** page, add the user to both groups.

For more information about users and groups, see [IAM Identities \(users, groups, and roles\)](#) in the *IAM User Guide*.

AWS managed policies for AWS Marketplace sellers

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining [customer managed policies](#) that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed

policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see [AWS managed policies](#) in the *IAM User Guide*.

This section lists each of the policies used to manage seller access to AWS Marketplace. For information about buyer policies, see [AWS managed policies for AWS Marketplace buyers](#) in the *AWS Marketplace Buyer Guide*.

Topics

- [AWS managed policy: AWSMarketplaceAmiIngestion](#)
- [AWS managed policy: AWSMarketplaceFullAccess](#)
- [AWS managed policy: AWSMarketplaceGetEntitlements](#)
- [AWS managed policy: AWSMarketplaceMeteringFullAccess](#)
- [AWS managed policy: AWSMarketplaceMeteringRegisterUsage](#)
- [AWS managed policy: AWSMarketplaceSellerFullAccess](#)
- [AWS managed policy: AWSMarketplaceSellerProductsFullAccess](#)
- [AWS managed policy: AWSMarketplaceSellerProductsReadOnly](#)
- [AWS managed policy: AWSVendorInsightsVendorFullAccess](#)
- [AWS managed policy: AWSVendorInsightsVendorReadOnly](#)
- [AWS Marketplace updates to AWS managed policies](#)

AWS managed policy: AWSMarketplaceAmiIngestion

You can create a service role with this policy that can then be used by AWS Marketplace to perform actions on your behalf. For more information about using `AWSMarketplaceAmiIngestion`, see [Give AWS Marketplace access to your AMI](#).

This policy is used to grant contributor permissions that allow AWS Marketplace to copy your Amazon Machine Images (AMIs) in order to list them on AWS Marketplace.

Permissions details

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

```

    {
      "Action": [
        "ec2:ModifySnapshotAttribute"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:ec2:us-east-1::snapshot/snap-*"
    },
    {
      "Action": [
        "ec2:DescribeImageAttribute",
        "ec2:DescribeImages",
        "ec2:DescribeSnapshotAttribute",
        "ec2:ModifyImageAttribute"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}

```

AWS managed policy: AWSMarketplaceFullAccess

You can attach the `AWSMarketplaceFullAccess` policy to your IAM identities.

This policy grants administrative permissions that allow full access to AWS Marketplace and related services, both as a seller and a buyer. These permissions include the following abilities:

- Subscribe and unsubscribe to AWS Marketplace software.
- Manage AWS Marketplace software instances from AWS Marketplace.
- Create and manage a private marketplace in your account.
- Provide access to Amazon EC2, AWS CloudFormation, and Amazon EC2 Systems Manager.

Permissions details

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "aws-marketplace:*",

```

```

        "cloudformation:CreateStack",
        "cloudformation:DescribeStackResource",
        "cloudformation:DescribeStackResources",
        "cloudformation:DescribeStacks",
        "cloudformation:List*",
        "ec2:AuthorizeSecurityGroupEgress",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:CreateSecurityGroup",
        "ec2:CreateTags",
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeAddresses",
        "ec2>DeleteSecurityGroup",
        "ec2:DescribeImages",
        "ec2:DescribeInstances",
        "ec2:DescribeKeyPairs",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeTags",
        "ec2:DescribeVpcs",
        "ec2:RunInstances",
        "ec2:StartInstances",
        "ec2:StopInstances",
        "ec2:TerminateInstances"
    ],
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": [
        "ec2:CopyImage",
        "ec2:DeregisterImage",
        "ec2:DescribeSnapshots",
        "ec2>DeleteSnapshot",
        "ec2:CreateImage",
        "ec2:DescribeInstanceStatus",
        "ssm:GetAutomationExecution",
        "ssm:UpdateDocumentDefaultVersion",
        "ssm:CreateDocument",
        "ssm:StartAutomationExecution",
        "ssm:ListDocuments",
        "ssm:UpdateDocument",
        "ssm:DescribeDocument",
        "sns:ListTopics",
        "sns:GetTopicAttributes",
    ]
}

```

```

        "sns:CreateTopic",
        "iam:GetRole",
        "iam:GetInstanceProfile",
        "iam:ListRoles",
        "iam:ListInstanceProfiles"
    ],
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": [
        "s3:ListBucket",
        "s3:GetObject"
    ],
    "Resource": [
        "arn:aws:s3::*image-build*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "sns:Publish",
        "sns:setTopicAttributes"
    ],
    "Resource": "arn:aws:sns:*:*:*image-build*"
},
{
    "Effect": "Allow",
    "Action": [
        "iam:PassRole"
    ],
    "Resource": [
        "*"
    ],
    "Condition": {
        "StringLike": {
            "iam:PassedToService": [
                "ec2.amazonaws.com",
                "ssm.amazonaws.com"
            ]
        }
    }
}
]

```

```
}
```

AWS managed policy: AWSMarketplaceGetEntitlements

You can attach the `AWSMarketplaceGetEntitlements` policy to your IAM identities.

This policy grants read-only permissions that allow software as a service (SaaS) product sellers to check whether a customer has subscribed to their AWS Marketplace SaaS product.

Permissions details

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid" : "AWSMarketplaceGetEntitlements",
      "Effect" : "Allow",
      "Action": [
        "aws-marketplace:GetEntitlements"
      ],
      "Resource": "*"
    }
  ]
}
```

AWS managed policy: AWSMarketplaceMeteringFullAccess

You can attach the `AWSMarketplaceMeteringFullAccess` policy to your IAM identities.

This policy grants contributor permissions that allow reporting metered usage that corresponds to AMI and container products with flexible consumption pricing on AWS Marketplace.

Permissions details

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "aws-marketplace:MeterUsage"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```



```
    }  
  ]  
}
```

AWS managed policy: AWSMarketplaceMeteringRegisterUsage

You can attach the `AWSMarketplaceMeteringRegisterUsage` policy to your IAM identities.

This policy grants contributor permissions that allow reporting metered usage that corresponds to container products with hourly pricing on AWS Marketplace.

Permissions details

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Action": [  
        "aws-marketplace:RegisterUsage"  
      ],  
      "Effect": "Allow",  
      "Resource": "*"   
    }  
  ]  
}
```

AWS managed policy: AWSMarketplaceSellerFullAccess

You can attach the `AWSMarketplaceSellerFullAccess` policy to your IAM identities.

This policy grants administrative permissions that allow full access to all seller operations on AWS Marketplace, including AWS Marketplace Management Portal, and managing the Amazon EC2 AMI used in AMI-based products.

Permissions details

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

```

    "Action": [
      "aws-marketplace-management:viewMarketing",
      "aws-marketplace-management:viewReports",
      "aws-marketplace-management:viewSupport",
      "aws-marketplace:ListChangeSets",
      "aws-marketplace:DescribeChangeSet",
      "aws-marketplace:StartChangeSet",
      "aws-marketplace:CancelChangeSet",
      "aws-marketplace:ListEntities",
      "aws-marketplace:DescribeEntity",
      "aws-marketplace:ListTasks",
      "aws-marketplace:DescribeTask",
      "aws-marketplace:UpdateTask",
      "aws-marketplace:CompleteTask",
      "aws-marketplace:GetSellerDashboard",
      "ec2:DescribeImages",
      "ec2:DescribeSnapshots",
      "ec2:ModifyImageAttribute",
      "ec2:ModifySnapshotAttribute"
    ],
    "Resource": "*"
  },
  {
    "Sid": "AgreementAccess",
    "Action": [
      "aws-marketplace:SearchAgreements",
      "aws-marketplace:DescribeAgreement",
      "aws-marketplace:GetAgreementTerms"
    ],
    "Effect": "Allow",
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "aws-marketplace:PartyType": "Proposer"
      },
      "ForAllValues:StringEquals": {
        "aws-marketplace:AgreementType": [
          "PurchaseAgreement"
        ]
      }
    }
  },
  {
    "Sid": "IAMGetRole",

```

```

    "Effect": "Allow",
    "Action": [
        "iam:GetRole"
    ],
    "Resource": "arn:aws:iam::*:role/*"
},
{
    "Sid": "AssetScanning",
    "Effect": "Allow",
    "Action": [
        "iam:PassRole"
    ],
    "Resource": "arn:aws:iam::*:role/*",
    "Condition": {
        "StringEquals": {
            "iam:PassedToService": "assets.marketplace.amazonaws.com"
        }
    }
},
{
    "Sid": "VendorInsights",
    "Effect": "Allow",
    "Action": [
        "vendor-insights:GetDataSource",
        "vendor-insights:ListDataSources",
        "vendor-insights:ListSecurityProfiles",
        "vendor-insights:GetSecurityProfile",
        "vendor-insights:GetSecurityProfileSnapshot",
        "vendor-insights:ListSecurityProfileSnapshots"
    ],
    "Resource": "*"
},
{
    "Sid": "TagManagement",
    "Effect": "Allow",
    "Action": [
        "aws-marketplace:TagResource",
        "aws-marketplace:UntagResource",
        "aws-marketplace:ListTagsForResource"
    ],
    "Resource": "arn:aws:aws-marketplace::*:AWSMarketplace/*"
},
{
    "Sid": "SellerSettings",

```

```

    "Effect": "Allow",
    "Action": [
        "aws-marketplace-management:GetSellerVerificationDetails",
        "aws-marketplace-management:PutSellerVerificationDetails",
        "aws-marketplace-management:GetBankAccountVerificationDetails",
        "aws-marketplace-management:PutBankAccountVerificationDetails",
        "aws-marketplace-management:GetSecondaryUserVerificationDetails",
        "aws-marketplace-management:PutSecondaryUserVerificationDetails",
        "aws-marketplace-management:GetAdditionalSellerNotificationRecipients",
        "aws-marketplace-management:PutAdditionalSellerNotificationRecipients",
        "payments:GetPaymentInstrument",
        "payments:CreatePaymentInstrument",
        "tax:GetTaxInterview",
        "tax:PutTaxInterview",
        "tax:GetTaxInfoReportingDocument"
    ],
    "Resource": "*"
},
{
    "Sid": "Support",
    "Effect": "Allow",
    "Action": [
        "support:CreateCase"
    ],
    "Resource": "*"
},
{
    "Sid": "ResourcePolicyManagement",
    "Effect": "Allow",
    "Action": [
        "aws-marketplace:GetResourcePolicy",
        "aws-marketplace:PutResourcePolicy",
        "aws-marketplace>DeleteResourcePolicy"
    ],
    "Resource": "arn:aws:aws-marketplace:*:*:AWSMarketplace/*"
},
{
    "Sid": "CreateServiceLinkedRole",
    "Effect": "Allow",
    "Action": "iam:CreateServiceLinkedRole",
    "Resource": "*",
    "Condition": {
        "StringEquals": {

```

```

        "iam:AWSServiceName": "resale-
authorization.marketplace.amazonaws.com"
    }
}
]
}

```

AWS managed policy: AWSMarketplaceSellerProductsFullAccess

You can attach the `AWSMarketplaceSellerProductsFullAccess` policy to your IAM identities.

This policy grants contributor permissions that allow full access to manage products and to the AWS Marketplace Management Portal, and managing the Amazon EC2 AMI used in AMI-based products.

Permissions details

```

{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "aws-marketplace:ListChangeSets",
      "aws-marketplace:DescribeChangeSet",
      "aws-marketplace:StartChangeSet",
      "aws-marketplace:CancelChangeSet",
      "aws-marketplace:ListEntities",
      "aws-marketplace:DescribeEntity",
      "aws-marketplace:ListTasks",
      "aws-marketplace:DescribeTask",
      "aws-marketplace:UpdateTask",
      "aws-marketplace:CompleteTask",
      "ec2:DescribeImages",
      "ec2:DescribeSnapshots",
      "ec2:ModifyImageAttribute",
      "ec2:ModifySnapshotAttribute"
    ],
    "Resource": "*"
  }],
  {
    "Effect": "Allow",
    "Action": [

```

```

        "iam:GetRole"
    ],
    "Resource": "arn:aws:iam::role/"
},
{
    "Effect": "Allow",
    "Action": [
        "iam:PassRole"
    ],
    "Resource": "arn:aws:iam::role/",
    "Condition": {
        "StringEquals": {
            "iam:PassedToService": "assets.marketplace.amazonaws.com"
        }
    }
},
{
    "Effect": "Allow",
    "Action": [
        "vendor-insights:GetDataSource",
        "vendor-insights:ListDataSources",
        "vendor-insights:ListSecurityProfiles",
        "vendor-insights:GetSecurityProfile",
        "vendor-insights:GetSecurityProfileSnapshot",
        "vendor-insights:ListSecurityProfileSnapshots"
    ],
    "Resource": "*"
}
{
    "Effect": "Allow",
    "Action": [
        "aws-marketplace:TagResource",
        "aws-marketplace:UntagResource",
        "aws-marketplace:ListTagsForResource"
    ],
    "Resource": "arn:aws:aws-marketplace:*:*:AWSMarketplace/*"
}
]
}

```

AWS managed policy: AWSMarketplaceSellerProductsReadOnly

You can attach the AWSMarketplaceSellerProductsReadOnly policy to your IAM identities.

This policy grants read-only permissions that allow access to view products on the AWS Marketplace Management Portal, and view the Amazon EC2 AMI used in AMI-based products.

Permissions details

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "aws-marketplace:ListChangeSets",
        "aws-marketplace:DescribeChangeSet",
        "aws-marketplace:ListEntities",
        "aws-marketplace:DescribeEntity",
        "aws-marketplace:ListTasks",
        "aws-marketplace:DescribeTask",
        "ec2:DescribeImages",
        "ec2:DescribeSnapshots"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "aws-marketplace:ListTagsForResource"
      ],
      "Resource": "arn:aws:aws-marketplace:*:*:AWSMarketplace/*"
    }
  ]
}
```

AWS managed policy: `AWSVendorInsightsVendorFullAccess`

You can attach the `AWSVendorInsightsVendorFullAccess` policy to your IAM identities.

This policy grants full access to create and manage all resources on AWS Marketplace Vendor Insights. AWS Marketplace Vendor Insights identifies assessor as the buyer and vendor is equal to the seller for the purposes of this guide. AWS Marketplace updated `AWSVendorInsightsVendorFullAccess` to add agreement search, updating profile snapshots, vendor tagging, and allows read-only access to AWS Artifact third-party reports.

Permissions details

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "aws-marketplace:DescribeEntity",
      "Resource": "arn:aws:aws-marketplace:*:*:/SaaSProduct/*"
    },
    {
      "Effect": "Allow",
      "Action": "aws-marketplace:ListEntities",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "vendor-insights:CreateDataSource",
        "vendor-insights:UpdateDataSource",
        "vendor-insights>DeleteDataSource",
        "vendor-insights:GetDataSource",
        "vendor-insights:ListDataSources",
        "vendor-insights:CreateSecurityProfile",
        "vendor-insights:ListSecurityProfiles",
        "vendor-insights:GetSecurityProfile",
        "vendor-insights:AssociateDataSource",
        "vendor-insights:DisassociateDataSource",
        "vendor-insights:UpdateSecurityProfile",
        "vendor-insights:ActivateSecurityProfile",
        "vendor-insights:DeactivateSecurityProfile",
        "vendor-insights:UpdateSecurityProfileSnapshotCreationConfiguration",
        "vendor-insights:UpdateSecurityProfileSnapshotReleaseConfiguration",
        "vendor-insights:GetSecurityProfileSnapshot",
        "vendor-insights:ListSecurityProfileSnapshots",
        "vendor-insights:TagResource",
        "vendor-insights:UntagResource",
        "vendor-insights:ListTagsForResource",
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
```



```

    "aws-marketplace:AcceptAgreementApprovalRequest",
    "aws-marketplace:RejectAgreementApprovalRequest",
    "aws-marketplace:GetAgreementApprovalRequest",
    "aws-marketplace:ListAgreementApprovalRequests"
    "aws-marketplace:CancelAgreement",
    "aws-marketplace:SearchAgreements"
  ],
  "Resource": "*",
  "Condition": {
    "ForAllValues:StringEquals": {
      "aws-marketplace:AgreementType": "VendorInsightsAgreement"
    }
  }
},
{
  "Effect": "Allow",
  "Action": [
    "artifact:GetReport",
    "artifact:GetReportMetadata",
    "artifact:GetTermForReport",
    "artifact:ListReports",
  ],
  "Resource": "arn:aws:artifact:*::report/*"
}
]
}

```

AWS managed policy: AWSVendorInsightsVendorReadOnly

You can attach the `AWSVendorInsightsVendorReadOnly` policy to your IAM identities.

This policy grants read-only access for viewing AWS Marketplace Vendor Insights profiles and related resources. AWS Marketplace Vendor Insights identifies assessor as the buyer and vendor is equal to the seller for the purposes of this guide. AWS Marketplace updated `AWSVendorInsightsVendorReadOnly` to add permissions to list tags and allows read-only access to AWS Artifact third-party reports.

Permissions details

```

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

```

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": "aws-marketplace:DescribeEntity",
    "Resource": "arn:aws:aws-marketplace:*:*:*/*SaaSProduct/*"
  },
  {
    "Effect": "Allow",
    "Action": "aws-marketplace:ListEntities",
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "vendor-insights:GetDataSource",
      "vendor-insights:ListDataSources",
      "vendor-insights:ListSecurityProfiles",
      "vendor-insights:GetSecurityProfile",
      "vendor-insights:GetSecurityProfileSnapshot",
      "vendor-insights:ListSecurityProfileSnapshots"
      "vendor-insights:ListTagsForResource"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "artifact:GetReport",
      "artifact:GetReportMetadata",
      "artifact:GetTermForReport",
      "artifact:ListReports"
    ],
    "Resource": "arn:aws:artifact:*:*:report/*"
  }
]
}

```

AWS Marketplace updates to AWS managed policies

View details about updates to AWS managed policies for AWS Marketplace since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the AWS Marketplace [Document history](#) page.

Change	Description	Date
AWSMarketplaceGetEntitlements – Updated policy	AWS Marketplace updated <code>AWSMarketplaceGetEntitlements</code> to add <code>sid</code> for the policy statement.	March 22, 2024
AWSMarketplaceSellerFullAccess – Updated policy	AWS Marketplace updated <code>AWSMarketplaceSellerFullAccess</code> to add permissions for creating service-linked roles.	March 15, 2024
AWSMarketplaceSellerFullAccess – Updated policy	AWS Marketplace updated <code>AWSMarketplaceSellerFullAccess</code> to add a permission for accessing tax information.	February 8, 2024
AWSVendorInsightsVendorFullAccess - Updated policy	AWS Marketplace updated <code>AWSVendorInsightsVendorFullAccess</code> to add permissions to update data sources.	October 18, 2023
AWSMarketplaceSellerFullAccess – Updated policy	AWS Marketplace updated <code>AWSMarketplaceSellerFullAccess</code> to add permissions for sharing entities.	June 1, 2023
AWSMarketplaceSellerFullAccess – Updated policy	AWS Marketplace updated <code>AWSMarketplaceSellerFullAccess</code> to add permissions related to account verifications, bank account verifications, case	June 1, 2023

Change	Description	Date
	management, and seller notification details.	
AWSMarketplaceSellerFullAccess – Updated policy	AWS Marketplace updated <code>AWSMarketplaceSellerFullAccess</code> to add permissions to access seller dashboards.	December 23, 2022
AWSMarketplaceSellerFullAccess , AWSMarketplaceSellerProductsFullAccess , AWSMarketplaceSellerProductsReadOnly – Update to existing policy	AWS Marketplace updated policies for the new tag-based authorization feature.	December 9, 2022
AWS Marketplace updated AWSVendorInsightsVendorFullAccess	AWS Marketplace updated <code>AWSMarketplaceSellerProductsFullAccess</code> to add agreement search, updating profile snapshots, vendor tagging, and allows read-only access to AWS Artifact third-party reports (preview).	November 30, 2022
AWS Marketplace updated AWSVendorInsightsVendorReadOnly	AWS Marketplace updated <code>AWSVendorInsightsVendorReadOnly</code> to add permissions to list tags and allows read-only access to AWS Artifact third-party reports (preview).	November 30, 2022

Change	Description	Date
AWSVendorInsightsVendorFullAccess and AWSVendorInsightsVendorReadOnly – Added new policies	AWS Marketplace added policies for the new feature AWS Marketplace Vendor Insights: <code>AWSMarketplaceSellerProductsFullAccess</code> and <code>AWSVendorInsightsVendorReadOnly</code> .	July 26, 2022
AWSMarketplaceSellerProductsFullAccess and AWSMarketplaceSellerFullAccess – Updated policies	AWS Marketplace updated policies for the new feature AWS Marketplace Vendor Insights: <code>AWSMarketplaceSellerProductsFullAccess</code> and <code>AWSMarketplaceSellerFullAccess</code> .	July 26, 2022
AWSMarketplaceSellerFullAccess and AWSMarketplaceSellerProductsFullAccess – Update to existing policies	AWS Marketplace updated the policies so that the <code>iam:PassedToService</code> condition is only applied to <code>iam:PassRole</code> .	November 22, 2021
AWSMarketplaceFullAccess – Update to an existing policy	AWS Marketplace removed a duplicate <code>ec2:DescribeAccountAttributes</code> permission from <code>AWSMarketplaceFullAccess</code> policy.	July 20, 2021
AWS Marketplace started tracking changes	AWS Marketplace started tracking changes for its AWS managed policies.	April 20, 2021

AWS Marketplace Commerce Analytics Service account permissions

Use the following IAM permissions policy to enroll in the AWS Marketplace Commerce Analytics Service.

For instructions on how to enroll, follow the [onboarding guide](#).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:ListRoles",
        "iam:CreateRole",
        "iam:CreatePolicy",
        "iam:AttachRolePolicy",
        "aws-marketplace-management:viewReports"
      ],
      "Resource": "*"
    }
  ]
}
```

Use the following IAM permissions policy to allow a user to make requests to the AWS Marketplace Commerce Analytics Service.

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

For more information about this feature, see [AWS Marketplace Commerce Analytics Service](#).

Amazon SQS permissions

As part of the SaaS product publication process, AWS Marketplace provides you an Amazon SNS topic you can use to receive notifications if a customer's subscription or entitlement status changes. You can configure one or more Amazon SQS queues to the topic so that the queues can take action on the notification. For example, if a customer adds more storage to the subscription they have to your SaaS product, the Amazon SNS topic can send a message to an Amazon SQS queue that starts a process to automatically increase the storage capacity available to that customer.

When you subscribe the Amazon Simple Queue Service (Amazon SQS) queue to the provided Amazon SNS topic, permissions are automatically added to allow the topic to publish messages to the queue. However, you still need an IAM policy for granting the AWS Marketplace Metering and Entitlement Service API user access to the queue. This can be applied to the same user if the services run with the same credentials. Create a policy with the following contents and attach it to your user or role.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "sqs:ReceiveMessage", "sqs:DeleteMessage", "sqs:GetQueueAttributes",
"sqs:GetQueueUrl"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:sqs:REGION_HERE:XXXXXXXXXXXX:NAME_HERE"
    }
  ]
}
```

Note

The Resource field is the Amazon Resource Name (ARN) of your Amazon SQS queue.

For more information on message notification and queuing for your SaaS products, see [the section called “Subscribing an SQS queue to the SNS topic”](#) and [the section called “Accessing the AWS Marketplace Metering and Entitlement Service APIs”](#).

AWS Marketplace metering and entitlement API permissions

Software as a service (SaaS) products, Amazon Machine Image (AMI) products, and container products can use the AWS Marketplace Metering Service and AWS Marketplace Entitlement Service APIs. Each type requires different AWS Identity and Access Management (IAM) permissions. For your product or products, you meter for all usage, and customers are billed by AWS based on the metering records that you provide. To enable the integration required to provide AWS Marketplace your metering records, the service account that the integration is using needs a constrained IAM policy to enable access. Attach the policy for the product type that you're sending metering information for to the user or role that you're using for the integration.

Topics

- [IAM policy for SaaS products](#)
- [IAM policy for AMI products](#)
- [IAM policy for container products](#)

IAM policy for SaaS products

In the following policy, the first permission, `aws-marketplace:ResolveCustomer`, is required for all SaaS integrations. The second permission, `aws-marketplace:BatchMeterUsage`, is needed for the AWS Marketplace Metering Service API. The third permission, `aws-marketplace:GetEntitlements`, is needed for the AWS Marketplace Entitlement Service API.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "aws-marketplace:ResolveCustomer",
        "aws-marketplace:BatchMeterUsage",
        "aws-marketplace:GetEntitlements"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```


For more information about SaaS products, see [SaaS-based products](#).

IAM policy for AMI products

Use the following IAM policy for AMI products.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        aws-marketplace:MeterUsage
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

For more information about AMI products, see [AMI-based products](#).

IAM policy for container products

Use the following IAM policy for container products.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "aws-marketplace:RegisterUsage"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

For more information about container products, see [Container-based products](#).

For more information about creating users, see [Creating a user in your AWS account](#) in the *IAM User Guide*. For more information about creating and assigning policies, see [Changing permissions for an IAM user](#).

This policy grants access to the APIs for the IAM role or user that you attach the policy to. For more information about how to enable role assumption by another account for these API calls, see [How to Best Architect Your AWS Marketplace SaaS Subscription Across Multiple AWS accounts](#) at the AWS Partner Network (APN) Blog.

Using service-linked roles for AWS Marketplace

AWS Marketplace uses AWS Identity and Access Management (IAM) [service-linked roles](#). A service-linked role is a unique type of IAM role that is linked directly to AWS Marketplace. Service-linked roles are predefined by AWS Marketplace and include all the permissions that the service requires to call other AWS services on your behalf.

Using roles for Resale Authorization for AWS Marketplace

AWS Marketplace uses AWS Identity and Access Management (IAM) [service-linked roles](#). A service-linked role is a unique type of IAM role that is linked directly to AWS Marketplace. Service-linked roles are predefined by AWS Marketplace and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up AWS Marketplace easier because you don't have to manually add the necessary permissions. AWS Marketplace defines the permissions of its service-linked roles, and unless defined otherwise, only AWS Marketplace can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete a service-linked role only after first deleting their related resources. This protects your AWS Marketplace resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see [AWS services that work with IAM](#) and look for the services that have **Yes** in the **Service-linked roles** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Service-linked role permissions for AWS Marketplace

AWS Marketplace uses the service-linked role named **AWSServiceRoleForMarketplaceResaleAuthorization**, which enables access to AWS services and resources used or managed by AWS Marketplace for Resale Authorizations.

The **AWSServiceRoleForMarketplaceResaleAuthorization** service-linked role trusts the following services to assume the role:

- `resale-authorization.marketplace.amazonaws.com`

The role permissions policy named **AWSMarketplaceResaleAuthorizationServiceRolePolicy** allows AWS Marketplace to complete the following actions on the specified resources.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "AllowResaleAuthorizationShareActionsRAMCreate",
    "Effect": "Allow",
    "Action": [
      "ram:CreateResourceShare"
    ],
    "Resource": [
      "arn:aws:ram:*:*:*"
    ],
    "Condition": {
      "StringEquals": {
        "ram:RequestedResourceType": "aws-marketplace:Entity"
      },
      "ArnLike": {
        "ram:ResourceArn": "arn:aws:aws-marketplace:*:*:AWSMarketplace/ResaleAuthorization/*"
      },
      "Null": {
        "ram:Principal": "true"
      }
    }
  },
  {
    "Sid": "AllowResaleAuthorizationShareActionsRAMAssociate",
    "Effect": "Allow",
    "Action": [
```

```

        "ram:AssociateResourceShare"
    ],
    "Resource": [
        "arn:aws:ram:*:*:*"
    ],
    "Condition": {
        "Null": {
            "ram:Principal": "false"
        },
        "StringEquals": {
            "ram:ResourceShareName": "AWSMarketplaceResaleAuthorization"
        }
    }
},
{
    "Sid": "AllowResaleAuthorizationShareActionsRAMAccept",
    "Effect": "Allow",
    "Action": [
        "ram:AcceptResourceShareInvitation"
    ],
    "Resource": [
        "arn:aws:ram:*:*:*"
    ],
    "Condition": {
        "StringEquals": {
            "ram:ResourceShareName": "AWSMarketplaceResaleAuthorization"
        }
    }
},
{
    "Sid": "AllowResaleAuthorizationShareActionsRAMGet",
    "Effect": "Allow",
    "Action": [
        "ram:GetResourceShareInvitations",
        "ram:GetResourceShareAssociations"
    ],
    "Resource": [
        "arn:aws:ram:*:*:*"
    ]
},
{
    "Sid": "AllowResaleAuthorizationShareActionsMarketplace",
    "Effect": "Allow",
    "Action": [

```

```

        "aws-marketplace:PutResourcePolicy",
        "aws-marketplace:GetResourcePolicy"
    ],
    "Resource": "arn:aws:aws-marketplace:*:*:AWSMarketplace/
ResaleAuthorization/*",
    "Condition": {
        "ForAnyValue:StringEquals": {
            "aws:CalledVia": ["ram.amazonaws.com"]
        }
    }
},
{
    "Sid": "AllowResaleAuthorizationShareActionsMarketplaceDescribe",
    "Effect": "Allow",
    "Action": [
        "aws-marketplace:DescribeEntity"
    ],
    "Resource": "arn:aws:aws-marketplace:*:*:AWSMarketplace/
ResaleAuthorization/*"
}
]
}

```

You must configure permissions to allow your users, groups, or roles to create, edit, or delete a service-linked role. For more information, see [Service-linked role permissions](#) in the *IAM User Guide*.

Creating a service-linked role for AWS Marketplace

You don't need to manually create a service-linked role. When you create a service-linked role in the AWS Marketplace Management Portal, AWS Marketplace creates the service-linked role for you.

To create a service-linked role

1. In the [AWS Marketplace Management Portal](#), sign in to the management account and choose **Settings**.
2. In the **Settings** section, select the **Service-linked roles** tab.
3. On the **Service-linked roles** page, select **Service-linked role for Resale Authorizations** or **Resale Authorizations integration**, and then choose **Create service-linked role** or **Configure integration**.

4. On the **Service-linked role for Resale Authorizations** or **Create Resale Authorizations integrations** page, review the information and confirm by choosing **Create service-linked role** or **Create integration**.

A message appears on the **Service-linked roles** page, indicating that the Resale Authorization service-linked role was successfully created.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you create a service-linked role in the AWS Marketplace Management Portal, AWS Marketplace creates the service-linked role for you again.

Editing a service-linked role for AWS Marketplace

AWS Marketplace does not allow you to edit the `AWSServiceRoleForMarketplaceResaleAuthorization` service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see [Editing a service-linked role](#) in the *IAM User Guide*.

Deleting a service-linked role for AWS Marketplace

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained.

Note

If independent software vendors (ISVs) don't have the role, AWS Resource Access Manager won't automatically share new Resale Authorizations with the targeted channel partner. If channel partners don't have the role, AWS Resource Access Manager won't automatically accept the Resale Authorization targeted to them.

To manually delete the service-linked role using IAM

Use the IAM console, the AWS CLI, or the AWS API to delete the `AWSServiceRoleForMarketplaceResaleAuthorization` service-linked role. For more information, see [Deleting a service-linked role](#) in the *IAM User Guide*.

Supported Regions for AWS Marketplace service-linked roles

AWS Marketplace supports using service-linked roles in all of the Regions where the service is available. For more information, see [AWS Regions and endpoints](#).

Logging AWS Marketplace Metering API calls with AWS CloudTrail

AWS Marketplace is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS Marketplace. CloudTrail captures API calls for AWS Marketplace as events. The calls captured include calls from the AWS Marketplace console and code calls to the AWS Marketplace API operations.

CloudTrail is enabled on your AWS account when you create the account. When supported event activity occurs in AWS Marketplace, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your account.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management user credentials.
- Whether the request was made with temporary security credentials for a role or a federated user.
- Whether the request was made by another AWS service.

AWS Marketplace supports logging the `BatchMeterUsage` operation as events in CloudTrail log files.

AWS Marketplace Metering API log file entry examples

Example: `BatchMeterUsage`

The following example shows a CloudTrail log entry that demonstrates the `BatchMeterUsage` action from the AWS Marketplace Metering Service. When the seller [sends metering records to report their customers' usage](#) for a software as a service (SaaS) product listed on AWS Marketplace, this CloudTrail log entry is logged in the seller's AWS account.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:user/Alice",
    "accountId": "123456789012",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2018-04-19T16:32:51Z",
  "eventSource": "metering-marketplace.amazonaws.com",
  "eventName": "BatchMeterUsage",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.0.2/24",
  "userAgent": "Coral/Netty14",
  "requestParameters": {
    "usageRecords": [
      {
        "dimension": "Dimension1",
        "timestamp": "Apr 19, 2018 4:32:50 PM",
        "customerIdentifier": "customer1",
        "quantity": 1
      }
    ],
    "productCode": "EXAMPLE_proCode"
  },
  "responseElements": {
    "results": [
      {
        "usageRecord": {
          "dimension": "Dimension1",
          "timestamp": "Apr 19, 2018 4:32:50 PM",
          "customerIdentifier": "customer1",
          "quantity": 1
        },
        "meteringRecordId": "bEXAMPLE-98f0-4e90-8bd2-bf0EXAMPLE1e",
        "status": "Success"
      }
    ],
    "unprocessedRecords": [ ]
  },
  "requestID": "dEXAMPLE-251d-11e7-8d11-1f3EXAMPLE8b",
}
```



```

    "eventID": "cEXAMPLE-e6c2-465d-b47f-150EXAMPLE97",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
  }
]
}

```

Example: RegisterUsage for containers

The following example shows a CloudTrail log entry that demonstrates the RegisterUsage action from the AWS Marketplace Metering Service. When an hourly-priced container product from AWS Marketplace is deployed in the buyer's AWS account, the software in the container calls RegisterUsage within the buyer's AWS account to initiate the hourly metering for that Amazon Elastic Container Service (Amazon ECS) task or Amazon Elastic Kubernetes Service (Amazon EKS) pod. This CloudTrail log entry is logged in the buyer's AWS account.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EX_PRINCIPAL_ID:botocore-session-1111111111",
    "arn": "arn:aws:sts::123456789012:assumed-role/Alice/botocore-session-1111111111",
    "accountId": "123456789012",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:role/Alice",
        "accountId": "123456789012",
        "userName": "Alice"
      },
      "webIdFederationData": {
        "federatedProvider": "arn:aws:iam::123456789012:oidc-provider/oidc.eks.us-east-1.amazonaws.com/id/EXAMPLEFA1C58F08CDB049167EXAMPLE",
        "attributes": {}
      },
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-07-23T02:19:34Z"
      }
    }
  }
}

```

```

    }
  }
},
"eventTime": "2020-07-23T02:19:46Z",
"eventSource": "metering-marketplace.amazonaws.com",
"eventName": "RegisterUsage",
"awsRegion": "us-east-1",
"sourceIPAddress": "1.2.3.4",
"userAgent": "aws-cli/1.18.103 Python/3.8.2 Linux/4.14.181-142.260.amzn2.x86_64
botocore/1.17.26",
"requestParameters": {
  "productCode": "EXAMPLE_proCode",
  "publicKeyVersion": 1
},
"responseElements": {
  "signature": "eyJhbGciOiJIUzI1Ni..."
},
"requestID": "dEXAMPLE-251d-11e7-8d11-1f3EXAMPLE8b",
"eventID": "cEXAMPLE-e6c2-465d-b47f-150EXAMPLE97",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

Example: MeterUsage for containers on Amazon EKS

The following example shows a CloudTrail log entry that demonstrates the `MeterUsage` action from the AWS Marketplace Metering Service for containers on Amazon EKS. When a container product with [custom metering](#) from AWS Marketplace is deployed in the buyer's AWS account, the software in the container calls `MeterUsage` within the buyer's AWS account to report each hour. This CloudTrail log entry is logged in the buyer's AWS account.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EX_PRINCIPAL_ID:botocore-session-1111111111",
    "arn": "arn:aws:sts::123456789012:assumed-role/Alice/botocore-
session-1111111111",
    "accountId": "123456789012",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",

```

```

        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:role/Alice",
        "accountId": "123456789012",
        "userName": "Alice"
    },
    "webIdFederationData": {
        "federatedProvider": "arn:aws:iam::123456789012:oidc-provider/
oidc.eks.us-east-1.amazonaws.com/id/EXAMPLEFA1C58F08CDB049167EXAMPLE",
        "attributes": {}
    },
    "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-07-23T01:03:26Z"
    }
}
},
"eventTime": "2020-07-23T01:38:13Z",
"eventSource": "metering-marketplace.amazonaws.com",
"eventName": "MeterUsage",
"awsRegion": "us-east-1",
"sourceIPAddress": "1.2.3.4",
"userAgent": "aws-cli/1.18.103 Python/3.8.2 Linux/4.14.181-142.260.amzn2.x86_64
botocore/1.17.26",
"requestParameters": {
    "timestamp": "Jul 23, 2020 1:35:44 AM",
    "usageQuantity": 1,
    "usageDimension": "Dimension1",
    "productCode": "EXAMPLE_proCode"
},
"responseElements": {
    "meteringRecordId": "bEXAMPLE-98f0-4e90-8bd2-bf0EXAMPLE1e"
},
"requestID": "dEXAMPLE-251d-11e7-8d11-1f3EXAMPLE8b",
"eventID": "cEXAMPLE-e6c2-465d-b47f-150EXAMPLE97",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

Example: MeterUsage on AMIs

The following example shows a CloudTrail log entry that demonstrates the MeterUsage action from the AWS Marketplace Metering Service for Amazon Machine Images (AMIs). When an AMI product with custom metering from AWS Marketplace is deployed in the buyer's AWS account, the

software from the AMI calls MeterUsage within the buyer's AWS account to report usage each hour. This CloudTrail log entry is logged in the buyer's AWS account.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EX_PRINCIPAL_ID:i-exampled859aa775c",
    "arn": "arn:aws:sts::123456789012:assumed-role/Alice/i-exampled859aa775c",
    "accountId": "123456789012",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:role/Alice",
        "accountId": "123456789012",
        "userName": "Alice"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-07-10T23:05:20Z"
      },
      "ec2RoleDelivery": "1.0"
    }
  },
  "eventTime": "2020-07-10T23:06:42Z",
  "eventSource": "metering-marketplace.amazonaws.com",
  "eventName": "MeterUsage",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "1.2.3.4",
  "userAgent": "aws-cli/1.16.102 Python/2.7.16 Linux/4.14.133-113.112.amzn2.x86_64
  botocore/1.12.92",
  "requestParameters": {
    "productCode": "EXAMPLE_proCode",
    "timestamp": "Jul 10, 2020 11:06:41 PM",
    "usageDimension": "Dimension1",
    "usageQuantity": 1,
    "dryRun": false
  },
  "responseElements": {
    "meteringRecordId": "bEXAMPLE-98f0-4e90-8bd2-bf0EXAMPLE1e"
  }
}
```

```
  },  
  "requestID": "dEXAMPLE-251d-11e7-8d11-1f3EXAMPLE8b",  
  "eventID": "cEXAMPLE-e6c2-465d-b47f-150EXAMPLE97",  
  "eventType": "AwsApiCall",  
  "recipientAccountId": "123456789012"  
}
```

Document history

The following table describes the documentation for this release of the *AWS Marketplace Seller Guide*.

For notification about updates to this documentation, you can subscribe to the RSS feed.

Change	Description	Date
Updated the Know Your Customer (KYC) process	Added additional step to the KYC process for sellers.	May 21, 2024
Updated private offer experience for AWS Marketplace sellers	Added content for an enhanced experience for creating and managing private offers.	May 20, 2024
Updated Requirements for Amazon EKS add-on products	Updated "Preparing your container product as an AWS Marketplace add-on" section and added "Add-on configuration requirements and best practices for add-on providers".	May 8, 2024
Updated permissions for AWS Marketplace sellers	Updated permissions examples to fix syntax errors.	April 2, 2024
Updated SaaS contract pricing	Updated content related to reporting overages for SaaS contract with pay-as-go pricing models.	April 2, 2024
New demo and private offer options on AWS Marketplace	AWS Marketplace now supports demo and private offer request options on product detail pages for select sellers.	April 1, 2024

New dashboard for taxation	AWS Marketplace now supports a taxation dashboard that provides visualization and fine-grained data on US and international tax for transactions in AWS Marketplace.	March 29, 2024
Updated AWSMarketplaceGetEntitlements	Added <code>sid</code> to the <code>AWSMarketplaceGetEntitlements</code> managed policy.	March 22, 2024
Updated professional services procedures	Updated the editing product visibility and removing a professional services product procedures.	March 19, 2024
Updated AMI access policies	Updated the section to clarify Linux-specific and Unix-like AMI access policies.	March 19, 2024
Removed deprecated permissions	Removed <code>aws-marketplace-management:viewSettings</code> , <code>aws-marketplace-management:uploadFiles</code> , and the <i>Using fine-grained permissions</i> section.	March 19, 2024
Added search engine optimization information for AWS Marketplace	Added content related to search engine optimization for AWS Marketplace.	March 19, 2024
Updated managed policy for sellers in AWS Marketplace	Updated <code>AWSMarketplaceSellerFullAccess</code> to add permissions related to creating service-linked roles.	March 15, 2024

New service-linked role for products in AWS Marketplace	AWS Marketplace now provides a service-linked role that enables access to AWS services and resources used or managed by AWS Marketplace for Resale Authorizations.	March 15, 2024
Updated managed policy for sellers in AWS Marketplace	Updated AWSMarketplaceSellerFullAccess to add permissions related to accessing tax information.	February 8, 2024
Amazon EventBridge events for security reports	AWS Marketplace now supports Amazon EventBridge events, formerly called Amazon CloudWatch Events, when a security vulnerability report is available for a seller's products.	January 31, 2024
Support for Amazon EKS add-ons	Added content and procedures related to publishing to Amazon EKS add-ons from AWS Marketplace container-based product.	January 29, 2024
Added support for professional service sellers to resell products	Sellers can now create resell opportunities for channel partners as independent software vendors (ISVs).	January 18, 2024
General availability for future dated agreements in AWS Marketplace	All ISVs and AWS Marketplace Channel Partners can specify a future start date as part of publishing a private offer with upfront pricing.	January 16, 2024

New dashboard for usage	AWS Marketplace now supports a usage dashboard that provides a visualization and fine-grained data for customers using SaaS and server usage-based products.	January 10, 2024
New Quick Launch deployment option for sellers	Added content and procedures related to the new Quick Launch deployment option for software as a service (SaaS) products.	November 29, 2023
Flexible payment schedules are available for private offers	Flexible Payment Schedules (FPS) for private offers are now available to all customers in the AWS Marketplace.	November 17, 2023
Added self-service content for container products	Added content and procedures related to self-service actions for container products.	November 3, 2023
New dashboard for agreements and renewals	AWS Marketplace now provides an agreements and renewals dashboard for sellers.	October 31, 2023
Amazon EventBridge events for change sets	AWS Marketplace now supports Amazon EventBridge events, formerly called Amazon CloudWatch Events, when a change set completes with a status of succeeded, failed, or cancelled.	October 31, 2023

Updated managed policy for sellers in AWS Marketplace Vendor Insights	AWS Marketplace updated <code>AWSVendorInsightsVendorFullAccess</code> to add permissions to update data sources.	October 18, 2023
Ending support for AWS Marketplace for Desktop Applications (AMDA)	AWS Marketplace ended support for AMDA on October 2, 2023. All content and procedures related to AMDA were removed from the guide.	October 2, 2023
Added self-service content for SaaS products	Added content and procedures related to self-service actions for software as a service (SaaS) products.	September 12, 2023
Daily disbursements are now available for AWS Marketplace sellers	Sellers in AWS Marketplace Management Portal now have the option to receive disbursements daily or monthly.	September 7, 2023
Support for Amazon EventBridge	AWS Marketplace now supports EventBridge events when sellers receive new offers.	September 6, 2023
New self-service onboarding experience for AWS Marketplace Vendor Insights	AWS Marketplace Vendor Insights now supports a self-service onboarding experience.	August 17, 2023
Updated managed policy for sellers in AWS Marketplace	Updated <code>AWSMarketplaceSellerFullAccess</code> to add permissions related to sharing entities.	June 1, 2023

Updated managed policy for sellers in AWS Marketplace	Updated AWSMarketplaceSellerFullAccess to add permissions related to account verifications, bank account verifications, case management, and seller notification details.	June 1, 2023
Added content about fine-grained access for AWS Marketplace	Added overview information, permissions, and procedures for fine-grained access in the AWS Marketplace Management Portal.	June 1, 2023
Added procedures for AMI products	Added self-service procedures for AMI products.	May 12, 2023
Ending support for Amazon Tax Exemption Program and US Tax Calculation Service on AWS Marketplace	Amazon Tax Exemption Program and US Tax Calculation Service were removed from the guide because support for this service ended effective March 6, 2023.	March 6, 2023
Added procedures for container products	Added new procedures for making changes to container product settings.	February 13, 2023

Ending support for AWS Marketplace Product Support Connection	AWS Marketplace Product Support Connection and sharing customer contact details using the Commerce Analytics Service are no longer supported as of November 30, 2022. AWS Marketplace removed AWS Marketplace Product Support Connection content from the guide.	January 27, 2023
Consulting partner name change	AWS Marketplace now refers to <i>consulting partners</i> as <i>channel partners</i> . The guide was updated to reflect the name change only.	January 26, 2023
Customer service team name change	The Managed Catalog Operations (MCO) team changed their name to AWS Marketplace Seller Operations team. The guide was updated to reflect the name change only.	January 24, 2023
Private offers page	Authenticated buyers can now view the AWS Marketplace private offers extended to their AWS account on the Private offers page.	January 19, 2023
Added topic about SaaS product settings	Added a new topic with procedures for making changes to SaaS product settings.	January 6, 2023

Updated managed policy for sellers in AWS Marketplace	AWS Marketplace updated <code>AWSMarketplaceSellerFullAccess</code> to add permissions to access seller dashboards.	December 23, 2022
Updated email notifications for sellers	Sellers are now notified when a private offer is published.	December 22, 2022
SaaS free trials for subscriptions are now available to sellers on AWS Marketplace	Sellers can now create free trials for subscription products.	December 16, 2022
Update seller experience for AMI self-service listing (version 2) on AWS Marketplace	Sellers on AWS Marketplace can now create a single-AMI product self-service listing. Sellers can make updates on their own without additional wait times for processing.	December 14, 2022
Updated three policies for tag-based authorization	Updated three policies (<code>AWSMarketplaceSellerFullAccess</code> , <code>AWSMarketplaceSellerProductsFullAccess</code> , and <code>AWSMarketplaceSellerProductsReadOnly</code>) for AWS Marketplace tag-based authorization feature.	December 9, 2022

[Updated policies for sellers in AWS Marketplace Vendor Insights](#)

Updated managed policies `AWSVendorInsightsAssessorFullAccess` and `AWSVendorInsightsVendorReadOnly` for AWS Marketplace Vendor Insights sellers.

November 30, 2022

[Controlling access for sellers in AWS Marketplace Vendor Insights](#)

Added a new topic for AWS Marketplace Vendor Insights to describe actions and permissions available to sellers.

November 30, 2022

[Updated four managed policies for AWS Marketplace Vendor Insights](#)

Updated `AWSVendorInsightsAssessorFullAccess`, `AWSVendorInsightsAssessorReadOnly`, `AWSVendorInsightsVendorFullAccess`, and `AWSVendorInsightsVendorReadOnly` managed policies for AWS Marketplace Vendor Insights.

November 28, 2022

[Sellers can publish add-on products in Amazon EKS](#)

The integration of AWS Marketplace and Amazon EKS enables sellers to present their products in the Amazon EKS console for buyers.

November 28, 2022

[Updated AWS Marketplace Vendor Insights setup](#)

Updated the setup procedure for AWS Marketplace Vendor Insights.

November 18, 2022

Updated two policies for AWS Marketplace Vendor Insights	Updated two policies <code>AWSMarketplaceSellerProductsFullAccess</code> and <code>AWSMarketplaceSellerFullAccess</code> for AWS Marketplace Vendor Insights.	July 26, 2022
Added two policies for AWS Marketplace Vendor Insights a feature offering software risk assessment.	Added two policies <code>AWSVendorInsightsVendorFullAccess</code> and <code>AWSVendorInsightsVendorReadOnly</code> for AWS Marketplace Vendor Insights a feature offering software risk assessment.	July 26, 2022
AWS Marketplace Vendor Insights is a new feature added to AWS Marketplace	AWS Marketplace Vendor Insights is a feature offering software risk assessment. AWS Marketplace Vendor Insights Vendor Insights is a feature offering software risk assessment.	July 26, 2022
AWS Marketplace Commerce Analytics Service permissions update	The AWS Marketplace Commerce Analytics Service has additional IAM permissions.	July 21, 2022
Seller Delivery Data Feeds Service section added	Documentation-only update to add the Seller Delivery Data Feeds Service section and reorganize the sections related to data feeds.	June 15, 2022

<u>Supplementary reports section added</u>	Added a new section for supplementary reports that AWS Marketplace provides for recent feature launches.	June 14, 2022
<u>SaaS free trials for contracts are now available to sellers on AWS Marketplace</u>	Sellers can now create free trials without additional development work by defining the free trial length, dimension(s) available in the trial period, and the amount of free usage capacity customers receive.	May 31, 2022
<u>Email notifications added to buyer and seller transactions</u>	New feature enabling email notifications to buyers and sellers verifying offers and agreements made in AWS Marketplace.	May 23, 2022
<u>Examples added to creating your machine learning product listing</u>	Documentation-only update to the machine learning section to include examples to show a comparison of the seller's view and the buyer's view when creating a machine learning product.	April 22, 2022
<u>Updates to the machine learning section</u>	Documentation-only updates have been made to the machine learning section to clarify procedures.	April 15, 2022
<u>Israel is now an eligible jurisdiction</u>	Residents in Israel are now eligible to become sellers on AWS Marketplace.	April 13, 2022

Updated for new items in offer data feed	Manufacturers will now receive offer information for offers created by their channel partner. This includes offers where the concerning account is the seller of record and the manufacturer for the offer as well.	March 29, 2022
Reseller opportunity notifications	Sellers now have the ability to receive notifications for reseller opportunities.	March 28, 2022
Added a video to professional services products	Updated the professional services products page with a video containing details on how to manage service products.	February 24, 2022
New topic about deploying a serverless SaaS integration solution	New information added for integrating serverless SaaS deployment, including a link to AWS Quick Start for a reference on deployment steps.	February 15, 2022
Minimal updates to Container-based requirements and AMI sections	Minimal updates to policies in Container-based requirements and removed incorrect information for AMI pricing contracts	February 14, 2022
Container versioning update	Documentation-only update to clarify how to push container images and other artifacts to repositories.	February 10, 2022

Update to ResolveCustomer code example for SaaS products	The ResolveCustomer code example for SaaS products has been updated to include CustomerAWSAccountID .	February 3, 2022
Added documentation for integrating AWS License Manager with AWS Marketplace for Containers Anywhere products	Documentation-only update to add detailed guidance on adding contract-pricing to your AWS Marketplace for Containers Anywhere products by integrating with License Manager.	February 1, 2022
SaaS SNS notifications update	Documentation-only update to clarify SaaS notification messages.	January 25, 2022
Ability for sellers to transact with EMEA-based buyers through Amazon Web Services EMEA SARL	Eligible AWS Marketplace sellers can now transact with customers whose AWS accounts are based in countries and territories in Europe, the Middle East, and Africa (EMEA) through Amazon Web Services EMEA SARL.	January 7, 2022

[Added documentation for creating delivery options for container-based products with a Helm chart delivery method](#)

Sellers can now provide delivery options with a Helm chart delivery method. Buyers can use these delivery options to launch a container-based application by installing a seller-provided Helm chart in their launch environment. When providing a Helm chart delivery method, sellers can enable QuickLaunch for buyers. QuickLaunch is a feature that buyers can use to use AWS CloudFormation to quickly create a new Amazon EKS cluster and launch a container-based application on it.

November 29, 2021

[Update to existing policies](#)

Security policies for AWS Marketplace sellers have been updated.

November 22, 2021

[Contract pricing for AMI and Container-based products](#)

Independent software vendors (ISVs) can now list a new AMI-based product or a Container-based product and offer upfront contract pricing for buyers.

November 17, 2021

[Vendor metered tagging](#)

Documentation-only update for vendor metered tagging, including code examples.

November 11, 2021

[Amazon Simple Notification Service for AMI or Container products](#)

Independent software vendors (ISVs) can receive notifications when customers subscribe to or unsubscribe from AMI or Container products through the Amazon Simple Notification Service.

November 10, 2021

[New seller permissions](#)

AWS Marketplace added new permissions to access the **Offers** and **Partners** tabs in the AWS Marketplace Management Portal.

November 9, 2021

[Ability to deploy endpoints configured for Asynchronous Inference for machine learning products](#)

For machine learning software that expects a higher payload inference than the maximum, or requires processing times that exceed the maximum processing time per invocation, buyers have the ability to deploy endpoints configured for Amazon SageMaker Asynchronous Inference.

November 8, 2021

[Refund policy and approvals](#)

Documentation-only update to clarify the refund policy and move all refund information to one central location in the *AWS Marketplace Seller Guide*.

August 20, 2021

Select or upload EULA for consulting partner private offers	Independent software vendors can now select or upload an End User License Agreement (EULA) when creating resale opportunities for consulting partners.	August 17, 2021
Custom product dimensions for SaaS contract products	Independent software vendors (ISVs) can now customize SaaS contract product dimensions when creating resale opportunities for consulting partners.	August 17, 2021
AWS Marketplace Field Demonstration Program	Documentation-only update to clarify the requirements for AWS Data Exchange dataset products for the AWS Marketplace Field Demonstration Program.	August 3, 2021
SaaS product guidelines update	Product guidelines for SaaS products have been updated.	July 29, 2021
Container-based product requirements update	Container-based product requirements have been updated.	July 29, 2021
AMI security policy update	Security policies for AMI products have been updated.	July 29, 2021
More eligible jurisdictions	The following are now eligible to become sellers on AWS Marketplace: Hong Kong SAR and Qatar.	June 23, 2021

Data feeds overview	Documentation-only update to give an overview of the structure of the data feeds available to sellers.	June 23, 2021
Updated the machine learning chapter	Documentation-only update to the information about creating and maintaining machine learning products.	May 27, 2021
Self-service updating for container products	Sellers now have a simpler and faster way to update their container-based products through the AWS Marketplace Management Portal.	December 17, 2020
Professional services	Sellers can now offer professional services to AWS Marketplace buyers. Added the professional services section to the documentation.	December 3, 2020
Self-service updating for AMI products	Sellers now have a simpler and faster way to update their Amazon Machine Image (AMI) based products through the AWS Marketplace Management Portal.	November 23, 2020
More eligible jurisdictions	The following are now eligible to become sellers on AWS Marketplace: Bahrain, Norway, Switzerland, and the United Arab Emirates (UAE).	June 17, 2020

You can offer upgrades and renewals on accepted private offers	For SaaS contract and SaaS contracts with consumption products, you can offer upgrades and renewals using private offers on previously-accepted private offers.	May 28, 2020
More information is available in data feeds	More information from reports is broken down into smaller data feeds to simplify finding and analyzing data.	May 21, 2020
Standardized license terms are now available	You can offer standardized license terms in place of custom EULAs to simplify the contracting process.	April 28, 2020
Australia and New Zealand are eligible jurisdictions	The following are now eligible to become sellers on AWS Marketplace: (i) Permanent residents and citizens of Australia (AU) New Zealand (NZ) or (ii) business entities organized or incorporated in one of those areas.	April 2, 2020
Container products now support custom metering and pricing enhancements	If you want to define your own pricing units and meter that usage to us for billing, integrate with the AWS Marketplace Metering Service's meterUsage action.	November 14, 2019
AWS Marketplace supports data products through AWS Data Exchange	You can now provide data products in AWS Marketplace.	November 13, 2019

Introducing the AWS Marketplace Catalog API service	The AWS Marketplace Catalog API service provides an API interface for approved sellers to programmatically manage their products.	November 12, 2019
AWS Marketplace supports paid hourly containers	AWS Marketplace now supports paid hourly containers running on Amazon Elastic Kubernetes Service (Amazon EKS).	September 25, 2019
Updated AMI product functionality	You can now deploy AMIs and Lambda functions together using AWS CloudFormation.	September 11, 2019
Added Security section	Consolidated security content under a new Security section.	May 7, 2019
Updated AMI security policies	Updated the security policies for AMI products	April 11, 2019
Added versioning information to Machine Learning Products section	Added content describing product version capability for machine learning products.	March 21, 2019
Added Machine Learning Products section	Added content for publishing machine learning products	November 28, 2018
Added Container-Based Products section	Added content for publishing container-based products.	November 27, 2018
Updated link for submitting seller help request	Changed email address to webform address.	October 22, 2018
Added SaaS contracts with consumption content	Restructured SaaS content and added content to support release of SaaS contracts with consumption features.	October 18, 2018

Added content about flexible payment schedule for private offers	Added content to support release of flexible payment scheduler for private offers content.	October 15, 2018
Updated IAM permissions content	Added content to support of new IAM permission for AMMP read only access.	October 9, 2018
Added content about consulting partner private offers	Added content to support Consulting Partner Private Offers feature release.	October 9, 2018
Added content about private image builds	Added content to support release of Private Image Build for AMIs feature.	August 13, 2018
Added search engine optimization guidance for sellers.	Added guidance for sellers who want to optimize their product for search.	July 3, 2018
Updated link to AWS Marketplace logos	Updated link to point to new AWS Marketplace logos.	June 12, 2018
Added seller guides	Converted all PDF seller guides to online content.	May 9, 2018

AWS Glossary

For the latest AWS terminology, see the [AWS glossary](#) in the *AWS Glossary Reference*.