AWS Application Discovery Service

User Guide

AWS Application Discovery Service: User Guide

Table of Contents

What Is AWS Application Discovery Service?

AWS Application Discovery Service helps you plan application migration projects by automatically identifying servers, virtual machines (VMs), software, and software dependencies running in your on-premises data centers. Application Discovery Service also collects application performance data, which can help you assess the outcome of your migration. The data collected by Application Discovery Service is securely retained in an Amazon-hosted and managed database in the cloud. You can export the data as a CSV or XML file into your preferred visualization tool or cloud-migration solution to plan your migration. For more information, see the Application Discovery Service FAQ.

Application Discovery Service offers two modes of operation.

- **Agentless discovery** mode is recommended for environments that use VMware vCenter Server. This mode doesn't require you to install an agent on each host. Agentless discovery gathers server information regardless of the operating systems, which minimizes the time required for initial on-premises infrastructure assessment. Agentless discovery doesn't collect information about software and software dependencies. It also doesn't work in non-VMware environments. We recommend that you use agent-based discovery for non-VMware environments and if you want to collect information about software and software dependencies. You can also run agent-based and agentless discovery simultaneously. Use agentless discovery to quickly complete the initial infrastructure assessment and then install agents on select hosts to gather information about software and software dependencies.
- **Agent-based discovery** mode collects a richer set of data than agentless discovery by using Amazon software, the AWS Application Discovery Agent, which you install on one or more hosts in your data center. The agent captures infrastructure and application information, including an inventory of installed software applications, system and process performance, resource utilization, and network dependencies between workloads. The information collected by agents is secured at rest and in transit to the Application Discovery Service database in the cloud.

Application Discovery Service integrates with application discovery solutions from AWS Partner Network (APN) partners. Third-party application discovery tools can query the Application Discovery Service and write to the Application Discovery Service database using a public API. You can then import the data into either a visualization tool or cloud-migration solution.

> **Important**
> Application Discovery Service doesn't gather sensitive information. All data is handled according to the AWS Privacy Policy. You can operate Application Discovery Service using offline mode to inspect collected data before it is shared with the service.

For more specific information about the data Application Discovery Service collects, see AWS Application Discovery Service Components (p. 4).

Request Access from AWS

To gain access to the Application Discovery Service, AWS partners and customers must submit a request to be whitelisted. When you complete the request, we will send you information about how to get started with the service.

Prerequisites

Agentless discovery is compatible only with VMware vCenter Server.

Agent-based discovery is compatible with the following operating systems.

Linux

- Ubuntu 14
- Amazon Linux 2012.03 or 2015.03
- Centos 6 or 7
- Redhat 6 or 7

Microsoft Windows

- Windows Server 2012 R2
- Windows Server 2012
- Windows Server 2008 R2

> **Note**
> There is a known issue on servers running Microsoft Windows Server 2008 R2 with Service Pack 1 where agents do not send data to Application Discovery Service. If you plan to run agents on this operating system, install the following hotfix, Availability of SHA-2 Code Signing Support for Windows 7 and Windows Server 2008 R2.

Firewall Configuration

The AWS Application Discovery Agent requires outbound access to arsenal.us-west-2.amazonaws.com:443. It does not require any inbound ports to be open. Agents also work with transparent web proxies.

Limitations

Application Discovery Service has the following limitations for agentless and agent-based discovery.

Agentless discovery

The service limits you to 10 GB of data per day. If you reach this limit, the service won't process any more data for that day. If you frequently reach this limit, contact AWS Support about extending the limit.

Agent-based discovery

Agent-based discovery currently has the following limitations.

- The AWS Application Discovery Agent does not support Linux environments with non-standard Ethernet naming conventions. The system requires an eth0 adapter.
- The service enforces the following maximum limits:
  - 250 active agents (agents that are collecting and sending data to Application Discovery Service in the cloud).
  - 10,000 inactive agents (agents that are responsive but not collecting data).
  - 10 GB of data per day (collected by all agents associated with a given AWS account).
  - 90 days of data storage (after which the data is purged).

Related Services

You can use the AWS VM Import/Export tools to import VM images from your local environment into AWS and convert them into ready-to-use Amazon Elastic Compute Cloud (EC2) Amazon Machine Images (AMIs) or instances. For more information, see Importing and Exporting Instances.

You have the flexibility to choose the discovery and migration tools that you need by integrating with AWS Partner tools using the Application Discovery Service API. For more information about the Application Discovery Service API, see the *Application Discovery Service API Reference*.

Accessing the Application Discovery Service

Application Discovery Service supports the following command line and programmatic access options:

AWS Command Line Interface (CLI)
The AWS CLI provides commands for a broad set of AWS products. It is supported on Windows, Mac, and Linux. For more information, see AWS Command Line Interface User Guide.

Application Discovery Service API
You can use the Application Discovery Service API to manage software agents in your data center, query discovered assets, categorize discovered assets using tags, and export data. Application Discovery Service uses JavaScript Object Notation format (JSON) to send and receive formatted data. JSON presents data in a hierarchy so that both data values and data structure are conveyed simultaneously. For more information about the Application Discovery Service API, see the *Application Discovery Service API Reference*.

AWS SDKs and Tools
If you prefer to build applications using language-specific APIs instead of submitting a request over HTTP or HTTPS, AWS provides libraries, sample code, tutorials, and other resources for software developers. These libraries provide basic functions that automate tasks such as cryptographically signing your requests, retrying requests, and handling error responses, making it is easier for you to get started. For more information, see AWS SDKs and Tools.

How to Get Started

- AWS Application Discovery Service Components (p. 4)
- Setting Up AWS Application Discovery Service (p. 11)
- AWS Application Discovery Service Walkthrough (p. 25)

AWS Application Discovery Service Components

AWS Application Discovery Service uses a combination of Amazon software agents and cloud-based services to identify, map, and store an inventory of the assets in your computing environment.

Contents

The Arsenal Service

Arsenal is an agent service managed and hosted by Amazon that sends data from AWS Application Discovery Agents and AWS Agentless Discovery Connector to the Application Discovery Service in the cloud. As you set up and configure Application Discovery Service, you will see the word *arsenal* in URLs and AWS Identity and Access Management (IAM) policies.

Agentless Discovery Components

Agentless discovery uses the AWS Agentless Discovery Connector to communicate with the Arsenal service. You install the Connector as a virtual machine (VM) in your VMware vCenter Server environment using an Open Virtualization Archive (OVA) file. When you start the Connector, it registers with Arsenal, and frequently pings Arsenal for configuration information. When you send a command for the Connector to start collecting data, it connects to VMWare vCenter Server and collects information about all the VMs and hosts managed by this specific vCenter. The collected data is sent to Arsenal using Secure Sockets Layer (SSL) encryption. The Connector is configured to automatically upgrade when new versions of the Connector become available. You can change this configuration setting at any time.

Data Collected During Agentless Discovery

Agentless discovery does not collect information about your applications. It only collects information about your VMware vCenter Server hosts and VMs, including performance data about those hosts and VMs. The information collected by agentless discovery is shown in the following tables. Items in **bold** are only captured if VMware vCenter Server tools are installed. Agentless discovery attempts to collect all of the following data, but in some situations, vCenter might not have the data, as noted in the following tables.

Inventory Data about VMs in vCenter

Data	Guaranteed or If Available
Timestamp	Guaranteed
OSType	If available
SystemRelease	If available
MoRefID (Unique vCenter Managed Object Reference ID)	Guaranteed
instanceUuid (Unique ID for a Virtual Machine. Not for host system)	If available
FolderPath (VM folder path in vCenter)	Guaranteed
Name (Name of the vCenter VM or host)	Guaranteed
Hostname	If available
Hypervisor	Guaranteed
Manufacturer	Guaranteed
ToolsStatus (VMware tools status)	If available
HostSystem (MoRefId of the host system of a VM)	Guaranteed for VM
Datacenter (MoRefID of the data center where the system is located)	Guaranteed
Type (Host or VM)	Guaranteed
vCenterId (Unique vCenter ID)	Guaranteed

Data	Guaranteed or If Available
smBiosId	If available
MacAddress	Guaranteed
IpAddress	If available
Network - List (A VMware object representation of a network)	If available
macAddress (For the network)	Guaranteed if network exists
portGroupName (For the network)	If available
portGroupId (For the network)	If available
virtualSwitchName	If available
Name (Network name specified by user)	If available
CPUType (vCPU for VM, actual model for host)	If available

Performance Data for VMs in vCenter

Data	Guaranteed or If Available
Timestamp	Guaranteed
MoRefID (Managed Object Reference ID of the system producing the metrics)	Guaranteed
Type (Host or VM)	Guaranteed
vCenterId (Unique ID of the vCenter)	Guaranteed
smBiosId	If available
PowerState	Guaranteed
MemorySize (Memory size of VM/host)	If available
MemoryReservation	If available

Data	Guaranteed or If Available
(Reservation set for a VM)	
ActiveRAM (Average RAM over polling period)	If available
MaxActiveRam (Max RAM over polling period)	If available
NetworkCards	If available
Name (Name associated with metrics collected)	If available
BytesReadPerSecond (Average over polling period)	If available
BytesWrittenPerSecond (Average over polling period)	If available
TotalUsage (Average transmitted/received over polling period)	If available
MaxTotalUsage (Max transmitted/received over polling period)	If available
Disks	If available
DeviceID (Name associated with metrics collected. For virtual device it will be the scsi id)	If available
Name	If available
Capacity	If available
scsi (For mapping performance metrics to a virtual disk)	If available
BytesReadPerSecond (Average over polling period)	If available
BytesWrittenPerSecond (Average over polling period)	If available
ReadOpsPerSecond (Average over polling period)	If available

Data	Guaranteed or If Available
WriteOpsPerSecond (Average over polling period)	If available
Cpus	If available
Name (Name associated with metrics collected)	If available
UsagePct	If available
UsageMHz (Average over polling period)	If available
MaxUsageMHz (Max over polling period)	If available
numCores	If available
speedMHz	If available
reservationMHz (reservation set for a VM)	If available

The AWS Application Discovery Agent

The AWS Application Discovery Agent is Amazon software that you install on on-premises servers and VMs that you target for discovery and migration. Agents run on Linux and Windows and collect server configuration and activity information about your applications and infrastructure. You can also install the agent on Amazon EC2 instances. When you start an agent, it registers with Arsenal and frequently pings the service for configuration information. When you send a command that tells an agent to start collecting data, it collects an extensive amount of data for the host or VM where it resides, including TCP and UDP connections to other hosts or VMs, which can help you map your IT assets without having to install an agent on every host or VM. Agents are configured to automatically upgrade when new versions become available. You can change this configuration setting at any time.

Agents collect information in the following categories and send it to Application Discovery Service using Secure Sockets Layer (SSL) encryption:

- User information (user name, home directory, etc)
- Group information (name)
- List of installed packages
- List of kernel modules
- All create and stop process events
- DNS queries
- NIC information
- TCP/UDP process listening ports
- TCPV4/V6 connections
- Operating system information
- System performance
- Process performance

After you install agents, you manage them using the Application Discovery Service API. The API includes actions to start and stop agents. You can also retrieve information about agents, including the host name where agents reside, their health, and the version number of each agent. For more information, see the *Application Discovery Service API Reference*.

Note
The AWS Application Discovery Agent is a component of the Amazon Inspector Service. For this reason, you will see the word *inspector* associated with the AWS agent download site and installation package.

Processing and Database Components

AWS Application Discovery Agents and the AWS Connector send data to Application Discovery Service, which includes a processing component that ingests the data and identifies (maps) IT assets. The service also includes the AWS Discovery database, a repository for discovered and mapped IT assets, which are called *configuration items*.

Data in the Discovery database is encrypted at rest. Encryption keys for the data are managed using the AWS Key Management Service (KMS).

Note
The AWS Discovery database is not a general-purpose enterprise configuration management database (CMDB). You can't save snapshots of discovered resources or track resource changes. The service does not alert you when resource configurations change. Similarly, though the service does collect performance data, it is not a general-purpose health monitoring solution.

When you use Application Discovery Service, you can specify filters and query specific configuration items in the AWS Discovery database. The service supports Server, Process, and Connection configuration items. This means you can specify a value for the following keys and query your IT assets.

Note
Server Performance, shown below, is an attribute of Server.

Server	Process	Connection	ServerPerformance
cpuType	name	sourceIp	numCores
hostName	commandLine	sourceProcess	numCpus
hypervisor	path	destinationIp	numDisks
osName	avgFreeRAMInKB	destinationPort	numNetworkCards
osVersion	minFreeRAMInKB	dstProcess	totalDiskSizeInKB
transportProtocol	avgDiskReadsPerSecondInKB	bytesIn	totalDiskFreeSizeInKB
lastReportedTime		ipVersion	totalRAMInKB
avgDiskWritesPerSecondInKB			
avgDiskReadIOPS			
avgDiskWriteIOPS			
maxDiskReadsPerSecondInKB			
maxDiskWritesPerSecondInKB			

Server	Process	Connection	ServerPerformance
maxDiskReadIOPS			
maxDiskWriteIOPS			
avgNetworkReadsPerSecondInKB			
avgNetworkWritesPerSecondInKB			
maxNetworkReadsPerSecondInKB			
maxNetworkWritesPerSecondInKB			

Setting Up AWS Application Discovery Service

Before you can use AWS Application Discovery Service, you must create an AWS account and configure access permissions for that account. You must also request permission to access Application Discovery Service directly from Amazon Web Services, a process called *whitelisting*. After you have been granted access to the service, you must set up and configure AWS Application Discovery Agents or agentless discovery in your computing environment.

Contents

Setting Up Access to AWS Application Discovery Service

To set up Application Discovery Service, complete the following tasks:

Application Discovery Service Set Up Tasks

Task	Details
Sign Up for AWS (p. 12)	You can skip this if you already have an Amazon Web Services (AWS) account.
Request Access (p. 12)	AWS partners and customers must request access to this service using the request form.
Create an IAM User (p. 12)	You can skip this if you already created an AWS Identity and Access Management (IAM) user account.
Attach Required IAM User Policies (p. 13)	The IAM user account requires access to specific Application Discovery Service resources. You can attach the policy in this section to provide access.

Task	Details
Create a Key Pair (p. 18)	If you want to complete the AWS Application Discovery Service Walkthrough (p. 25), you must create a key pair named *discovery*.

Sign Up for AWS

If you don't have an AWS account, use the following procedure to create one.

To create an AWS account

1. Open http://aws.amazon.com/, and then choose **Create an AWS Account**.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.

Make a note of your AWS account number, because you'll need it for the next task.

Request Access

Your AWS account must be granted access to Application Discovery Service, a process called *whitelisting*. This is true for AWS partners and customers alike. To request access, sign up for the AWS Application Discovery Service here. We will send you information about how to get started.

Create an IAM User

Services in AWS, such as Application Discovery Service, require that you provide credentials when you access them, so that the service can determine whether you have permission to access its resources. We don't recommend that you access AWS using the credentials for your AWS account; we recommend that you use AWS Identity and Access Management (IAM) instead. Create an IAM user, and then add the user to an IAM group with administrative permissions and grant this user administrative permissions. You can then access AWS using a special URL and the credentials for the IAM user.

For more information about IAM, see What Is IAM?

If you signed up for AWS but have not created an IAM user for yourself, you can create one using the IAM console. If you aren't familiar with using the console, see Working with the AWS Management Console for an overview.

To create an IAM user for yourself and add the user to an Administrators group

1. Sign in to the Identity and Access Management (IAM) console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Users**, and then choose **Add user**.
3. For **User name**, type a user name, such as `Administrator`. The name can consist of letters, digits, and the following characters: plus (+), equal (=), comma (,), period (.), at (@), underscore (_), and hyphen (-). The name is not case sensitive and can be a maximum of 64 characters in length.
4. Select the check box next to **AWS Management Console access**, select **Custom password**, and then type the new user's password in the text box. You can optionally select **Require password reset** to force the user to select a new password the next time the user signs in.

5. Choose **Next: Permissions**.
6. On the **Set permissions for user** page, choose **Add user to group**.
7. Choose **Create group**.
8. In the **Create group** dialog box, type the name for the new group. The name can consist of letters, digits, and the following characters: plus (+), equal (=), comma (,), period (.), at (@), underscore (_), and hyphen (-). The name is not case sensitive and can be a maximum of 128 characters in length.
9. For **Filter**, choose **Job function**.
10. In the policy list, select the check box for **AdministratorAccess**. Then choose **Create group**.
11. Back in the list of groups, select the check box for your new group. Choose **Refresh** if necessary to see the group in the list.
12. Choose **Next: Review** to see the list of group memberships to be added to the new user. When you are ready to proceed, choose Add permissions.

You can use this same process to create more groups and users, and to give your users access to your AWS account resources. To learn about using policies to restrict users' permissions to specific AWS resources, go to Access Management and Example Policies for Administering AWS Resources.

To sign in as this new IAM user, sign out of the AWS console, then use the following URL, where *your_aws_account_id* is your AWS account number without the hyphens (for example, if your AWS account number is `1234-5678-9012`, your AWS account ID is `123456789012`):

```
https://your_aws_account_id.signin.aws.amazon.com/console/
```

Enter the IAM user name (not your email address) and password that you just created. When you're signed in, the navigation bar displays "*your_user_name @ your_aws_account_id*".

If you don't want the URL for your sign-in page to contain your AWS account ID, you can create an account alias. From the IAM console, click **Dashboard** in the navigation pane. From the dashboard, click **Customize** and enter an alias such as your company name. To sign in after you create an account alias, use the following URL:

```
https://your_account_alias.signin.aws.amazon.com/console/
```

To verify the sign-in link for IAM users for your account, open the IAM console and check under **IAM users sign-in link** on the dashboard.

Attach Required IAM User Policies

Application Discovery Service uses the following IAM managed policies to control access to the service or components of the service. For information about how to attach managed policies to an IAM user account, see Working with Managed Policies.

AWSApplicationDiscoveryServiceFullAccess
: Grants the IAM user account access to the Application Discovery Service API. With this policy, the user can configure Application Discovery Service, start and stop agents, start and stop agentless discovery, and query data from the AWS Discovery Service database. This policy also grants the user access to Arsenal. Arsenal is an agent service managed and hosted by Amazon that forwards data to Application Discovery Service in the cloud.

AWSApplicationDiscoveryAgentAccess
: Grants AWS Application Discovery Agents access to register and communicate with Application Discovery Service. This policy needs to be attached to any user whose credentials are to be used by an AWS Application Discovery Agent.

AWSAgentlessDiscoveryService

Grants the AWS Agentless Discovery Connector running in your VMware vCenter Server access to register, communicate with, and share Connector health metrics with Application Discovery Service. This policy needs to be attached to any user whose credentials are to be used by the Connector.

Note

The `AWSAgentlessDiscoveryService` policy uses the following API actions: `awsconnector:RegisterConnector` and `awsconnector:GetConnectorHealth`. For more information about these actions, see API Actions of the AWSAgentlessDiscoveryService IAM Policy (p. 16).

Each of the Application Discovery Service managed policies is shown here so that you can customize them, if you want.

AWSApplicationDiscoveryServiceFullAccess

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

AWSApplicationDiscoveryAgentAccess

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

AWSAgentlessDiscoveryService

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "awsconnector:RegisterConnector",
                "awsconnector:GetConnectorHealth"
            ],
            "Resource": "*"
```

```
        },
        {
            "Effect": "Allow",
            "Action": "iam:GetUser",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::connector-platform-upgrade-info/*",
                "arn:aws:s3:::connector-platform-upgrade-info",
                "arn:aws:s3:::connector-platform-upgrade-bundles/*",
                "arn:aws:s3:::connector-platform-upgrade-bundles",
                "arn:aws:s3:::connector-platform-release-notes/*",
                "arn:aws:s3:::connector-platform-release-notes",
                "arn:aws:s3:::prod.agentless.discovery.connector.upgrade/*",
                "arn:aws:s3:::prod.agentless.discovery.connector.upgrade"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": [
                "arn:aws:s3:::import-to-ec2-connector-debug-logs/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "SNS:Publish"
            ],
            "Resource": "arn:aws:sns:*:*:metrics-sns-topic-for-*"
        },
        {
            "Sid": "Discovery",
            "Effect": "Allow",
            "Action": [
                "Discovery:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "arsenal",
            "Effect": "Allow",
            "Action": [
                "arsenal:RegisterOnPremisesAgent"
            ],
            "Resource": "*"
        }
    ]
}
```

API Actions of the AWSAgentlessDiscoveryService IAM Policy

The `AWSAgentlessDiscoveryService` IAM policy uses `awsconnector:RegisterConnector` and `awsconnector:GetConnectorHealth` to grant the AWS Agentless Discovery Connector access to register, communicate with, and share Connector health metrics with Application Discovery Service. More specifically, these APIs perform the following operations and return the following errors:

```
<operation name="RegisterConnector">
    <input target="RegisterConnectorRequest" /> Request type for
 RegisterConnector API.
    <output target="RegisterConnectorResponse" /> Response type for
 RegisterConnector API.
    <member name="snsTopicArn" target="String" /> Metrics SNS topic arn that
 is created/whitelisted for caller.
    <error target="AuthenticationFailureException" /> This exception is
 thrown if the credentials passed in the request could not be validated or
 user is not authorized to perform the operation.
    <error target="ServerInternalErrorException" /> This exception is thrown
 if there is erroneous logic in the service. It also includes all service
 dependency exceptions.
    <error target="ServiceUnavailableException" /> The request has failed due
 to a temporary failure of the server.
    <error target="ServerThrottleException" /> This exception is thrown
 if maximum number of request(for a given API) from an IAM user has been
 reached.
    <error target="InvalidParameterException" /> The request is missing
 required parameter(s) or has invalid parameter(s).
</operation>

<operation name="GetConnectorHealth">
    <input target="GetConnectorHealthRequest" /> Request type for
 GetConnectorHealth API.
    <member name="connectorId" target="String" /> Connector Id that will be
 used to verify identity of caller.
    <output target="GetConnectorHealthResponse" /> Response type for
 GetConnectorHealth API.
    <member name="serviceHealthList" target="ServiceHealthList" /> Contains
 all services' health information.
    <member target="ServiceHealth" /> The object that contains all health
 information for a given service.
    <member name="serviceName" target="String" /> The name of the service
 which was using connector health metrics publisher.
    <member name="healthList" target="HealthList" /> The list of health for
 the given service.
    <member target="Health" /> The object that represent a unique health
 metric that was published from the connector.
    <error target="AuthenticationFailureException" /> This exception is
 thrown if there is erroneous logic in the service. It also includes all
 service dependency exceptions.
    <error target="ServiceUnavailableException" /> The request has failed due
 to a temporary failure of the server.
    <error target="ServerThrottleException" /> This exception is thrown
 if maximum number of request(for a given API) from an IAM user has been
 reached.
    <error target="InvalidParameterException" /> The request is missing
 required parameter(s) or has invalid parameter(s).
```

```
</operation>

<structure name="Health"> The object that represent a unique health metric
 that was published from the connector.
    <member name="name" target="String" /> The name of the health that
 corresponds to "metric" field in Connector Metrics Publisher. The value of
 name is not user visible label that will show in Connector dashboard.
    <member name="value" target="String" /> The value for the health metric.
 It's a json that contains more information about how a health will display
 in connector dashboard.
    <member name="lastChecked" target="TimeStamp" /> The publish time for the
 last received metric.
</structure>
```

IAM Policy for the Walkthrough

If you plan to complete the AWS Application Discovery Service Walkthrough (p. 25) you must also attach the following IAM policy to your IAM user account. This policy is specific to the walkthrough and enables you to download files from Amazon S3, launch AWS CloudFormation templates, and access the Application Discovery Service API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1461631295000",
            "Effect": "Allow",
            "Action": [
                "s3:*"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "Stmt1461631338000",
            "Effect": "Allow",
            "Action": [
                "discovery:*"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "Stmt1461631339000",
            "Effect": "Allow",
            "Action": [
                "cloudformation:*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

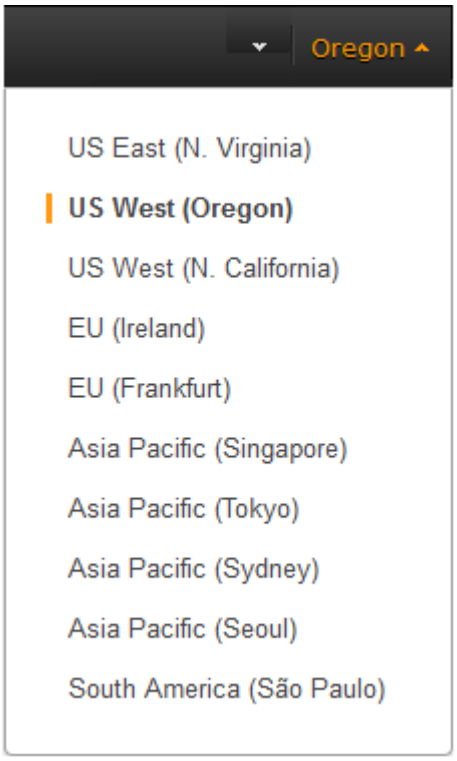
Create a Key Pair

If you plan to test Application Discovery Service using EC2 instances or if you plan to complete the AWS Application Discovery Service Walkthrough (p. 25), then you must create a key pair. AWS uses public-key cryptography (key pairs) to secure the login information for EC2 instances. For the walkthrough, you must create a key pair named *discovery* because the walkthrough uses AWS CloudFormation templates to create instances. These instances use a key pair named *discovery*. In order to access these instances, you must provide a key pair with the same name.

If you haven't created a key pair already, you can create one using the Amazon EC2 console. Note that if you plan to launch instances in multiple regions, you'll need to create a key pair in each region. For more information about regions, see Regions and Availability Zones (Linux).

To create a key pair

1. Sign in to AWS using the URL that you created when you created your AWS user account.
2. From the AWS dashboard, choose **EC2** to open the Amazon EC2 console.
3. From the navigation bar, select a region for the key pair. You can select any region that's available to you, regardless of your location. However, key pairs are specific to a region; for example, if you plan to launch an instance in the US West (Oregon) Region, you must create a key pair for the instance in the US West (Oregon) Region.



4. In the navigation pane, under **NETWORK & SECURITY**, click **Key Pairs**.

   Tip
   The navigation pane is on the left side of the console. If you do not see the pane, it might be minimized; click the arrow to expand the pane. You might have to scroll down to see the **Key Pairs** link.

5. Click **Create Key Pair**.
6. Enter a name for the new key pair in the **Key pair name** field of the **Create Key Pair** dialog box, and then click **Create**. Choose a name that is easy for you to remember, such as your IAM user name, followed by `-key-pair`, plus the region name. For example, *me*-key-pair-*uswest2*.

   For the walkthrough, create a key pair named *discovery*.
7. The private key file is automatically downloaded by your browser. The base file name is the name you specified as the name of your key pair, and the file name extension is `.pem`. Save the private key file in a safe place.

   Important
   This is the only chance for you to save the private key file. You'll need to provide the name of your key pair when you launch an instance and the corresponding private key each time you connect to the instance.
8. If you will use an SSH client on a Mac or Linux computer to connect to your Linux instance, use the following command to set the permissions of your private key file so that only you can read it.

```
$ chmod 400 your_user_name-key-pair-region_name.pem
```

For more information, see Amazon EC2 Key Pairs.

Setting Up Agentless Discovery

To set up agentless discovery, you must deploy the AWS Agentless Discovery Connector virtual appliance on a VMware vCenter Server host in your on-premises environment. After you deploy the virtual appliance, you must configure the Connector using the web-based Connector Management Console.

Deploying the Connector Virtual Appliance

Before you can download the AWS Connector virtual appliance, you must register with AWS. You receive information about how to download the virtual appliance (as an Open Virtualization Archive (OVA) file) after you register with AWS. For more information, see Request Access from AWS (p. 2).

To deploy the Connector virtual appliance

1. Sign in to vCenter as a VMware administrator.
2. From the **File** menu, choose **Deploy OVF Template**. Enter the URL that was sent to you after your completed the registration.

3. Complete the wizard.
4. On the **Disk Format** page, select one of the thick provision disk types. We recommend that you select **Thick Provision Eager Zeroed**, because it has the best performance and reliability. However, it requires several hours to zero out the disk. Do not select **Thin Provision**. This option makes deployment faster but significantly reduces disk performance. For more information, see Types of supported virtual disks in the VMware documentation.
5. Locate the newly deployed template in the vSphere client inventory tree, right-click it, and select **Power**, **Power On**. Right-click the template again and select **Open Console**. The console displays the IP address of the Connector console. Save the IP address in a secure location. You'll need it to complete the Connector setup process.

Configuring the Connector

To finish the setup process, open a web browser and complete the following procedure.

To configure the Connector using the Connector management console

1. In a web browser, type the following URL in the address bar: https://*ip_address*/, where *ip_address* is the IP address of the Connector management console that you saved earlier.
2. In **Step 1: License agreement**, read and accept the agreement, and then choose **Next**.
3. In **Step 2: Create a password**, type a strong password for access to the Connector, and then choose **Next**.
4. In **Step 3: Network information**, read the information provided and configure network settings.
5. In **Step 4: Log uploads and upgrades**, choose the options you want and then choose **Next**.
6. In **Step 5: Discovery Connector setup**, choose **Configure vCenter credentials**.

   a. In the **vCenter IP address** field, type the IP address of your VMware vCenter Server host.
   b. In the **vCenter username** field, type the name of a local or domain user that the Connector uses to communicate with vCenter. For domain users, use the form *domain\username* or *username@domain*.
   c. In the **vCenter password** field, type the user password.
   d. Choose **Ignore security certificate** to bypass SSL certificate validation with vCenter.
   e. Choose **Save**.
7. Choose **Configure AWS credentials** and type the credentials for the IAM user who is assigned the

   ```
   AWSAgentlessDiscoveryService
   ```

   IAM policy that you created in Attach Required IAM User Policies (p. 13). Choose **Save**.
8. Choose **Configure where to publish data** and choose the publishing options you want. Choose **Save**.

   Note
   After you complete this initial setup, you can access Connector settings by using SSH and the connector IP address: root@*Connector_IP_address*. The default user name is ec2-user and the default password is ec2pass. We strongly encourage you to change the value of the default user name and password.

You've completed the setup process and are ready to start using agentless discovery with Application Discovery Service. You can use the Application Discovery Service command line interface (CLI) to start collecting data, manage the service, tag and query configuration items, and export data. You can export data as a CSV or an XML file to an Amazon S3 bucket or an application that enables you to

view and evaluate the data. For an example of how to use the API, see AWS Application Discovery Service Walkthrough (p. 25). For more information about the Application Discovery Service API, see the *Application Discovery Service API Reference*.

Setting Up AWS Application Discovery Agents

The AWS Application Discovery Agent is Amazon software that you install on on-premises servers and virtual machines (VMs) that you can target for discovery and migration. Agents run on Linux and Windows servers and collect configuration and activity information about your applications and infrastructure. Agents send this data to Application Discovery Service using Secure Sockets Layer (SSL) encryption. Several minutes after you instruct agents to start collecting data, Application Discovery Service receives the data and begins to process connections and discover dependencies. Discovered IT assets are called *configuration items*. For more information about the AWS Application Discovery Agent, see AWS Application Discovery Service Components (p. 4).

Before You Begin

If you installed a Linux preview version of the AWS Application Discovery Agent, you must uninstall it or you won't be able to install the current version. This does not apply to the agent for Windows. Use the following command to download the removal script to your Linux instance or server:

```
curl -O https://d1wk0tztpsntt1.cloudfront.net/linux/latest/
remove_preview_agent
```

To remove the preview agent, run the following command:

```
sudo ./remove_preview_agent
```

Installing the AWS Application Discovery Agent

When you are ready to install the AWS Application Discovery Agent in an on-premise data center, work with application owners to identify the IP address or DNS name of at least one of the server or VM workloads hosting the application. With the help of the application's operations team, install an AWS Application Discovery Agent within that workload. The agent will identify the servers or VMs that communicate with the workload and report the data to Application Discovery Service. You can then install agents on one or more of these servers to discover their dependencies. You will then iterate through this process until you have discovered all of the applications dependencies.

> **Important**
> Only certified AWS partners can download the installer from the Amazon Partner Network. If you are an AWS partner who wants to join the list of certified partners, contact the Application Discovery Service team through the AWS partner program.

This section includes the following information.

- On-Premises Installation for Microsoft Windows (p. 21)
- On-Premises Installation for Linux (p. 23)

On-Premises Installation for Microsoft Windows

Use the following procedure to install an AWS Application Discovery Agent on a Windows-based VM or server in your data center.

To install the AWS Application Discovery Agent in your data center

1. Download the agent installer to a host server or VM.
2. Open a command prompt as an administrator and navigate to the location where you saved the installation package.
3. Run the following command to install the agent:

```
DiscoveryAgentInstall.exe REGION=us-west-2
```

Note
AWS Application Discovery Agents automatically download and apply updates as they become available. If you don't want agents to download and apply updates automatically, then run the following command when you install the agent:

```
DiscoveryAgentInstall.exe REGION=us-west-2 AUTOUPDATE=No
```

4. Open the following file and specify your AWS credentials. If you do not locate the file, you must create it.

   %SystemRoot%\system32\config\systemprofile\.aws\credentials

   Specify credentials in the following format:

```
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

Note
The IAM policy attached to your AWS account must have access to Application Discovery Service resources. For more information, see Attach Required IAM User Policies (p. 13).

5. Update your firewall settings. The AWS Application Discovery Agent requires outbound access to arsenal.us-west-2.amazonaws.com:443. It does not require any inbound ports to be open. Agents also work with transparent web proxies.

Verifying the Installation and Managing AWS Application Discovery Agents

Use the following procedure to verify the agent installation.

To verify that the AWS Application Discovery Agent is running

1. On the EC2 instance where you installed the AWS Application Discovery Agent, open a command prompt with administrator permissions and navigate to C:/Program Files/Amazon Web Services/Aws Agent.
2. Run the following command.

```
AWSAgentStatus.exe
```

   This command returns the status of the currently running agent, or an error stating that the agent cannot be contacted.

To start or stop an AWS Application Discovery Agent, use Windows Services Manager to start or stop the AWS Agent Service and the AWS Agent Updater Service. To uninstall an AWS Agent, use Add/Remove Programs to remove these services.

On-Premises Installation for Linux

Use the following procedure to install an AWS Application Discovery Agent on a Linux-based VM or server in your data center.

To install the AWS Application Discovery Agent in your data center

1. Log onto the Linux-based server or VM and open the AWS CLI.
2. Open the following file and specify your AWS credentials.

   ~/.aws/credentials

   ```
   [default]
   aws_access_key_id=AKIAIOSFODNN7EXAMPLE
   aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   ```

   Note
   The IAM policy attached to your AWS account must have access to Application Discovery Service resources. For more information, see Attach Required IAM User Policies (p. 13).
3. Download the agent installer to a host server or VM.
4. Open the following file and verify that the endpoint is (arsenal.us-west-2.amazonaws.com) and the region is (us-west-2):

   /opt/aws/awsagent/etc/agent.cfg
5. Run the following command to install the agent in the us-west-2 region:

   ```
   sudo bash install -r us-west-2
   ```

   Note
   Agents automatically download and apply updates as they become available. If you don't want agents to download and apply updates automatically, then run the following command when you install the agent:

   ```
   sudo bash install -u false
   ```
6. After the installation completes, use the following command to remove the agent installation script:

   ```
   rm install
   ```
7. Update your firewall settings. The AWS Application Discovery Agent requires outbound access to arsenal.us-west-2.amazonaws.com:443. It does not require any inbound ports to be open. Agents also work with transparent web proxies.

Verifying the Installation and Managing AWS Application Discovery Agents

The following table lists commands that you can use to manage or uninstall agents.

Linux Commands for AWS Application Discovery Agents

Task	Command
Start an agent	**sudo /etc/init.d/awsagent start**
Verify an agent is running	**sudo /opt/aws/awsagent/bin/awsagent status**
Stop an agent	**sudo /etc/init.d/awsagent stop**
Restart an agent	**sudo /etc/init.d/awsagent restart**
Uninstall an agent from Amazon Linux, CentOS, or Red Hat	**yum remove AwsAgent**
Uninstall an agent from Ubuntu Server	**apt-get remove awsagent**

After you install agents, you can use the Application Discovery Service API to programmatically manage agents, tag and query configuration items, and export data. You can export data as a CSV file to an Amazon S3 bucket or an application that enables you to view and evaluate the data. For an example of how to use the API, see AWS Application Discovery Service Walkthrough (p. 25). For more information about the Application Discovery Service API, see the *Application Discovery Service API Reference*.

AWS Application Discovery Service Walkthrough

This section walks you through an example that shows you how to use AWS Application Discovery Service with Amazon EC2 instances. The walkthrough uses sample scripts and a sample application that you'll need to download.

This topic includes the following.

Setting Up the Environment

The following procedure shows you how to set up an environment from an Amazon EC2 instance. From this instance, you will download and configure the AWS Command Line Interface (CLI), download and run a setup script, and launch a sample application that you will use to access the Application Discovery Service API.

Before you begin

Verify that you have completed the steps in Setting Up AWS Application Discovery Service (p. 11). By completing the steps in that topic, you ensure the following:

- You have an AWS account.
- Your account has been *whitelisted* and granted access to Application Discovery Service.
- You have a key pair named *discovery*.

- The IAM policy attached to your user account has access to Application Discovery Service resources, including the Arsenal service.

You can setup either Ubuntu Linux or Microsoft Windows EC2 instances for this walkthrough.

- Configure an Ubuntu Linux Environment (p. 26).
- Configure a Microsoft Windows Environment (p. 27).

Configure an Ubuntu Linux Environment

1. Open the Amazon EC2 console and switch to the us-west-2 region.
2. Launch an Ubuntu 14.04 Amazon Machine Image (AMI) using this ID: ami-966687f6. You must launch this AMI because it includes dependencies required for this walkthrough.
3. Log onto the Ubuntu instance and use the following commands to install and update the AWS CLI.

   ```
   sudo apt-get update
   sudo apt-get install awscli
   aws configure
   ```

   When prompted, specify your whitelisted AWS credentials and set the default region to us-west-2.
4. Use the following command to download the training script from an Amazon S3 bucket. The blank space and period (.) at the end of the command are required.

   ```
   aws s3 cp s3://discovery.training.cli.formal/trainingsetup.sh .
   ```
5. The script performs the following tasks:

   - Installs the dependencies required for the Application Discovery Service sample application to run.
   - Downloads an AWS CloudFormation template and stores it locally in the /tmp folder.
   - Installs the Application Discovery Service sample application as a Python package.
   - Installs sample scripts that use the Application Discovery Service API to implement commonly used search queries as a standalone executable.

   Note
   You can view the source code for the scripts in the following directory: ~/discoverytrainingcli/discoverycli/src/. The source code also serves as an example of how to call the API. A readme document with information about the CLI commands is located in the following directory: ~/discoverytrainingcli/discoverycli/doc/README.
6. Use the following command to run the training set up script.

   ```
   sudo bash trainingsetup.sh .
   ```

   During the installation, the script prompts you to specify your AWS credentials. The sample application uses these credentials to make calls to Application Discovery Service.
7. Use the following command to view the sample application Help and verify that the sample application installed correctly.

   ```
   awsdiscovery –h
   ```

```
acbc328a2d9d:DiscoveryCli TestUser$ awsdiscovery -h
usage: awsdiscovery [-h]

                    {start-data-collection,list-connections,list-processe
scribe-agents,describe-export-tasks,list-servers}
                    ...

positional arguments:
  {start-data-collection,list-connections,list-processes,create-tag,delet
ribe-export-tasks,list-servers}

optional arguments:
  -h, --help            show this help message and exit
```

8. Use the following command to provision Topology 1 using AWS CloudFormation templates. Topology 1 includes two clients that will communicate with two Web servers but only one database.

```
aws cloudformation create-stack --stack-name topology1 --template-body
 file:///tmp/SA_cloud_formation_topology2_vpc_with_key_without_EIP
```

9. After the command execution completes, you can view the resources that were created using the following command.

```
aws cloudformation list-stack-resources --stack-name topology1
```

10. Use the following command to provision Topology 2 using AWS CloudFormation templates. Topology 2 includes a client that will communicate with a Web server and one database.

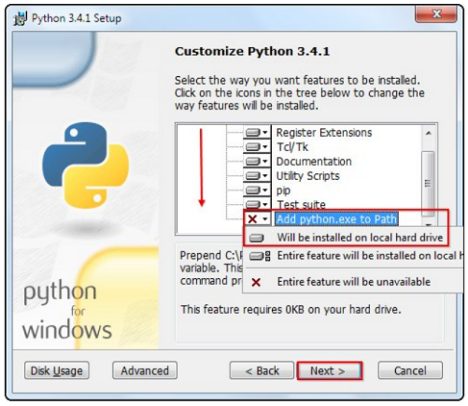
```
aws cloudformation create-stack --stack-name topology2 --template-body
 file:///tmp/SA_cloud_formation_vpc_with_key_without_EIP
```

11. After the command execution completes, you can view the resources that were created using the following command.

```
aws cloudformation list-stack-resources --stack-name topology2
```

Configure a Microsoft Windows Environment

1. Open the Amazon EC2 console and switch to the us-west-2 region.
2. Launch a Microsoft Windows Server 2012 R2 Base Amazon Machine Image (AMI).
3. Log onto the Windows instance, download, and install Python. The sample application for this walkthrough is a Python application.
4. In the Install Wizard, scroll down to locate the **Add Python.exe to Path** option. Choose **Will be installed on local hard drive** and then complete the wizard.



5. Open a command prompt and run the following command to install the AWS CLI.

```
pip install awscli
```

   If the AWS CLI is already installed, run the following command to update it.

```
pip install --upgrade awscli
```

6. Run the following command to configure the session.

```
aws configure
```

   When prompted, specify your AWS credentials and set the default region to us-west-2. The AWS credentials must have been granted access to the Application Discovery Service. For more information, see Setting Up AWS Application Discovery Service (p. 11).

7. Use the following command to download the training script from an Amazon S3 bucket. The blank space and period (.) at the end of the command are required.

```
aws s3 cp s3://discovery.training.cli.formal/DiscoveryCli.tgz .
```

   By default, the command downloads the file to the current user folder. For example, if you run the command as the Administrator, the command downloads the file to the C:\Users\Administrator folder.

8. Unzip the DiscoveryCli.tgz file using a zip/unzip utility like WinZip or 7-Zip. The extraction places a folder called DiscoveryCli in the specified location. Inside that folder, locate and unzip the DiscoveryCli.tar file.

9. In a command prompt, run the following command in the directory where the setup.py file was extracted during the most recent unzip. If you chose the defaults, the file is located here: C:\Users*user name*\DiscoveryCli\DiscoveryCli\DiscoveryCli

```
python setup.py install
```

10. Run the following command to download files for this walkthrough. Replace *Username* with the name of your Windows account.

```
aws s3 cp s3://poseidon.service.json/service-2.json c:\Users\Username\.aws
\models\
```

11. The command performs the following tasks:

- Installs the dependencies required for the Application Discovery Service sample application to run.
- Installs the Application Discovery Service sample application as a Python package.
- Installs sample scripts that use the Application Discovery Service API to implement commonly used search queries as a standalone executable.

Note
You can view the source code for the scripts in the following directory: C:\Users*user name*\DiscoveryCli\DiscoveryCli\DiscoveryCli\src. The source code also serves as an example of how to call the API. A README document with information about the CLI commands is located in the following directory: C:\Users*user name*\DiscoveryCli\DiscoveryCli\DiscoveryCli\doc\README

12. Use the following command to view the sample application Help and verify that the sample application installed correctly.

```
python C:\python27\Scripts\awsdiscovery --help
```

13. Use the following command to download AWS CloudFormation templates and store them locally in the /tmp folder. This command provisions Topology 1. Topology 1 includes two clients that will communicate with two Web servers but only one database.

```
aws s3 cp s3://sa.cf.template/SA_cloud_formation_vpc_with_key_without_EIP
 C:\tmp\
```

14. Launch the template using the following command.

```
C:\Users\user name\DiscoveryCli\DiscoveryCli>aws cloudformation
 create-stack --stack-name topology1 --template-body file://C:/tmp/
SA_cloud_formation_vpc_with_key_without_EIP
```

15. Use the following command to provision Topology 2. Topology 2 includes a client that will communicate with a Web server and one database.
16. Launch the template using the following command.

```
C:\Users\user name\DiscoveryCli\DiscoveryCli>aws cloudformation
 create-stack --stack-name topology2 --template-body file://C:/tmp/
SA_cloud_formation_topology2_vpc_with_key_without_EIP
```

Managing AWS Application Discovery Agents

After you install the AWS Application Discovery Agent on an Amazon EC2 instance, or on a server or virtual machine (VM) in your data center, you're ready for the next section of the walkthrough. This section shows you how to manage the agents that were installed in the topologies as part of the AWS CloudFormation templates.

Note
Instance IDs throughout this walkthrough have been made generic and are shown, for example, in the following format: i-xxxxxxx1

1. Use the following command to get the status and location of installed agents.

   On Linux:

```
awsdiscovery describe-agents
```

On Windows:

```
python C:\python27\Scripts\awsdiscovery describe-agents
```

The command output includes the agent status.

- UNKNOWN – Application Discovery Service does not know the agent status.
- HEALTHY – Agents responded to the command, but are not collecting data.
- RUNNING – Agents are collecting data.

The following sample output shows that all agents are healthy, but not running.

```
+------------+----------+-----------------+---------------+
| Id         | Health   | HostName        | IPAddress     |
+============+==========+=================+===============+
|i-xxxxxxx1 | HEALTHY  | ip-10-0-1-250   | 10.0.1.250    |
+------------+----------+-----------------+---------------+
|i-xxxxxxx2 | HEALTHY  | ip-10-0-1-187   | 10.0.1.187    |
+------------+----------+-----------------+---------------+
|i-xxxxxxx3 | HEALTHY  | ip-10-0-0-61    | 10.0.0.61     |
+------------+----------+-----------------+---------------+
|i-xxxxxxx4 | HEALTHY  | ip-10-0-1-31    | 10.0.1.31     |
+------------+----------+-----------------+---------------+
|i-xxxxxxx5 | HEALTHY  | ip-10-0-1-105   | 10.0.1.105    |
+------------+----------+-----------------+---------------+
|i-xxxxxxx6 | HEALTHY  | ip-10-0-1-249   | 10.0.1.249    |
+------------+----------+-----------------+---------------+
|i-xxxxxxx7 | HEALTHY  | ip-10-0-0-109   | 10.0.0.109    |
+------------+----------+-----------------+---------------+
|i-xxxxxxx8 | HEALTHY  | ip-10-0-1-166   | 10.0.1.166    |
+------------+----------+-----------------+---------------+
```

2. Use the following command to instruct specific agents to start collecting data.

   On Linux:

   ```
   awsdiscovery start-data-collection i-xxxxxxx1 i-xxxxxx2 i-xxxxxx3 i-
   xxxxxx4 i-xxxxxx5 i-xxxxxx6 i-xxxxxx7 i-xxxxxx8
   ```

   On Windows:

   ```
   python C:\python27\Scripts\awsdiscovery start-data-collection i-xxxxxxx1
    i-xxxxxx2 i-xxxxxx3 i-xxxxxx4 i-xxxxxx5 i-xxxxxx6 i-xxxxxx7 i-xxxxxx8
   ```

```
+------------+---------------+
| Id         | Description   |
+============+===============+
| i-xxxxxxx1| Succeeded     |
+------------+---------------+
| i-xxxxxxx2| Succeeded     |
+------------+---------------+
|i-xxxxxxx3 | Succeeded     |
+------------+---------------+
|i-xxxxxxx4 | Succeeded     |
+------------+---------------+
|i-xxxxxxx5| Succeeded     |
+------------+---------------+
|i-xxxxxxx6| Succeeded     |
+------------+---------------+
|i-xxxxxxx7| Succeeded     |
+------------+---------------+
|i-xxxxxxx8| Succeeded     |
+------------+---------------+
```

3. Use the following command to get the agent status after they were instructed to start collecting data.

   On Linux:

```
awsdiscovery describe-agents
```

On Windows:

```
python C:\python27\Scripts\awsdiscovery describe-agents
```

```
+-----------+----------+-----------------+---------------+
| Id        | Health   | HostName        | IPAddress     |
+===========+==========+=================+===============+
|i-xxxxxxx1| RUNNING  | ip-10-0-1-250   | 10.0.1.250    |
+-----------+----------+-----------------+---------------+
|i-xxxxxxx2| RUNNING  | ip-10-0-1-187   | 10.0.1.187    |
+-----------+----------+-----------------+---------------+
|i-xxxxxxx3| RUNNING  | ip-10-0-0-61    | 10.0.0.61     |
+-----------+----------+-----------------+---------------+
|i-xxxxxxx4| RUNNING  | ip-10-0-1-31    | 10.0.1.31     |
+-----------+----------+-----------------+---------------+
|i-xxxxxxx5| RUNNING  | ip-10-0-1-105   | 10.0.1.105    |
+-----------+----------+-----------------+---------------+
|i-xxxxxxx6| RUNNING  | ip-10-0-1-249   | 10.0.1.249    |
+-----------+----------+-----------------+---------------+
|i-xxxxxxx7| RUNNING  | ip-10-0-0-109   | 10.0.0.109    |
+-----------+----------+-----------------+---------------+
|i-xxxxxxx8| RUNNING  | ip-10-0-1-166   | 10.0.1.166    |
+-----------+----------+-----------------+---------------+
```

Application Discovery Service takes several minutes to receive and process the data.

4. Use the following command to instruct specific agents to stop collecting data.

   On Linux:

   ```
   awsdiscovery stop-data-collection i-xxxxxxx1 i-xxxxxxx2 i-xxxxxxx3 i-
   xxxxxxx4 i-xxxxxxx5 i-xxxxxxx6 i-xxxxxxx7 i-xxxxxxx8
   ```

   On Windows:

   ```
   python C:\python27\Scripts\awsdiscovery stop-data-collection i-xxxxxxx1 i-
   xxxxxxx2 i-xxxxxxx3 i-xxxxxxx4 i-xxxxxxx5 i-xxxxxxx6 i-xxxxxxx7 i-xxxxxxx8
   ```

```
+-----------+--------------+
| Id        | Description  |
+===========+==============+
|i-xxxxxxx1| Succeeded    |
+-----------+--------------+
|i-xxxxxxx2| Succeeded    |
+-----------+--------------+
|i-xxxxxxx3| Succeeded    |
+-----------+--------------+
|i-xxxxxxx4| Succeeded    |
+-----------+--------------+
|i-xxxxxxx5| Succeeded    |
+-----------+--------------+
|i-xxxxxxx6| Succeeded    |
+-----------+--------------+
|i-xxxxxxx7| Succeeded    |
+-----------+--------------+
|i-xxxxxxx8| Succeeded    |
+-----------+--------------+
```

Working with Configuration Items

A *configuration item* is an IT asset that was discovered in your data center by an agent. With Application Discovery Service, you can query specific configuration items.

1. Use the following command to list discovered servers.

On Linux:

```
awsdiscovery list-servers
```

On Windows:

```
python C:\python27\Scripts\awsdiscovery list-servers
```

```
+--------------+-----------------------------------------+------------+-------------------+
| Hostname     | OsName                                  | IP         | MAC               |
+==============+=========================================+============+===================+
| ip-10-0-0-109 | Linux - Amazon Linux AMI release 2015.03 | 10.0.0.109 | 06:22:14:9c:96:f9 |
+--------------+-----------------------------------------+------------+-------------------+
| ip-10-0-1-249 | Linux - Amazon Linux AMI release 2015.03 | 10.0.1.249 | 06:cd:9c:00:bf:b3 |
+--------------+-----------------------------------------+------------+-------------------+
| ip-10-0-1-31  | Linux - Amazon Linux AMI release 2015.03 | 10.0.1.31  | 06:2c:f4:70:73:13 |
+--------------+-----------------------------------------+------------+-------------------+
| ip-10-0-0-61  | Linux - "Ubuntu 14.04.2 LTS"             | 10.0.0.61  | 06:3f:d8:ee:cb:e3 |
+--------------+-----------------------------------------+------------+-------------------+
| ip-10-0-1-250 | Linux - Amazon Linux AMI release 2015.03 | 10.0.1.250 | 06:8a:6c:92:32:b1 |
+--------------+-----------------------------------------+------------+-------------------+
| ip-10-0-1-187 | Linux - Amazon Linux AMI release 2015.03 | 10.0.1.187 | 06:69:7a:61:cb:1d |
+--------------+-----------------------------------------+------------+-------------------+
| ip-10-0-1-105 | Linux - Amazon Linux AMI release 2015.03 | 10.0.1.105 | 06:82:6b:4e:03:bb |
+--------------+-----------------------------------------+------------+-------------------+
| ip-10-0-1-166 | Linux - "Ubuntu 14.04.2 LTS"             | 10.0.1.166 | 06:77:95:91:a6:e7 |
+--------------+-----------------------------------------+------------+-------------------+
```

2. Use the following command to list servers running a specific process.

   On Linux:

   ```
   awsdiscovery list-servers --process_name mysql
   ```

   On Windows:

   ```
   python C:\python27\Scripts\awsdiscovery list-servers --process_name mysql
   ```

   ```
   +--------------+-----------------------------------------+------------+-------------------+
   | Hostname     | OsName                                  | IP         | MAC               |
   +==============+=========================================+============+===================+
   | ip-10-0-0-61  | Linux - "Ubuntu 14.04.2 LTS"             | 10.0.0.61  | 06:3f:d8:ee:cb:e3 |
   +--------------+-----------------------------------------+------------+-------------------+
   | ip-10-0-1-187 | Linux - Amazon Linux AMI release 2015.03 | 10.0.1.187 | 06:69:7a:61:cb:1d |
   +--------------+-----------------------------------------+------------+-------------------+
   | ip-10-0-1-166 | Linux - "Ubuntu 14.04.2 LTS"             | 10.0.1.166 | 06:77:95:91:a6:e7 |
   +--------------+-----------------------------------------+------------+-------------------+
   ```

3. Use the following command to list a server by its MAC address and display detailed attributes.

   On Linux:

   ```
   awsdiscovery list-servers --mac 06:22:14:9c:96:f9 –verbose
   ```

   On Windows:

   ```
   python C:\python27\Scripts\awsdiscovery list-servers --mac
    06:22:14:9c:96:f9 –verbose
   ```

   ```
   +--------------+-----------+-------------------+-----------+----------+-----------+-----------------+-----------------+----------+-----------------+-------------+
   | HostName     | IP        | MAC               |  Prov CPU |  Max CPU |  Avg CPU |  Prov mem(in MB) |  Max mem(in MB) |  Num disk |  Disk size(in MB) |  Ntwk cards |
   +==============+===========+===================+===========+==========+==========+=================+=================+==========+=================+=============+
   | ip-10-0-0-109 | 10.0.0.109 | 06:22:14:9c:96:f9 |         1 |       61 |    0.175 |          592.301 |          141.23 |         1 |          7935.43 |           2 |
   +--------------+-----------+-------------------+-----------+----------+----------+-----------------+-----------------+----------+-----------------+-------------+
   ```

Query Server Dependencies and Connections

This section shows you how to query configuration items based on the following discovery objectives. In this case, the user wants to:

- Inspect server dependencies based on network connectivity.
- Identify the processes in a server that were responsible for the connection.

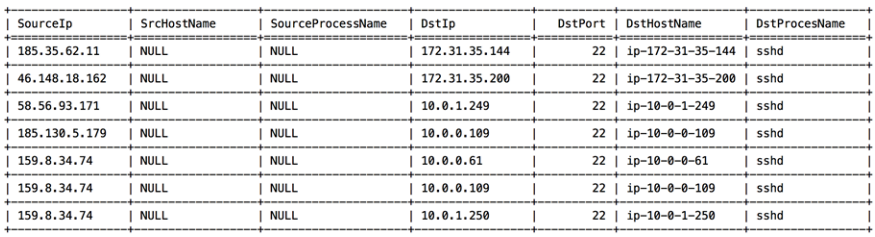
1. Use the following command to list all connections.

   On Linux:

   ```
   awsdiscovery list-connections
   ```

   On Windows:

   ```
   python C:\python27\Scripts\awsdiscovery list-connections
   ```

SourceIp	SrcHostName	SourceProcessName	DstIp	DstPort	DstHostName	DstProcesName
185.35.62.11	NULL	NULL	172.31.35.144	22	ip-172-31-35-144	sshd
46.148.18.162	NULL	NULL	172.31.35.200	22	ip-172-31-35-200	sshd
58.56.93.171	NULL	NULL	10.0.1.249	22	ip-10-0-1-249	sshd
185.130.5.179	NULL	NULL	10.0.0.109	22	ip-10-0-0-109	sshd
159.8.34.74	NULL	NULL	10.0.0.61	22	ip-10-0-0-61	sshd
159.8.34.74	NULL	NULL	10.0.0.109	22	ip-10-0-0-109	sshd
159.8.34.74	NULL	NULL	10.0.1.250	22	ip-10-0-1-250	sshd

   Output in the image has been truncated.

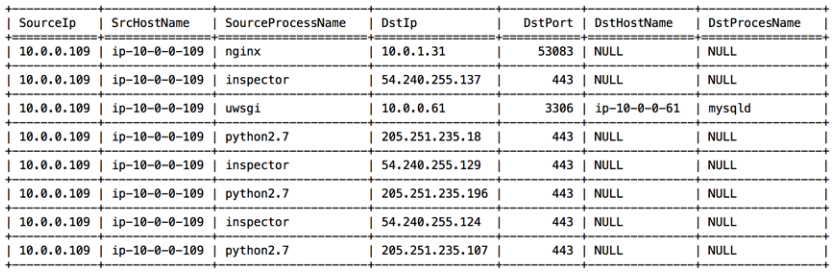
2. Use the following command to list all dependencies for a specific host.

   On Linux:

   ```
   awsdiscovery list-connections --src_hostname ip-10-0-0-109
   ```

   On Windows:

   ```
   python C:\python27\Scripts\awsdiscovery list-connections --src_hostname
    ip-10-0-0-109
   ```

SourceIp	SrcHostName	SourceProcessName	DstIp	DstPort	DstHostName	DstProcesName
10.0.0.109	ip-10-0-0-109	nginx	10.0.1.31	53083	NULL	NULL
10.0.0.109	ip-10-0-0-109	inspector	54.240.255.137	443	NULL	NULL
10.0.0.109	ip-10-0-0-109	uwsgi	10.0.0.61	3306	ip-10-0-0-61	mysqld
10.0.0.109	ip-10-0-0-109	python2.7	205.251.235.18	443	NULL	NULL
10.0.0.109	ip-10-0-0-109	inspector	54.240.255.129	443	NULL	NULL
10.0.0.109	ip-10-0-0-109	python2.7	205.251.235.196	443	NULL	NULL
10.0.0.109	ip-10-0-0-109	inspector	54.240.255.124	443	NULL	NULL
10.0.0.109	ip-10-0-0-109	python2.7	205.251.235.107	443	NULL	NULL

   Output in the image has been truncated.

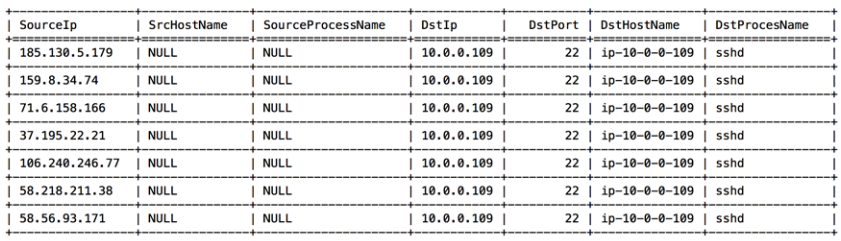
3. Use the following command to list all dependents for a specific host.

On Linux:

```
awsdiscovery list-connections --dest_hostname ip-10-0-0-109
```

On Windows:

```
python C:\python27\Scripts\awsdiscovery list-connections --dest_hostname
 ip-10-0-0-109
```



SourceIp	SrcHostName	SourceProcessName	DstIp	DstPort	DstHostName	DstProcesName
185.130.5.179	NULL	NULL	10.0.0.109	22	ip-10-0-0-109	sshd
159.8.34.74	NULL	NULL	10.0.0.109	22	ip-10-0-0-109	sshd
71.6.158.166	NULL	NULL	10.0.0.109	22	ip-10-0-0-109	sshd
37.195.22.21	NULL	NULL	10.0.0.109	22	ip-10-0-0-109	sshd
106.240.246.77	NULL	NULL	10.0.0.109	22	ip-10-0-0-109	sshd
58.218.211.38	NULL	NULL	10.0.0.109	22	ip-10-0-0-109	sshd
58.56.93.171	NULL	NULL	10.0.0.109	22	ip-10-0-0-109	sshd

Output in the image has been truncated.

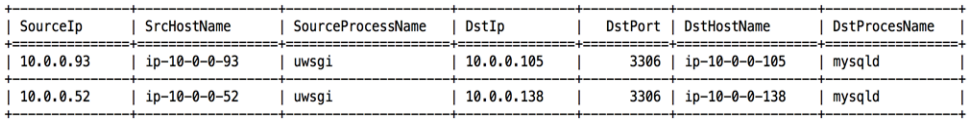
4. Use the following command to list Linux-uwsgi servers communicating with Ubuntu-MySQL servers.

   On Linux:

   ```
   aluwsgi_ubmysql
   ```

   On Windows

   ```
   python C:\python27\Scripts\aluwsgi_ubmysql
   ```



SourceIp	SrcHostName	SourceProcessName	DstIp	DstPort	DstHostName	DstProcesName
10.0.0.93	ip-10-0-0-93	uwsgi	10.0.0.105	3306	ip-10-0-0-105	mysqld
10.0.0.52	ip-10-0-0-52	uwsgi	10.0.0.138	3306	ip-10-0-0-138	mysqld

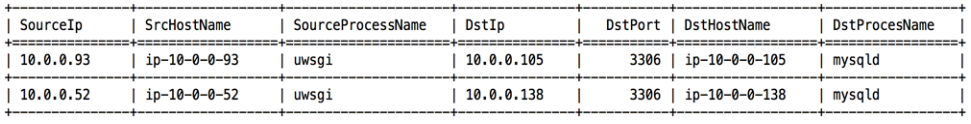
5. Use the following command to list Linux-uwsgi servers communicating with Ubuntu-MySQL servers.

   On Linux:

   ```
   aluwsgi_ubmysql
   ```

   On Windows:

   ```
   python C:\python27\Scripts\aluwsgi_ubmysql
   ```



SourceIp	SrcHostName	SourceProcessName	DstIp	DstPort	DstHostName	DstProcesName
10.0.0.93	ip-10-0-0-93	uwsgi	10.0.0.105	3306	ip-10-0-0-105	mysqld
10.0.0.52	ip-10-0-0-52	uwsgi	10.0.0.138	3306	ip-10-0-0-138	mysqld

Working with Multi-Layered Stacks

The previous examples focused on discovery in a two-layered stack (for example, a server communicating with a database). This section shows you how to query configuration items for a three-layered stack. In the samples below, the stack includes a client computer that communicates with an NGINX Web server that communicates with a MySQL database.

Use the following command to list three-layered stacks

On Linux:

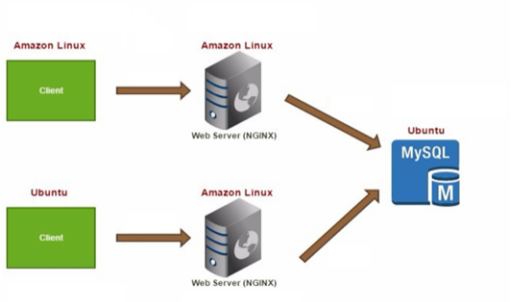
```
three_layered_stacks
```

On Windows:

```
python C:\python27\Scripts\three_layered_stacks
```

```
+--------------------------+----------------------------+----------------------------+
| Client(ClientProcess)    | NginxServer(NginxSrvProc)  | MySQLServer(MySqlSrvProc)  |
+==========================+============================+============================+
| ip-10-0-1-166(python2.7) | ip-10-0-1-250(nginx / uwsgi) | ip-10-0-0-61(mysqld)       |
+--------------------------+----------------------------+----------------------------+
| ip-10-0-1-31(python2.7)  | ip-10-0-0-109(nginx / uwsgi) | ip-10-0-0-61(mysqld)       |
+--------------------------+----------------------------+----------------------------+
| ip-10-0-1-249(python2.7) | ip-10-0-1-105(nginx / uwsgi) | ip-10-0-1-187(mysqld)      |
+--------------------------+----------------------------+----------------------------+
```

Rows one and two of the command output reveal two clients communicating with two Web servers but only one database. The topology is as follows:



Row three of the command output reveals another three layers stack. The topology is as follows:

Tag and Export Configuration Items

You can tag discovered configuration items. Tags are metadata that help you categorize IT assets in your data center. Tags use a *key*,*value* format. You can also export configuration items and tagged items to a database or an Amazon S3 bucket as a CSV file.

1. Use the following command to tag servers. The output shows if the configuration item was tagged or not.

On Linux:

```
awsdiscovery create-tag --hostname ip-10-0-1-166 --tag
 key=serverType,value=webServer
```

On Windows:

```
python C:\python27\Scripts\awsdiscovery create-tag --hostname
 ip-10-0-1-166 --tag key=serverType,value=webServer
```

2. Use the following command to list tagged servers.

   On Linux:

   ```
   awsdiscovery list-servers --tag key=serverType,value=webserver
   ```

   On Windows:

   ```
   python C:\python27\Scripts\awsdiscovery list-servers --tag
    key=serverType,value=webserver
   ```

   ```
   +--------------+----------------------------+-----------+-------------------+
   | Hostname     | OsName                     | IP        | MAC               |
   +==============+============================+===========+===================+
   | ip-10-0-1-166 | Linux - "Ubuntu 14.04.2 LTS" | 10.0.1.166 | 06:77:95:91:a6:e7 |
   +--------------+----------------------------+-----------+-------------------+
   ```

3. Use the following command to delete a tag. The output shows if the configuration item was untagged or not.

   On Linux:

   ```
   awsdiscovery delete-tag –tag --hostname ip-10-0-1-166 --tag
    key=serverType,value=webServer
   ```

   On Windows:

   ```
   python C:\python27\Scripts\awsdiscovery delete-tag –tag --hostname
    ip-10-0-1-166 --tag key=serverType,value=webServer
   ```

4. Use the following command to export data. The output includes an export ID which you will use to view the export status.

   On Linux:

   ```
   awsdiscovery create-export-task
   ```

   On Windows:

   ```
   python C:\python27\Scripts\awsdiscovery create-export-task
   ```

5. Use the following command to view the export status. The output returns the status of the export and a URL where you can view the data.

   On Linux:

```
awsdiscovery describe-export-tasks export-xxxxxx-xxxx-xxxx-
xxxx-6ed91c5c014e
```

On Windows:

```
python C:\python27\Scripts\awsdiscovery describe-export-tasks export-
xxxxxx-xxxx-xxxx-xxxx-6ed91c5c014e
```

Troubleshooting the Discovery Agent

The following documentation can help you troubleshoot problems with the AWS Application Discovery Service that you might encounter while migrating your on-premises servers to Amazon EC2.

General Troubleshooting Tools

Before you begin troubleshooting specific issues, it helps to familiarize yourself with following basic approaches and tools.

Inspect the Discovery Agent Configuration

You can find and inspect your existing Discovery Agent configuration file for your operating system:

- On Linux: `/opt/aws/aws/agent/etc/agent.cfg`
- On Windows: `%Program Files%AWS%agent.cfg`

Run the Discovery Agent in Offline Mode

To inspect the raw data captured by your Discovery Agent, you can run it in offline mode. While in offline mode, data collected by the agent is logged to a local file instead of being sent to the AWS Application Discovery Service.

Complete the following procedure to enter offline mode.

1. Create a temporary folder, such as `/var/tmp` for Linux or `C:\tmp` for Windows.
2. Edit the file the appropriate file for your operating system:

   - Linux: `/opt/aws/aws/agent/etc/agent.cfg`
   - Windows: `C:\ProgramData\Amazon Web Services\AWS Agent\agent.cfg`

   Add the following lines; in addition to your temporary folder path, supply a file name where indicated.

```
{
```

```
     "MustCollect" : true,
     "Publisher": "File",
     "MsgFile": "temporary path and file",
     "SystemPerformanceMsgs": true,
     "SystemPerformanceUpdateFrequency": 1,
     "SystemPerformanceMsgFrequency": 30,
     "ProcessPerformanceMsgs": true,
     "ProcessPerformanceUpdateFrequency": 1,
     "ProcessPerformanceMsgFrequency": 30,
     "ListeningPortInfo": true,
     "Users": true,
     "Groups": true,
     "NetworkInterfaces": true,
     "Terminals": true,
     "PackageInfo": true,
     "InstanceMetaData": true,
     "NetworkConnections": true,
     "ListeningPorts" : true,
     "Processes": true,
     "CodeModules": true,
     "KernelModules": true,
     "DnsEntries": true,
     "Subscribers": "discovery"
}
```

This configuration causes the agent to write events to a location specified by `MsgFile`. The metrics that the agent collects are also specified. (In normal operation, the AWS endpoint would supply this information.)

3. Stop and restart the agent using the command appropriate for your operating system:
   - For Linux, `sudo /etc/init.d/awsagent stop` and `sudo /etc/init.d/awsagent start`
   - For Windows, `sc stop awsagent` and `sc start awsagent`
4. To inspect what events have been emitted by the agent, open the log file that you designated.

Enable Logging

Add the following lines to `agent.cfg` to activate debugging:

```
{
  "SubSystems" : "ALL",
  "LogLevels" : "LogAll",
  "LogFile" : "c:\\tmp\\agent.log"
}
```

As in the previous procedure, stop and restart the agent for the change to take effect.

Troubleshooting the Discovery Agent Data

The following actions can assist with troubleshooting Discovery Agent data.

- Although AWS does not currently support console-based data visualization for the Discovery Agent, you can manually convert the CSV format of discovery data to GraphML format and view it offline with any open-source visualization tool that can consume GraphML format.

- Download sample agent data in CSV format.
- Contact AWS Support to have obsolete discovery data purged from the repository.

Troubleshooting Discovery Agent Configuration

Use the following information to diagnose or repair errors with Discovery Agent configuration.

Discovery Agent Fails to Register

If you encounter the error "Registration failure reason: Authentication failure: Authentication failed as the incoming request was not signed by AWS credentials", try the following approaches:

- If you are using Linux, confirm that the `.aws/credentials` file is located in the root user's home directory as described in On-Premises Installation for Linux. The file must not be in some other user's home directory.
- If you are using Windows, confirm that the AWS credentials are properly installed, as described in On-Premises Installation for Microsoft Windows. Check that the permissions provided for the AWS credentials conform to the appropriate managed policy, as described in Attach Required IAM User Policies.
- Run the following:
  - Linux: /opt/aws/awsagent/etc/agent.cfg
  - Windows: C:\ProgramData\Amazon Web Services\AWS Agent .\AWSAgentStatus.exe
- Check the region in the output. Agents should be configured to communicate with the endpoint of ADS in the `us-west-2` region.
- Check the time skew from your NTP servers and correct if necessary. Incorrect time skew causes the agent registration call to fail.
- Check that you are meeting all Prerequisites, including OS support. If you attempt to install the inspector agent package on Ubuntu and the operation fails with message containing "Failed to find an inspector agent package for this OS...", contact AWS Support.
- If you use Agentless Discovery and don't see inventory information after starting data collection with the connector, confirm that you have registered the connector with your vCenter Server instance. Agentless Discovery does not support a standalone ESX host that is not part of the vCenter Server instance.
- If you are using Windows 2008 R2, confirm that the security patches are up to date.

Failure to Create Discovery Agent Credentials for Windows Server

If you try to install the Discovery Agent on Windows Server but cannot find or create the credentials folder at `%SystemRoot%\system32\config\systemprofile\.aws`, you need to create the folder using the command line. If you still cannot create the `.aws` folder, try appending a period to the folder name (that is, `.aws.`).

Document History for AWS Application Discovery Service

The following table describes the documentation for this release of Application Discovery Service.

Latest documentation update: April, 15, 2016

Change	Description	Date
Initial publication	Released *AWS Application Discovery Service User Guide*.	May 12, 2016
Updates	Added Windows Server details to AWS Application Discovery Service Walkthrough (p. 25). Fixed various command issues.	May 20, 2016
Launch of agentless discovery	Added content that describes how to setup and configure agentless discovery.	July 28, 2016

AWS Glossary

For the latest AWS terminology, see the AWS Glossary in the *AWS General Reference*.