AWS Health

User Guide

AWS Health: User Guide

Table of Contents

What Is AWS Health?

AWS Health provides ongoing visibility into the state of your AWS resources, services, and accounts. The service gives you awareness and remediation guidance for resource performance or availability issues that affect your applications running on AWS. AWS Health provides relevant and timely information to help you manage events in progress. AWS Health also helps to be aware of and to prepare for planned activities. The service delivers alerts and notifications triggered by changes in the health of AWS resources, so that you get near-instant event visibility and guidance to help accelerate troubleshooting.

All customers can use the Personal Health Dashboard (PHD), powered by the AWS Health API. The dashboard requires no setup, and it's ready to use for authenticated AWS users (p. 10). For more service highlights, see the AWS Personal Health Dashboard detail page.

Additionally, AWS Support customers who have a Business or Enterprise support plan can use the AWS Health API to integrate with in-house and third-party systems.

Topics

Are You a First-Time AWS Health User?

If you are a first-time user of AWS Health, begin by reading the following sections:

- What Is AWS Health—The rest of this section describes the underlying data model, the operations it supports, and the AWS SDKs that you can use to interact with the service.
- Getting Started with the AWS Personal Health Dashboard (p. 4)—The Personal Health Dashboard section describes using the Personal Health Dashboard to view events and affected entities and perform advanced filtering.
- Getting Started with the AWS Health API (p. 6)—The AWS Health API section describes the operations that retrieve information about events and entities.

AWS Health provides a console, called the Personal Health Dashboard, to all customers. You do not need to write code or perform any actions to set up the dashboard. If you have a Business or Enterprise support plan, you can access the information presented on the dashboard programmatically. You can use the AWS Command Line Interface (AWS CLI) or write code to make requests, by using either the REST API directly or the AWS SDKs.

For more information about using AWS Health with the AWS CLI, see the AWS CLI Reference for AWS Health. For instructions for installing the AWS CLI, see Installing the AWS Command Line Interface Interface.

Supported Operations in AWS Health

AWS Health supports the following operations for getting information about events that affect an AWS account:

- The event types supported by AWS Health.

- Information about one or more events that match specified filter criteria.
- Information about the entities that are affected by one or more events.
- Categorized counts of events or entities that match specified filter criteria.

All operations are non-mutating. That is, they retrieve data but do not modify it. The following sections summarize the AWS Health operations:

Event Types

The DescribeEventTypes operation retrieves event types that match the optional specified filter. An event type is a template definition of an event's AWS service, event type code, and category. An event type and event are similar to a class and object in object-oriented programming. The number of event types supported by AWS Health will grow over time.

Events

The DescribeEvents operation retrieves summary information about events that are related to an AWS account. The events can be related to AWS operational issues, scheduled changes to AWS infrastructure, or security and billing notifications. The DescribeEventDetails operation retrieves detailed information about one or more events, such as the AWS service, Region, Availability Zone, event start and end times, and a text description.

Affected Entities

The DescribeAffectedEntities operation retrieves information about entities that are affected by one or more events. The results can be filtered by additional criteria, such as status, that might be assigned to AWS resources.

Aggregation

The DescribeEventAggregates operation retrieves a count of the events in each event type category, optionally filtered by other criteria. The DescribeEntityAggregates operation retrieves a count of the entities (resources) that are affected by one or more specified events.

For more information about these operations, see the AWS Health API Reference.

Accessing the AWS Health API

AWS Health is a RESTful web service that uses HTTPS as a transport and JavaScript Object Notation (JSON) as a message serialization format. Your application code can make requests directly to the AWS Health web service API. When using the REST API directly, you must write the necessary code to sign and authenticate your requests. For more information about the API, see the AWS Health API Reference.

> **Note**
> You must have a Business or Enterprise support plan to use the AWS Health API.

You can simplify application development by using the AWS SDKs that wrap the AWS Health REST API calls. You provide your credentials, and these libraries take care of authentication and request signing.

AWS Health also provides a Personal Health Dashboard in the AWS Management Console that is available to all AWS customers. You can use the Personal Health Dashboard to view and search for events and affected entities.

Endpoints

AWS Health has a single global endpoint: https://health.us-east-1.amazonaws.com

You can view events in specific AWS Regions by using the filters supported by the API. For more information about AWS endpoints and Regions for all services, see AWS Regions and Endpoints in the *AWS General Reference*.

Signing AWS Health API Requests

Requests must be signed using an access key ID and a secret access key. We strongly recommend that you do not use your AWS root account credentials for regular access to AWS Health. You can use the credentials for an IAM user.

To sign your API requests, we recommend using AWS Signature Version 4. For more information, see Signing AWS API Requests in the *AWS General Reference*.

Getting Started with the AWS Personal Health Dashboard

The AWS Personal Health Dashboard provides information about AWS Health events that can affect your account. The information is presented in two ways: a dashboard that shows recent and upcoming events organized by category, and a full event log that shows all events from the past 90 days.

Dashboard Landing Page

The Personal Health Dashboard organizes issues in three groups: open issues, scheduled changes, and other notifications. By default, the open issues and other notifications are restricted to issues whose start time is within the last seven days. The scheduled changes group contains items that are ongoing or upcoming.

When you select an event in the dashboard list, the Event Details pane appears with information about the event and resources that are affected by the event. For more information, see Event Details Pane (p. 4).

You can filter the items that appear in any group by choosing options from the filter list. For example, you can narrow the results by Availability Zone, Region, event end time or last update time, AWS service, and others.

To see all the events that apply to your account, rather than the recent ones that appear in the dashboard, choose the **See all issues** link above the list to display the Event Log (p. 4).

To be notified or take an automatic action when events change, you can set up rules by using Amazon CloudWatch Events. Choose the **Set up notifications with CloudWatch Events** link to go to the CloudWatch Events console. For more information, see Monitoring AWS Health Events with Amazon CloudWatch Events (p. 13).

Event Log

The Event log page of the Personal Health Dashboard displays all the AWS Health events that apply to your account. The column layout and behavior is similar to the Dashboard, but the log page includes additional columns and filter options for **Status** and **Event category**.

When you select an event in the Event log list, the Event Details pane appears with information about the event and resources that are affected by the event. For more information, see Event Details Pane (p. 4).

You can filter the items by choosing options from the filter list. For example, you can narrow the results by status (closed, open, or upcoming), event category (issue, notification, or scheduled change), Availability Zone, Region, event end time or last update time, AWS service, and others.

Event Details Pane

The Event details pane has two tabs. The **Details** tab displays a text description of the event and relevant data about the event: the event name, status, Region and Availability Zone, start time, end time, and

category. The **Affected resources** tab displays information about any AWS resources that are affected by the event:

- The resource ID (for example, an Amazon EBS volume ID such as `vol-a1b2c34f`) or Amazon Resource Name (ARN), if available or relevant.

You can filter the items that appear in the resources list by choosing options from the filter list. You can narrow the results by resource ID or ARN.

Alerts for AWS Health Events

The Personal Health Dashboard includes a bell icon in the console navigation bar with an Alerts drop-down menu, which can display the number of recent AWS Health events that appear on the dashboard in each category. This bell icon appears on several AWS consoles, including Amazon EC2, Amazon RDS, IAM, and AWS Trusted Advisor. You can use this feature to see at a glance whether your account is affected by any recent events, and you can choose one of the menu items to go directly to the Personal Health Dashboard for more information.

Getting Started with the AWS Health API

The AWS Health API provides programmatic access to the AWS Health information that is presented in the Personal Health Dashboard. You can use these API operations to get information about events that affect your AWS resources:

- `DescribeEvents`: Summary information about events.
- `DescribeEventDetails`: Detailed information about one or more events.
- `DescribeAffectedEntities`: Information about AWS resources that are affected by one or more events.

In addition, these operations provide information about event types and summary counts of events or affected entities:

- `DescribeEventTypes`: Information about the kinds of events that AWS Health tracks.
- `DescribeEventAggregates`: A count of the number of events that meet specified criteria.
- `DescribeEntityAggregates`: A count of the number of affected entities that meet specified criteria.

The Health API requires a Business or Enterprise support plan from AWS Support. Calling the Health API from an account that does not have a Business or Enterprise support plan causes a `SubscriptionRequiredException`.

For detailed descriptions of the AWS Health API, see the AWS Health API Reference.

Service Endpoint

The endpoint for the AWS Health API is:

- https://health.us-east-1.amazonaws.com

The AWS Health service has only one endpoint, in the US East (N. Virginia) Region, that provides event data for all regions. You can filter query results to get information about events that affect one or more regions, but the query should always be addressed to the health.us-east-1 endpoint.

Java Code Examples

The following section demonstrates using the AWS Health API with the AWS SDK for Java.

Sample Java Code for the AWS Health API

The following Java code examples demonstrate how to initialize an AWS Health client and retrieve information about events and entities.

Step 1: Initialize Credentials

Valid credentials are required to communicate with the AWS Health API. You can use the key pair of any IAM user associated with the AWS account.

Create and initialize an AWSCredentials instance:

```
AWSCredentials credentials = null;
try {
       credentials = new ProfileCredentialsProvider("default").getCredentials();
} catch (Exception e) {
throw new AmazonClientException(
   "Cannot load the credentials from the credential profiles file. "
   + "Please make sure that your credentials file is at the correct "
   + "location (/home/username/.aws/credentials), and is in valid format.", e);
}
```

Step 2: Initialize an AWS Health API Client

Use the initialized credentials object from the previous step to create an AWS Health client:

```
import com.amazonaws.services.health.AWSHealthClient;

AWSHealth awsHealthClient = new AWSHealthClient(credentials);
```

Step 3: Use AWS Health API Operations to Get Event Information

DescribeEvents

```
import com.amazonaws.services.health.model.DescribeEventsRequest;
import com.amazonaws.services.health.model.DescribeEventsResult;
import com.amazonaws.services.health.model.Event;
import com.amazonaws.services.health.model.EventFilter;

DescribeEventsRequest request = new DescribeEventsRequest();

EventFilter filter = new EventFilter();
// Filter on any field from the supported AWS Health EventFilter model.
// Here is an example for region us-east-1 events from the EC2 service.
filter.setServices(singletonList("EC2");
filter.setRegions(singletonList("us-east-1"));
request.setFilter(filter);

DescribeEventsResult response = awsHealthClient.describeEvents(request);
List<Event> resultEvents = response.getEvents();

Event currentEvent = null;
for (Event event : resultEvents) {
    // Display result event data; here is a subset.
    System.out.println(event.getArn());
    System.out.println(event.getService());
    System.out.println(event.getRegion());
    System.out.println(event.getAvailabilityZone());
    System.out.println(event.getStartTime());
    System.out.println(event.getEndTime());
 }
```

DescribeEventAggregates

```
import com.amazonaws.services.health.model.DescribeEventAggregatesRequest;
import com.amazonaws.services.health.model.DescribeEventAggregatesResult;
import com.amazonaws.services.health.model.EventAggregate;
```

```
import com.amazonaws.services.health.model.EventFilter;

DescribeEventAggregatesRequest request = new DescribeEventAggregatesRequest();
// set the aggregation field
request.setAggregateField("eventTypeCategory");

// filter more on result if needed
EventFilter filter = new EventFilter();
filter.setRegions(singleton("us-east-1"));
request.setFilter(filter);

DescribeEventAggregatesResult response = awsHealthClient.describeEventAggregates(request);

// print event count for each eventTypeCategory
for (EventAggregate aggregate: response.getEventAggregates()) {
    System.out.println("Event Category:" + aggregate.getAggregateValue());
    System.out.println("Event Count:" + aggregate.getCount());
 }
```

DescribeEventDetails

```
import com.amazonaws.services.health.model.DescribeEventDetailsRequest;
import com.amazonaws.services.health.model.DescribeEventDetailsResult;
import com.amazonaws.services.health.model.Event;
import com.amazonaws.services.health.model.EventDetails;


DescribeEventDetailsRequest describeEventDetailsRequest = new
 DescribeEventDetailsRequest();
// set event ARN and local value

describeEventDetailsRequest.setEventArns(singletonList("arn:aws:health:us-
east-1::event/service/eventTypeCode/eventId"));
describeEventDetailsRequest.setLocale("en-US");
filter.setEventArns
DescribeEventDetailsResult describeEventDetailsResult =
 awsHealthClient.describeEventDetails(request);
EventDetails eventDetail = describeEventDetailsResult.getSuccessfulSet().get(0);

// check event-related fields
Event event = eventDetail.getEvent();
System.out.println(event.getService());
System.out.println(event.getRegion());
System.out.println(event.getAvailabilityZone());
System.out.println(event.getStartTime());
System.out.println(event.getEndTime());

// print out event description
System.out.println(eventDetail.getEventDescription().getLatestDescription());
```

DescribeAffectedEntities

```
import com.amazonaws.services.health.model.AffectedEntity;
import com.amazonaws.services.health.model.DateTimeRange;
import com.amazonaws.services.health.model.DescribeAffectedEntitiesRequest;
import
 com.amdescribeEventDetailsRequestazonaws.services.health.model.DescribeAffectedEntitiesResult;

DescribeAffectedEntitiesRequest request = new DescribeAffectedEntitiesRequest();
EntityFilter filter = new EntityFilter();

filter.setEventArns(singletonList("arn:aws:health:us-
east-1::event/service/eventTypeCode/eventId"));
```

```
DescribeAffectedEntitiesResult response =
 awsHealthClient.describeAffectedEntities(request);

for (AffectedEntity affectedEntity: response.getEntities()) {
    System.out.println(affectedEntity.getEntityValue());
    System.out.println(affectedEntity.getAwsAccountId());
    System.out.println(affectedEntity.getEntityArn());
 }
```

DescribeEntityAggregates

```
import com.amazonaws.services.health.model.DescribeEntityAggregatesRequest;
import com.amazonaws.services.health.model.DescribeEntityAggregatesResult;
import com.amazonaws.services.health.model.EntityAggregate;

DescribeEntityAggregatesRequest request = new DescribeEntityAggregatesRequest();

request.setEventArns(singletonList("arn:aws:health:us-
east-1::event/service/eventTypeCode/eventId"));

DescribeEntityAggregatesResult response =
 awsHealthClient.describeEntityAggregates(request);

for (EntityAggregate entityAggregate : response.getEntityAggregates()) {
    System.out.println(entityAggregate.getEventArn());
    System.out.println(entityAggregate.getCount());
 }
```

Controlling Access to the AWS Personal Health Dashboard and AWS Health

You can use IAM to create identities (users, groups, or roles), and then give those identities permissions to access the Personal Health Dashboard and AWS Health API.

By default, IAM users do not have access to the Personal Health Dashboard or AWS Health. You give users access to your account's AWS Health information by attaching IAM policies to a single user, a group of users, or a role. For more information, see Identities (Users, Groups, and Roles) and Overview of IAM Policies.

After you create IAM users, you can give those users individual passwords. Then, they can sign in to your account and view AWS Health information by using an account-specific sign-in page. For more information, see How Users Sign In to Your Account.

> **Important**
> An IAM user with permissions to view Personal Health Dashboard has read-only access to health information across all AWS services on the account, which can include, but is not limited to, AWS resource IDs such as Amazon EC2 instance IDs, EC2 instance IP addresses, and general security notifications. For example, if an IAM policy grants access only to Personal Health Dashboard and AWS Health API, then the user or role that the policy applies to can access all information posted about AWS services and related resources, even if other IAM policies do not allow that access.

To allow access to the Personal Health Dashboard and AWS Health, set the `Action` element of an IAM policy to `health:Describe*`. AWS Health supports access control to Events based on the `eventTypeCode` and service. See Resource- and Action-based Conditions (p. 11). To allow access to all events, set the `Resource` element to `*`.

> **Note**
> Although the Personal Health Dashboard is available for all AWS accounts, the AWS Health API is available only to accounts with a Business or Enterprise support plan. For more information, see AWS Support.

For example, this policy statement grants access to Personal Health Dashboard and AWS Health:

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

This policy statement denies access to Personal Health Dashboard and AWS Health:

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
  {
    "Effect": "Deny",
    "Action": [
      "health:*"
    ],
    "Resource": "*"
  }]
}
```

If the user or group that you want to give permissions to already has a policy, you can add the AWS Health-specific policy statement illustrated here to that policy.

Resource- and Action-based Conditions

AWS Health supports IAM conditions for `DescribeAffectedEntities` and `DescribeEventDetails`. This allows you to restrict Events vended by AWS Health API on a per user, group, or role basis. You can achieve this by populating the conditions block of the IAM Policy or by setting the `Resource` element. You can use String Conditions to restrict access based on certain Health Event fields. The following fields are supported:

- `eventTypeCode`
- `service`

For example, this policy statement grants access to Personal Health Dashboard and AWS Health, but denies access to any Health Events relating to EC2:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "health:Describe*",
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": [
                "health:DescribeAffectedEntities",
                "health:DescribeEventDetails"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "health:service": "EC2"
                }
            }
        }
    ]
}
```

The following policy has the same effect, but makes use of the `Resource` element:

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

```
    "Effect": "Allow",
    "Action": [
      "health:Describe*"
    ],
    "Resource": "*",
  },
  {
    "Effect": "Deny",
    "Action": [
      "health:DescribeEventDetails",
      "health:DescribeAffectedEntities"
    ],
    "Resource": "arn:aws:health:*::event/EC2/*/*",
  }]
}
```

This policy statement grants access to Personal Health Dashboard and AWS Health, but denies access to any Health Events with type `AWS_EC2_*`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "health:Describe*",
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": [
                "health:DescribeAffectedEntities",
                "health:DescribeEventDetails"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "health:eventTypeCode": "AWS_EC2_*"
                }
            }
        }
    ]
}
```

Important
An **AccessDeniedException** will be produced when a user attempts to access an event that is denied by the associated user, group, or role. This applies only to `DescribeAffectedEntities` and `DescribeEventDetails`.

Monitoring AWS Health Events with Amazon CloudWatch Events

You can use Amazon CloudWatch Events to detect and react to changes in the status of AWS Personal Health Dashboard (AWS Health) events. Then, based on the rules that you create, CloudWatch Events invokes one or more target actions when an event matches the values that you specify in a rule. Depending on the type of event, you can send notifications, capture event information, take corrective action, initiate events, or take other actions. You can select the following types of targets when using CloudWatch Events as a part of your AWS Health workflow:

- AWS Lambda functions
- Kinesis streams
- Amazon SQS queues
- Built-in targets (CloudWatch alarm actions)
- Amazon SNS topics

The following are some use cases:

- Use a Lambda function to pass a notification to a Slack channel when an event occurs.
- Send custom text or SMS notifications with Amazon SNS when an AWS Health event happens by using Lambda and CloudWatch Events.

For samples of automation and customized alerts that you can create in response to AWS Health events, see the AWS Health Tools in GitHub.

> **Note**
> Only those AWS Health events that are specific to your AWS account and resources are published to CloudWatch Events. This includes events such as EBS volume lost, EC2 instance store drive performance degraded, and all the scheduled change events. In contrast, Service Health Dashboard events provide information about the regional availability of a service and are not specific to AWS accounts, so they are not published to CloudWatch Events. These event types have the word "operational" in the title in the Personal Health Dashboard; for example, "SWF operational issue".

The remainder of this topic describes the basic procedure for creating a CloudWatch Events rule for AWS Health. Before you create event rules for AWS Health, however, you should do the following:

- Familiarize yourself with events, rules, and targets in CloudWatch Events. For more information, see What Is Amazon CloudWatch Events? and New CloudWatch Events – Track and Respond to Changes to Your AWS Resources.
- Create the target or targets to use in your event rules.

To create a CloudWatch Events rule for AWS Health:

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Events**.
3. Choose **Create rule**, and then under **Event Source**, for **Service Name**, choose **Health**.

4. Specify AWS services:
   - To make a rule that applies to all AWS services, for **Event Type**, choose **All Events**. If you choose all events, you cannot choose event type categories or event type codes.
   - To make a rule that applies to events for one service only, choose **Specific Health events**, choose **Specific service(s)**, and then choose a service name from the list. For example, **EC2**. Note: you cannot choose more than one service.
5. Specify event type categories (if you have selected a specific service):
   - To make a rule that applies to all event type categories, choose **Any event type category**.
   - To make a rule that applies to one event type category only, choose **Specific event type category(s)**, and then choose a value from the list. For example, **scheduledChange**. Note: you can't choose more than one category.
6. Specify event type codes (if you have selected a specific service and a specific event type category):
   - To make a rule that applies to all event type codes, choose **Any event type code**.
   - To make a rule that applies to one or more event type codes only, choose **Specific event type code(s)**, and then choose one or more values from the list. For example, **AWS_EC2_PERSISTENT_INSTANCE_RETIREMENT_SCHEDULED**.
7. Specify affected resources:
   - To make a rule that applies to all resources, choose **Any resource**.
   - To make a rule that applies to one or more resources only, choose **Specific resource(s)**, and then type the IDs of one or more resources. For example, **`i-a1b2c34f`**.
8. Review your rule setup to be sure it meets your event-monitoring requirements.
9. In the **Targets** area, choose **Add target***.
10. In the **Select target type** list, choose the type of target you have prepared to use with this rule, and then configure any additional options required by that type.
11. Choose **Configure details**.
12. On the **Configure rule details** page, type a name and description for the rule, and then choose the **State** box to enable the rule as soon as it is created.
13. If you're satisfied with the rule, choose **Create rule**.

Automating Actions for EC2 Instances

You can automate actions in response to new scheduled events for your EC2 instances. For example, you can create CloudWatch Events rules for EC2 scheduled events generated by the AWS Health service. These rules can then trigger targets, such as AWS Systems Manager Automation documents, to automate actions. You can find more options at Automating Amazon EC2 with CloudWatch Events.

For example, when an Amazon EC2 instance retirement event is scheduled for an EBS-backed EC2 instance, you can automate the stop and start of the instance so that you don't have to perform these actions manually.

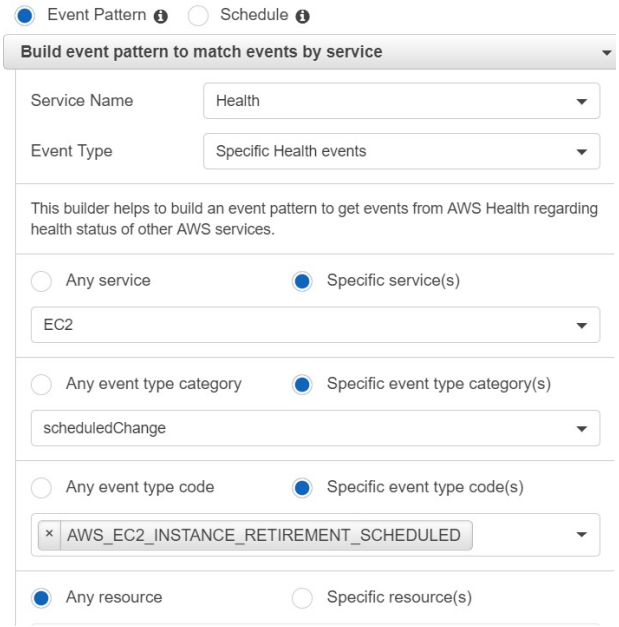
Automate the stop and start of EBS-backed EC2 instances that are scheduled for retirement

1. Open the CloudWatch console to create a CloudWatch Events rule:
   - Go to the https://console.aws.amazon.com/cloudwatch/.
   - Under **Events**, choose **Rules**.
2. Edit the Event Pattern Preview, and then insert the following inputs:

Example

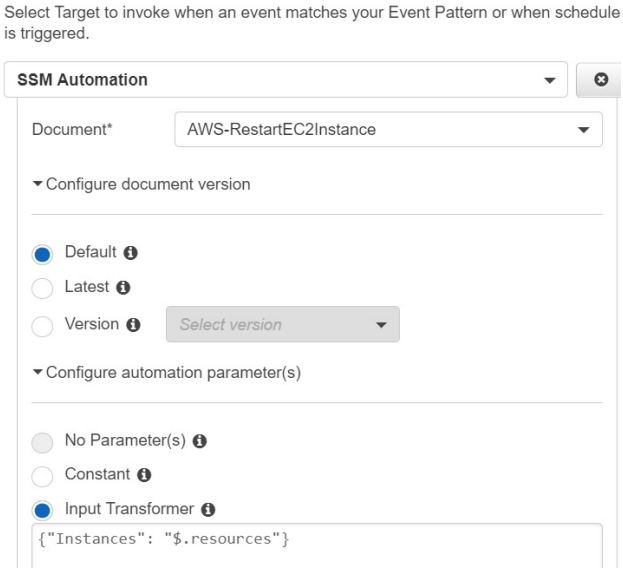
```
{
  "source": [
    "aws.health"
  ],
  "detail-type": [
    "AWS Health Event"
  ],
  "detail": {
    "service": [
      "EC2"
    ],
    "eventTypeCategory": [
      "scheduledChange"
    ],
    "eventTypeCode": [
      "AWS_EC2_INSTANCE_RETIREMENT_SCHEDULED"
    ]
  }
}
```

You should see the following fields populate in the console:



3. Select **Save**.
4. Add the Systems Manager Automation document target by selecting **Add target***, and then selecting **SSM Automation**, as shown in this figure:

Example



5. Choose the **AWS-RestartEC2Instance** Systems Manager document from the list.
6. Configure the Input Transformer as shown here, with `{"Instances":"$.resources"}` as the InputPathsMap and `{"InstanceId": <Instances>}` as the **Input Template**.
7. Choose an existing IAM role, and then create a new one with permissions to execute the SSM automation document.

If you don't have an existing IAM role with required EC2 and Systems Manager permissions, then create one

1. Set up the required IAM permissions for CloudWatch Events to use by creating an IAM policy. Then, associate it with an IAM role for CloudWatch. For this example name the IAM role `AutomationCWRole`. Here is an example of such an IAM policy:

   ### Example

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:StartInstances",
        "ec2:StopInstances",
        "ec2:DescribeInstanceStatus"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ssm:*"
      ],
      "Resource": [
        "*"
```

```
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": [
        "arn:aws:sns:*:*:Automation*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::<AccountId>:role/AutomationCWRole"
    }
  ]
}
```

2. Be sure to update the role ARN with the account ID and role name. Also, be sure that the role has **events.amazonaws.com** and **ssm.amazonaws.com** configured as a trusted entity for the IAM role as shown here:

   Example

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

Logging AWS Health API Calls with AWS CloudTrail

AWS Health is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS Health. CloudTrail captures API calls for AWS Health as events. The calls captured include calls from the AWS Health console and code calls to the AWS Health API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS Health. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to AWS Health, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, including how to configure and enable it, see the AWS CloudTrail User Guide.

AWS Health Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When supported event activity occurs in AWS Health, 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 AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for AWS Health, create a *trail*. A trail enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for Creating a Trail
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

All AWS Health actions *except* `DescribeEventAggregates` are logged by CloudTrail and are documented in the AWS Health API Reference. For example, calls to the `DescribeEvents`, `DescribeEventDetails`, and `DescribeAffectedEntities` actions generate entries in the CloudTrail log files.

AWS Health supports logging the following actions as events in CloudTrail log files:

- Whether the request was made with root or IAM credentials
- Whether the request was made with temporary security credentials for a role or federated user
- Whether the request was made by another AWS service

For more information, see the CloudTrail userIdentity Element.

You can store your log files in your Amazon S3 bucket for as long as you want. You can also define Amazon S3 lifecycle rules to archive or delete log files automatically. By default, your log files are encrypted with Amazon S3 server-side encryption (SSE).

To be notified upon log file delivery, you can configure CloudTrail to publish Amazon SNS notifications when new log files are delivered. For more information, see Configuring Amazon SNS Notifications for CloudTrail.

You can also aggregate AWS Health log files from multiple AWS regions and multiple AWS accounts into a single Amazon S3 bucket.

For more information, see Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts.

Example: AWS Health Log File Entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the DescribeEntityAggregates action.


```
{
   "Records": [
   {
   "eventVersion": "1.05",
   "userIdentity": {
      "type": "IAMUser",
      "principalId": "AIDACKCEVSQ6C2EXAMPLE",
      "arn": "arn:aws:iam::111122223333:user/JaneDoe",
      "accountId": "111122223333",
      "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
      "userName": "JaneDoe",
      "sessionContext": {"attributes": {
         "mfaAuthenticated": "false",
         "creationDate": "2016-11-21T07:06:15Z"
      }},
      "invokedBy": "AWS Internal"
    },
   "eventTime": "2016-11-21T07:06:28Z",
   "eventSource": "health.amazonaws.com",
   "eventName": "DescribeEntityAggregates",
   "awsRegion": "us-east-1",
   "sourceIPAddress": "203.0.113.0",
   "userAgent": "AWS Internal",
   "requestParameters": {"eventArns": ["arn:aws:health:us-east-1::event/EBS/
EBS_LOST_VOLUME/EBS_LOST_VOLUME_123"]},
   "responseElements": null,
   "requestID": "05b299bc-afb9-11e6-8ef4-c34387f40bd4",
   "eventID": "e4deb9dc-dbc2-4bdb-8515-73e8abcbc29b",
   "eventType": "AwsApiCall",
   "recipientAccountId": "111122223333"
   }
   ],
   ...
}
```

Document History for AWS Health

The following table describes the documentation for this release of AWS Health.

- **API version:** 2016-08-04
- **Latest documentation update:** August 2, 2018

Change	Description	Date Changed
Added new section "Resource- and Action-based Conditions" to explain Events restrictions vended by the AWS Health API.	See Controlling Access to the AWS Personal Health Dashboard and AWS Health (p. 10).	August 2, 2018
Added a note about the visibility of AWS Health information.	See Controlling Access to the AWS Personal Health Dashboard and AWS Health (p. 10).	August 16, 2017
Service release.	AWS Health released.	December 1, 2016

AWS Glossary

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