

Lab 186-Monitoring Infrastructure

Lab Overview

The ability to monitor your applications and infrastructure is critical for delivering reliable, consistent IT services.

Monitoring requirements range from collecting statistics for long-term analysis to quickly reacting to changes and outages. Monitoring can also support compliance reporting by continuously checking that infrastructure is meeting organizational standards.

This lab shows you how to use Amazon CloudWatch Metrics, Amazon CloudWatch Logs, Amazon CloudWatch Events, and AWS Config to monitor your applications and infrastructure.

After completing this lab, you will be able to:

- Use the AWS Systems Manager Run Command to install the CloudWatch agent on Amazon Elastic Compute Cloud (Amazon EC2) instances
- Monitor application logs using CloudWatch agent and CloudWatch Logs
- Monitor system metrics using CloudWatch agent and CloudWatch Metrics
- Create real time notifications using CloudWatch Events
- Track infrastructure compliance using AWS Config

Duration

This lab requires approximately **60 minutes** to complete.

Task 1: Installing the CloudWatch agent

You can use the CloudWatch agent to collect metrics from EC2 instances and on-premises servers, including the following:

- **System-level metrics from EC2 instances**, such as CPU allocation, free disk space, and memory utilization. These metrics are collected from the machine itself and complement the standard CloudWatch metrics that CloudWatch collects.
- **System-level metrics from on-premises servers** that enable the monitoring of hybrid environments and servers not managed by AWS.
- **System and application logs** from both Linux and Windows servers.
- **Custom metrics** from applications and services using the [StatsD](#) and [collectd](#) protocols.

In this task, you use Systems Manager to install the CloudWatch agent on an EC2 instance. You configure it to collect both application and system metrics.



5. In the **AWS Management Console**, on the **Services** menu, select **Systems Manager**.
6. In the left navigation pane, choose **Run Command**.
If there is no visible navigation pane, choose the icon in the top-left corner to make it appear.
You will use the Run Command to deploy a pre-written command that installs the CloudWatch agent.
7. Choose **Run a Command**.
8. Select the button next to **AWS-ConfigureAWSPackage** (typically appears toward the top of the list).
9. Scroll to the **Command parameters** section and configure the following information:

- **Action:** Select **Install**.
 - **Name:** Enter `AmazonCloudWatchAgent`
 - **Version:** Enter `latest`
10. In the **Targets** section, select **Choose instances manually**, and then under **Instances**, select the check box next to **Web Server**.
This configuration installs the CloudWatch agent on the web server.
11. At the bottom of the page, choose **Run**
12. Wait for the **Overall status** to change to **Success**. You can occasionally choose refresh toward the top of the page to update the status.
You can view the output from the job to confirm that it ran successfully.
13. Under **Targets and outputs**, choose next to the instance, and then click **View output**.
14. Expand **Step 1 - Output**.
You should see the message **Successfully installed**
`arn:aws:ssm:::package/AmazonCloudWatchAgent`.
If you see the message **Step execution skipped due to unsatisfied preconditions: "'StringEquals': [platformType, Windows]'**. **Step name: createDownloadFolder**, then expand **Step 2 - Output** instead. You can select this option because the instance you are using was created from a Linux AML. You can safely ignore this message.
You now configure the CloudWatch agent to collect the desired log information. The instance has a web server installed, so you configure the CloudWatch agent to collect the web server logs and general system metrics.
You will store the configuration file in AWS Systems Manager Parameter Store, which the CloudWatch agent can then retrieve.
15. In the left navigation pane, choose **Parameter Store**.
16. Choose **Create parameter**, and then configure the following information:
- **Name:** Enter `Monitor-Web-Server`
 - **Description:** Enter `Collect web logs and system metrics`
 - **Value:** Copy and paste the following configuration:

```
17. {
  "logs": {
    "logs_collected": {
      "files": {
        "collect_list": [
          {
            "log_group_name": "HttpAccessLog",
            "file_path": "/var/log/httpd/access_log",
            "log_stream_name": "{instance_id}",
            "timestamp_format": "%b %d %H:%M:%S"
          }
        ]
      }
    }
  }
}
```

```

        {
            "log_group_name": "HttpErrorLog",
            "file_path": "/var/log/httpd/error_log",
            "log_stream_name": "{instance_id}",
            "timestamp_format": "%b %d %H:%M:%S"
        }
    ]
}
},
"metrics": {
    "metrics_collected": {
        "cpu": {
            "measurement": [
                "cpu_usage_idle",
                "cpu_usage_iowait",
                "cpu_usage_user",
                "cpu_usage_system"
            ],
            "metrics_collection_interval": 10,
            "totalcpu": false
        },
        "disk": {
            "measurement": [
                "used_percent",
                "inodes_free"
            ],
            "metrics_collection_interval": 10,
            "resources": [
                "*"
            ]
        },
        "diskio": {
            "measurement": [
                "io_time"
            ],
            "metrics_collection_interval": 10,
            "resources": [
                "*"
            ]
        },
        "mem": {
            "measurement": [
                "mem_used_percent"
            ],

```

```

        "metrics_collection_interval": 10
    },
    "swap": {
        "measurement": [
            "swap_used_percent"
        ],
        "metrics_collection_interval": 10
    }
}
}
}
}

```

18. Examine the above configuration. It defines the following items to be monitored:

- **Logs:** Two web server log files to be collected and sent to CloudWatch Logs
- **Metrics:** CPU, disk, and memory metrics to sent to CloudWatch Metrics

19. Choose **Create parameter**

This parameter will be referenced when starting the CloudWatch agent.

You now use another Run Command to start the CloudWatch agent on the web server.

20. In the left navigation pane, choose **Run Command**.

21. Choose **Run command**

22. Choose the box, and then select the following:

- Select **Document name prefix**.
- Select **Equals**.
- Enter `AmazonCloudWatch-ManageAgent`
- Verify that the filter is **Document name prefix : Equals : AmazonCloudWatch-ManageAgent**
- Press Enter.

23. Before running the command, you can view the definition of the command.

24. Choose **AmazonCloudWatch-ManageAgent** (choose the name itself).

A new web browser tab opens that shows the definition of the command.

Browse through the content of each tab to see how a command document is defined.

25. Choose the **Content** tab, and scroll to the bottom to see the actual script that will run on the target instance.

The script references the AWS Systems Manager Parameter Store because it retrieves the CloudWatch agent configuration that you defined earlier.

26. Close the current web browser tab, which should return you to the **Run a command** tab that you were using earlier.

Verify that you have selected the button next to **AmazonCloudWatch-ManageAgent**.

27. In the **Command parameters** section, configure the following information:

- **Action:** Select **configure**.
- **Mode:** Select **ec2**.
- **Optional Configuration Source:** Select **ssm**.
- **Optional Configuration Location:** Enter `Monitor-Web-Server`
- **Optional Restart:** Select **yes**.

28. This configures the agent to use the configuration you previously stored in the Parameter Store.

29. In the **Targets** section, select **Choose instances manually**.

30. In the **Instances** section, select the check box next to **Web Server**.

31. Choose **Run**

32. Wait for the **Overall status** to change to **Success**. You can occasionally choose refresh toward the top of the page to update the status.

The CloudWatch agent is now running on the instance and sending log and metric data to CloudWatch.

The screenshot displays the AWS Systems Manager console interface. A green banner at the top indicates that the command 'a90cdf82-b128-468e-8a5f-030b7bec83' was successfully sent. The main content area shows the command details, including its ID and a table of targets and outputs. The table shows one target, 'ip-10-0-0-8.us-west-2.compute.internal', with a status of 'Success'. The overall status is also 'Success'. The left sidebar contains navigation links for various AWS Systems Manager features. The bottom of the screen shows the Windows taskbar with the date and time as 06/01/2023.

Command ID: a90cdf82-b128-468e-8a5f-030b7bec83

Command status

Overall status	Detailed status	# targets	# completed	# error	# delivery timed out
Success	Success	1	1	0	0

Targets and outputs

Instance ID	Instance name	Status	Detailed Status	Start time	Finish time
<input type="checkbox"/> i-0c9dd19da572319f7	ip-10-0-0-8.us-west-2.compute.internal	Success	Success	Tue, 06 Jan 2026 10:50:27 GMT	Tue, 06 Jan 2

CloudWatch alarm

Task 2: Monitoring application logs using CloudWatch Logs

You can use CloudWatch Logs to monitor applications and systems using log data. For example, CloudWatch Logs can track the number of errors that occur in your application logs and send you a notification whenever the rate of errors exceeds a threshold that you specify.

CloudWatch Logs uses your existing log data for monitoring, so no code changes are required. For example, you can monitor application logs for specific literal terms (such as "NullPointerException") or count the number of occurrences of a literal term at a particular position in log data (such as 404 status codes in a web server access log). When the term you are searching for is found, CloudWatch Logs reports the data to a CloudWatch metric that you specify. Log data is encrypted while in transit and while it is at rest.

In this task, you generate log data on the Web Server and then monitor the logs using CloudWatch Logs.



The Web Server generates two types of log data:

- Access logs
- Error logs

You begin by accessing the web server.

29. Choose the **Details** dropdown menu above these instructions, and then choose **Show**

Copy the **WebServerIP** value.

30. Open a new web browser tab, paste the **WebServerIP** you copied, and then press Enter.

You should see a web server **Test Page**.

You now generate log data by attempting to access a page that does not exist.

31. Append `/start` to the browser URL, and press Enter.
You receive an error message because the page is not found. This is okay! It generates data in the access logs that are being sent to CloudWatch Logs.
32. Keep this tab open in your web browser, but return to the browser tab showing the AWS Management Console.
33. From the **Services** menu, choose **CloudWatch**.
34. In the left navigation pane, choose **Log groups**.
You should see two logs listed: **HttpAccessLog** and **HttpErrorLog**.
If these logs are not listed, try waiting a minute, and then choose **Refresh**.
35. Choose **HttpAccessLog** (choose the name itself).
36. In the **Logs streams** section, choose the **Log stream** in the table (choose the name itself). It has the same ID as the EC2 instance that the log is attached to. Log data should be displayed, consisting of **GET** requests that were sent to the web server. You can view additional information by choosing to expand the lines. The log data includes information about the computer and the browser that made the request.
You should see a line with your `/start` request with a code of 404, which means that the page was not found.
This demonstrates how log files can be automatically shipped from an EC2 instance or an on-premises server to CloudWatch Logs. The log data is accessible without having to log in to each individual server. Log data can also be collected from multiple servers, such as an Auto Scaling fleet of web servers.

Create a metric filter in CloudWatch Logs

You now configure a filter to identify **404 Errors** in the log file. This error would normally indicate that the web server is generating invalid links that users are choosing.

37. In the left navigation pane, choose **Log groups**.
38. Select the check box next to **HttpAccessLog**.
39. From the **Actions** dropdown menu, select **Create metric filter**.
A filter pattern defines the fields in the log file and filters the data for specific values.
40. Paste the following line into the **Filter pattern** box:
41. `[ip, id, user, timestamp, request, status_code=404, size]`
42. This line tells CloudWatch Logs how to interpret the fields in the log data and defines a filter to find lines only with **status_code=404**, which indicates that a page was not found.

43. In the **Test pattern** section, use the dropdown menu to select the EC2 instance id. It is be similar to **i-0f07ab62aae4xxxx9**.
44. Choose **Test pattern**
45. In the **Results** section, choose **Show test results**.
You should see at least one result with a **\$status_code** of **404**. This status code indicates that a page was requested that was not found.
46. Choose **Next**
47. In the **Create filter name** section, in the **Filter name** box, enter **404Errors**
48. In the **Metric details** section, configure the following information:
 - **Metric namespace:** Enter **LogMetrics**
 - **Metric name:** Enter **404Errors**
 - **Metric value:** Enter **1**
49. Choose **Next**. If **Next** is not enabled, click an empty text field, this will shift focus and enable it.
50. On the **Review and create** page, choose **Create metric filter**
This metric filter can now be used in an alarm.

Create an alarm using the filter

You now configure an alarm to send a notification when too many **404 Not Found** errors are received.

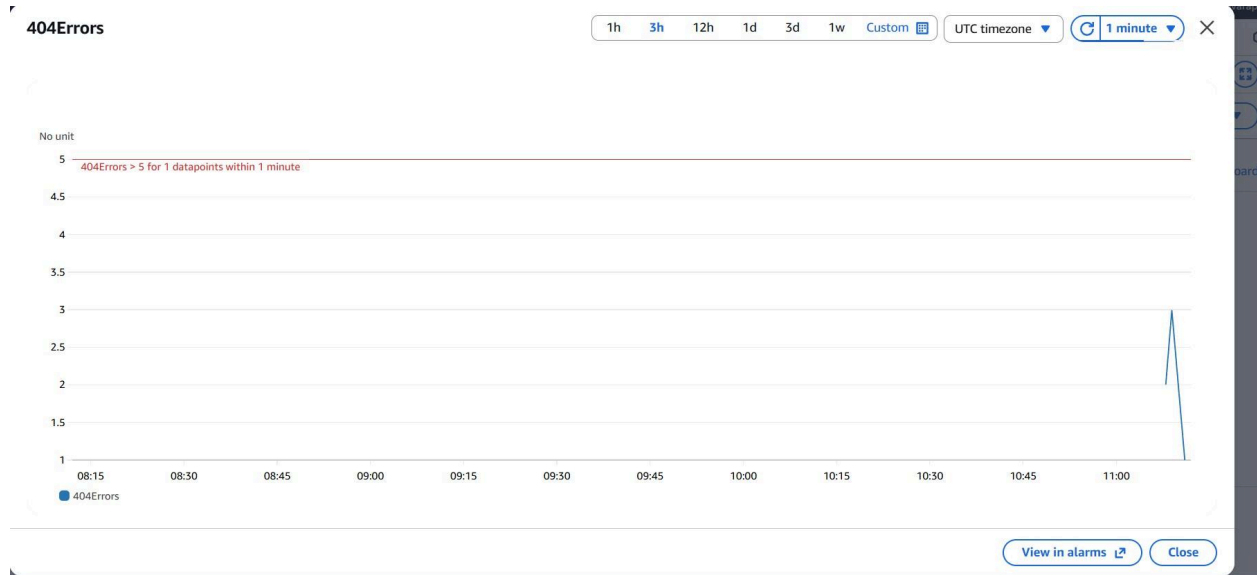
49. In the **404Errors** panel, choose the check box in the top-right corner.
50. In the **Metric filters** section, choose **Create alarm**
51. Configure the following settings:
 - In the **Metrics** section, for **Period**, select **1 minute**.
 - In the **Conditions** section, select the following:
 - **Whenever 404Errors is:** Select **Greater/Equal**
 - **than:** Enter **5**
 - Choose **Next**
52. In the **Notification** section, configure the following:
 - **Select an SNS Topic:** Select **Create new topic**.
 - **Email endpoints that will receive the notification:** Enter an email address that you can access from the classroom.
 - Choose **Create topic**
 - Choose **Next**
53. For **Name and description**, configure the following settings:
 - **Alarm name:** Enter **404 Errors**

- **Alarm description:** Enter `Alert when too many 404s detected on an instance`
 - Choose **Next**
54. Choose **Create alarm**
 55. Go to your email, look for a confirmation message, and select the **Confirm subscription** link.
 56. Return to the AWS Management Console.
 57. In the left navigation pane, choose **CloudWatch** (at the very top).
Your alarm might appear in **orange**, indicating that there is **Insufficient data** to trigger the alarm. This alarm appears because no data has been received in the past minute.
You now access the web server to generate log data.
 58. Return to the web browser tab with the web server.

If the web server browser tab is no longer open, choose the **Details** dropdown menu above these instructions, and then choose **Show**

Copy the **WebServerIP** value, and paste it into a new browser tab.

59. Attempt to go to pages that do not exist by adding a page name after the IP address. Repeat this step at least five times.
For example, enter `http://192.0.2.0/start2`
Each separate request generates a separate log entry.
60. Wait 1-2 minutes for the alarm to trigger. In the AWS Management Console, you can occasionally choose **Refresh** to update the status.
The graph shown on the CloudWatch page should turn **red** to indicate that it is in the **Alarm** state.
61. Check your email. You should have received an email with the subject **ALARM: "404 Errors"**.
This task demonstrates how you can create an alarm from application log data and receive alerts when unusual behavior is detected in the log file. The log file is accessible within CloudWatch Logs to perform further analysis to diagnose the activities that triggered the alarm.



Task 3: Monitoring instance metrics using CloudWatch

Metrics are data about the performance of your systems. CloudWatch stores metrics for the AWS services you use. You can also publish your own application metrics either via the CloudWatch agent or directly from your application. CloudWatch can present the metrics for search, graphs, dashboards, and alarms.

In the task, you use metrics that CloudWatch provides.



62. On the **Services** menu, choose **EC2**.

63. In the left navigation pane, choose **Instances**.

64. Select the check box next to **Web Server**.

65. Choose the **Monitoring** tab in the lower half of the page.

Examine the metrics presented. You can also select a chart to display more information.

CloudWatch captures metrics about CPU, disk, and network usage on the instance. These metrics view the instance from the outside as a virtual machine but do not give insight into what is running inside the instance, such as measuring free memory or free disk space. Fortunately, you can obtain information about what is happening inside the instance by using information that the CloudWatch agent captures because the CloudWatch agent runs inside the instance to collect metrics.

66. From the **Services** menu, select **CloudWatch**.

67. In the left navigation pane, choose **Metrics**. Then expand **Metrics** and select All metrics.

The lower half of the page displays the various metrics that CloudWatch has collected. AWS automatically generates some of these metrics, and the CloudWatch agent collects some others.

68. Choose **CWAgent**, and then choose **device**, **fstype**, **host**, **path**.

You see the disk space metrics that the CloudWatch agent is capturing.

69. Above the table, choose **CWAgent** (in the line that says **All > CWAgent > device, fstype, host, path**).

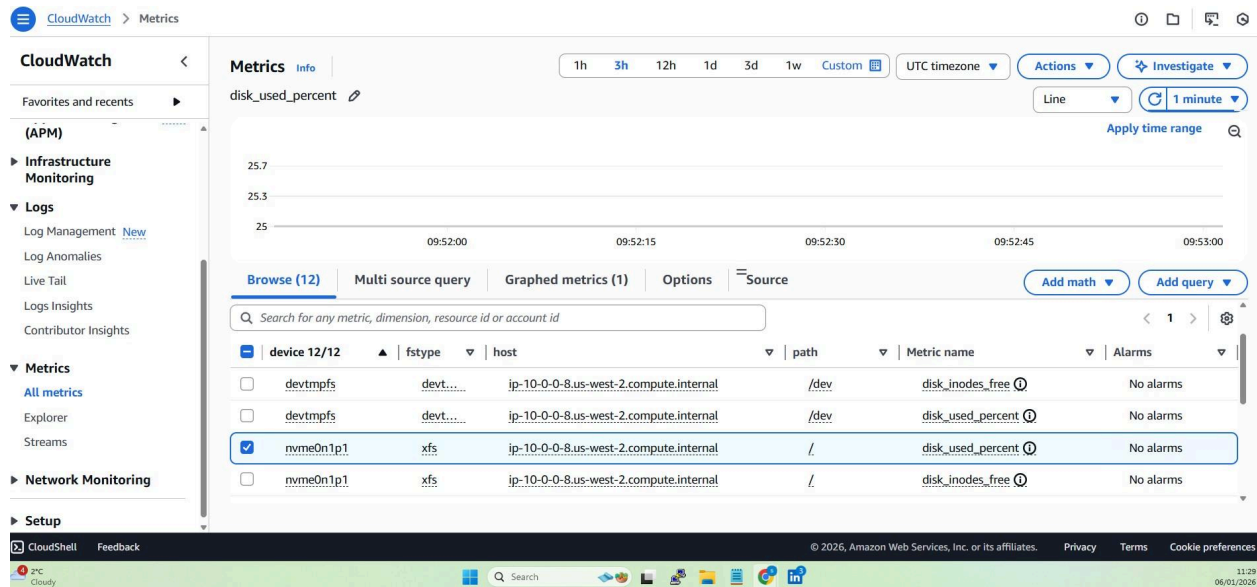
70. Choose **host**.

You see metrics relating to system memory.

71. Above the table, choose **All** (in the line that says **All > CWAgent > device, fstype, host, path**).

Explore the other metrics that CloudWatch is capturing. These are automatically generated metrics coming from the AWS services that have been used in this AWS account.

You can select metrics that you want to appear on the graph.

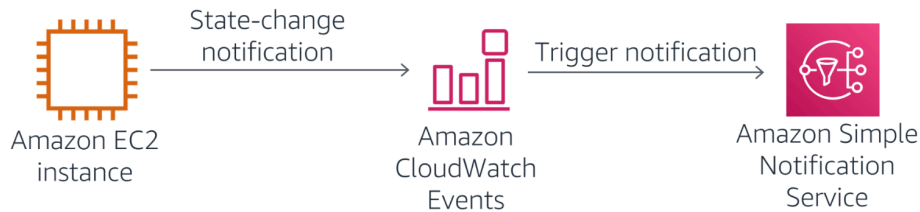


Task 4: Creating real time notifications

CloudWatch Events deliver a near-real-time stream of system events that describe changes in AWS resources. Simple rules can match events and route them to one or more target functions or streams. CloudWatch Events become aware of operational changes as they occur.

CloudWatch Events respond to these operational changes and take corrective action as necessary by sending messages to respond to the environment, activating functions, making changes, and capturing state information. You can also use CloudWatch Events to schedule automated actions that self trigger at certain times using cron or rate expressions.

In this task, you create a real time notification that informs you when an instance is stopped or terminated.



72. On the **CloudWatch** console, in the left navigation pane expand **Events**, choose **Rules**.

73. Choose **Create rule**.

74. In the **Rule definition**, configure the following settings:

- For **Name**, enter `Instance_Stopped_Terminated`
- Choose **Next**

75. In the **Build event pattern** section

- Confirm that **Event source** is *AWS Services*.
- For **AWS service**, choose **EC2**.
- For **Event Type**: Select **EC2 Instance State-change Notification**.
- For **Event Type Specification 1** choose
 - Select the check box for **Specific state(s)**.
 - For **Specific state(s)** , from the dropdown menu, select **stopped** and **terminated**.
- Choose **Next**

76. In the **Select target(s)** section

- For **Target1**:
 - For Target types, choose **AWS Service**
 - For **Select a target**, choose **SNS topic**
 - For **Topic**, select the **Default_CloudWatch_Alarms_Topic**
 - Under **Permissions**, Clear the checkbox *Use execution role (recommended)*.
 - Choose **Next**
 - Choose **Next**

77. At the bottom of the **Review and create** page

- Choose **Create rule**

Configure a real time notification

You can configure Amazon Simple Notification Service (Amazon SNS) to send the real time notifications to your phone via SMS, or to your email address. Because configuring SMS messaging requires opening a ticket with AWS Support as well as time to configure the changes to your account, you will use the same email address you used earlier to complete this exercise.

You can read more about configuring SMS messaging with SNS in the [Amazon Simple Notification Service Developer Guide](#).

78. On the **Services** menu, choose **Simple Notification Service**.

79. In the left navigation pane, choose **Topics**.

80. Choose the link in the **Name** column.

You should see a single subscription associated with your email address. This is the Topic you configured in Task 2.

81. On the **Services** menu, choose **EC2**.

82. In the left navigation pane, choose **Instances**.

83. Select the check box next to **Web Server**.

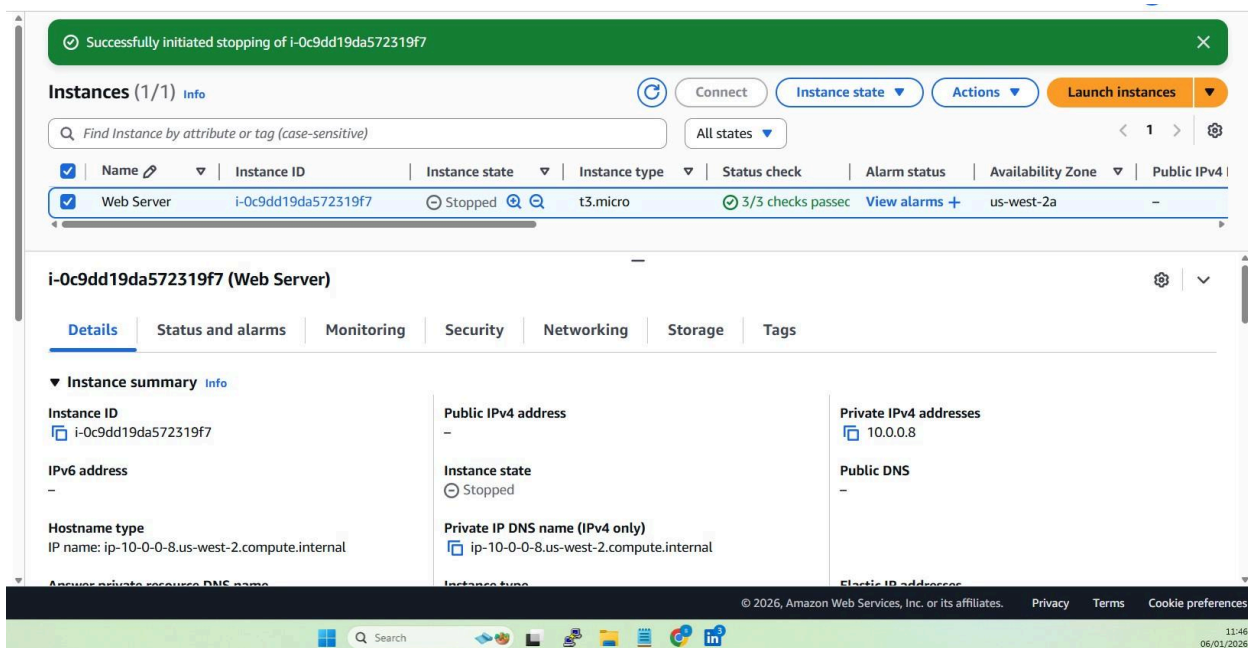
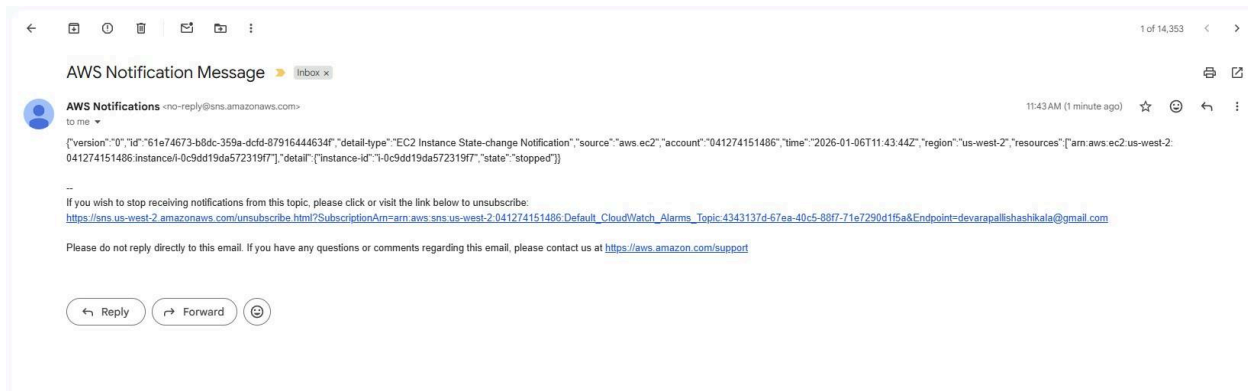
84. Choose **Instance state**, then **Stop instance**, and then **Stop**

The **Web Server** instance enters the **Stopping** state. After a minute, it enters the **Stopped** state.

You should then receive an email with details about the instance that was stopped.

The message is formatted in JSON. To receive a message that is easier to read, you could create an AWS Lambda function that CloudWatch Events triggers. The Lambda function could then format a more readable message and send it via Amazon SNS.

This task demonstrates how to receive real time notifications when infrastructure changes.



Task 5: Monitoring for infrastructure compliance

With AWS Config, you can assess, audit, and evaluate the configurations of your AWS resources. AWS Config continuously monitors and records your AWS resource configurations and allows you to automate the evaluation of recorded configurations against desired configurations.

With AWS Config, you can review changes in configurations and relationships between AWS resources, dive into detailed resource configuration histories, and determine your overall compliance against the configurations specified in your internal guidelines. With

AWS Config, you can simplify compliance auditing, security analysis, change management, and operational troubleshooting.

In this task, you activate AWS Config rules to ensure compliance of tagging and Amazon Elastic Block Store (Amazon EBS) volumes.

85. On the **Services** menu, choose **Config**.

86. If a **Get started** button appears, do the following:

- Choose **Get started**
- Choose **Next**
- Choose **Next**
- Choose **Confirm**

87. This configures AWS Config for initial use. A **Welcome to AWS Config** window pops up. You can close it.

88. In the left navigation pane, choose **Rules** (the one toward the top).

89. Choose **Add rule**

90. In the **AWS Managed Rules** section in the search field, enter `required-tags`

91. Select the button next to **required-tags**.

92. Choose **Next**

You configure the rule to require a project code for each resource.

93. In the **Configure rule** page, scroll to **Parameters**, and configure the following settings:

- To the right of **tag1Key**, enter `project` (replace any existing value).
- Choose **Next** (at the bottom of the page).
- Choose **Add rule**

94. This rule now looks for resources that do not have a **project** tag. This takes a few minutes to complete, so continue with the next steps. You do not need to wait.

You now add a rule that looks for EBS volumes that are not attached to EC2 instances.

95. Choose **Add rule**

96. In the **AWS Managed Rules** section in the search field, enter

`ec2-volume-inuse-check`

97. Select the button next to **ec2-volume-inuse-check**.

98. Choose **Next**

99. Choose **Next** again.

100. Choose **Add rule**

101. Wait until at least one of the rules has completed evaluation. Refresh your browser page if necessary.

If you receive a message that there are **No resources in scope**, wait a few

minutes longer. This message is an indication that AWS Config is still scanning available resources. The message eventually disappears.

102. Choose each of the rules to view the result of the audits.

103. Under **Resources in scope** select **Compliant** from the list.

The following should be among the results:

- **required-tags:** A compliant EC2 instance (because the Web Server has a **project** tag) and many non-compliant resources that do not have a **project** tag
- **ec2-volume-inuse-check:** One compliant volume (attached to an instance) and one non-compliant volume (not attached to an instance)

104. AWS Config has a large library of pre-defined compliance checks, and you can create additional checks by writing your own AWS Config rule using Lambda.