Using the AWS Systems Manager Run Command for Automation

**SPL-BE-100-CESMRC-1 - Version 1.0.2**

© 2024 Amazon Web Services, Inc. or its affiliates. All rights reserved. This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited. All trademarks are the property of their owners.

Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

Corrections, feedback, or other questions? Contact us at [*AWS Training and Certification*](https://support.aws.amazon.com/#/contacts/aws-training).

**Lab overview**

In this lab, you acquire practical skills with Run Command, a capability of AWS Systems Manager, to manage a fleet of three Amazon Elastic Compute Cloud (Amazon EC2) instances. Through the Run Command console, you run the *AWS-RunShellScript* command document to seamlessly transition from using Apache HTTP Server to using the nginx web server across the entire fleet of hosts. Later, you are challenged to update only one of the three EC2 instances back to using the Apache HTTP Server, instead of using the nginx web server.

OBJECTIVES

By the end of this lab, you are able to do the following:

* Review the current infrastructure, which encompasses three EC2 instances and an Application Load Balancer.
* Test the DNS name of the Application Load Balancer to see the Apache HTTP Server response that’s passed from one of the three instances in the Application Load Balancer target group.
* Launch a Run Command feature from the Systems Manager console with the *AWS-RunShellScript* command document to uninstall Apache HTTP Server and install the nginx web server.

TECHNICAL KNOWLEDGE PREREQUISITES

This lab requires:

* Familiarity navigating the AWS Management Console.
* Experience using and updating Bash scripts is helpful.

ICON KEY

Various icons are used throughout this lab to call attention to different types of instructions and notes. The following list explains the purpose for each icon:

* **Note:** A hint, tip, or important guidance.
* **Knowledge check:** An opportunity to check your knowledge and test what you have learned.
* **Refresh:** A time when you might need to refresh a web browser page or list to show new information.
* **Copy edit:** A time when copying a command, script, or other text to a text editor (to edit specific variables within it) might be easier than editing directly in the command line or terminal.
* **Hint:** A hint to a question or challenge.
* An answer to a question or challenge.
* **Task complete:** A conclusion or summary point in the lab.

**Start lab**

1. To launch the lab, at the top of the page, choose **Start lab**.

**Caution:** You must wait for the provisioned AWS services to be ready before you can continue.

1. To open the lab, choose **Open Console**.

You are automatically signed in to the AWS Management Console in a new web browser tab.

**WARNING:** **Do not change the Region unless instructed.**

COMMON SIGN-IN ERRORS

**Error: You must first sign out**



If you see the message, **You must first log out before logging into a different AWS account:**

* Choose the **click here** link.
* Close your **Amazon Web Services Sign In** web browser tab and return to your initial lab page.
* Choose **Open Console** again.

**Error: Choosing Start Lab has no effect**

In some cases, certain pop-up or script blocker web browser extensions might prevent the **Start Lab** button from working as intended. If you experience an issue starting the lab:

* Add the lab domain name to your pop-up or script blocker’s allow list or turn it off.
* Refresh the page and try again.

**Task 1: Review the existing infrastructure**

In this task, you review the three EC2 instances that have been created for use in this lab. You also review the Application Load Balancer.

TASK 1.1: REVIEW THE EC2 INSTANCES

In this lab, three EC2 instances have been created by using the Amazon Linux 2 Amazon Machine Image (AMI). This AMI comes with the AWS Systems Manager Agent (SSM Agent) preinstalled. The SSM Agent is software that instances use to communicate with the Systems Manager service. You can use the SSM Agent to manage your instances through Systems Manager capabilities, such as Run Command, Inventory, State Manager, and more.

The SSM Agent makes it possible for Systems Manager to update, configure, and run commands on your instances, which can be especially helpful for batch operations or fleet management tasks. You can use it to run shell scripts, install software or patches, configure software, and gather information about operating system resources.

In addition to the SSM Agent, the instances have an instance profile attached. An instance profile is a container for an AWS Identity and Access Management (IAM) role that can be used to pass permissions to an EC2 instance at launch time. Essentially, an instance profile is a way to grant an EC2 instance permissions to perform certain actions, such as accessing Amazon Simple Storage Service (Amazon S3) buckets, invoking AWS Lambda functions, or any other API actions that you can define in an IAM policy.

The instance profile that’s attached to the three EC2 instances used in this lab has the *AmazonSSMManagedInstanceCore* policy attached to it. This policy is a managed IAM policy that grants the minimum permissions required for an instance to be managed by Systems Manager. For this lab, the policy allows you to use one of the core Systems Manager capabilities, known as Run Command operations, to run shell scripts.

The Amazon EC2 User Data for each instance consists of Bash commands to perform the following actions:

* Creates a token to provide authentication to access the instance metadata service.
* Creates a variable named *INSTANCE\_ID* to store the EC2 instance ID. It uses the token to access the metadata service and retrieve the value of the *instance-id*.
* Uses the yum package manager to install Apache HTTP Server.
* Uses an echo command to create the index.html file that is used by Apache HTTP Server as the default webpage for serving HTTP requests. The echo command adds the instance ID to the first line in the file, followed by a second line that reads, *Hello, from Apache!*
* Uses *systemctl* to allow Apache HTTP Server to start automatically when the system boots up, and to start Apache HTTP Server immediately. The *systmctl* utility is a command line utility that you can use to manage and control the *systemd* system and service manager for a Linux operating system.

The following is an example of the Amazon EC2 User Data that’s used by all three instances:

#!/bin/bash

#Creates token to authenticate and retrieve instance metadata

TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600"`

#Variable for instance ID

INSTANCE\_ID=$(curl -H "X-aws-ec2-metadata-token: $TOKEN" -v http://169.254.169.254/latest/meta-data/instance-id)

# Install Apache and configure the index.html file

yum install -y httpd

echo "<br><br><br><h4 align="Center">INSTANCE ID: $INSTANCE\_ID</h4><br><h1 align="Center">Hello, from Apache!</h1>" > /var/www/html/index.html

# Enable and start Apache

systemctl enable httpd

systemctl start httpd

Review the three EC2 instances used in this lab.

1. At the top of the AWS Management Console, in the search bar, search for and choose

EC2

.

1. In the **Resources** section, choose the **Instances (running)** link.

The EC2 console shows a list of three instances named *WebServer1*, *WebServer2*, and *WebServer3*. The instance states should be set to *Running* and the instance types are set to *t3.micro*.

**Note:** The *t3.micro* instance type is used for low-traffic web servers; development and test environments; and other small, non-resource-intensive applications.

1. Reveal additional details by selecting the checkbox for one of the instances.

Key settings to acknowledge for this lab include the following:

Under *Details:*

* + IAM Role: This is set to the *WebServer-Role*. The *AmazonSSMManagedInstanceCore* policy is attached to this role. You can verify that the policy is attached by selecting the link. Then, you can expand the *AmazonSSMManagedInstanceCore* policy to see the Systems Manager permissions that are granted by this policy.

Under *Tags:*

* + Each instance has two specific tags that are assigned for use in this lab.
  + One tag is the *Name* tag with a value of *WebServer1-3*, which is unique to each instance.
  + Another tag, shared by each instance, has a key of *ServerType* with a value of *WebServer*. This tag is used later in the lab when you use the Run Command option with a shell script to manage the fleet of instances.

1. To verify the User Data, keep the instance selected and choose **Actions**, then **Instance settings**, and finally **Edit user data** .

In the **Current user data** text box, review the code snippet.

1. When you finish verifying the User Data, close this window and return the EC2 dashboard by choosing **Cancel** .

If you like, you can complete the same tasks for the remaining two instances. Otherwise, you can move on to the next task.

TASK 1.2: REVIEW THE APPLICATION LOAD BALANCER

An Elastic Load Balancing (ELB) Application Load Balancer is a fully managed load-balancing service for HTTP and HTTPS traffic. It is optimized to route incoming requests to multiple targets, such as EC2 instances, in multiple Availability Zones. It operates at Layer 7 of the Open Systems Interconnection (OSI) model, which means it can make routing decisions at the application layer. Because it works at the application layer, an Application Load Balancer can support more advanced features, such host-based or path-based routing. It can also support routing based on standard or custom HTTP headers, methods, query parameters, and source IP addresses.

For this lab, the Application Load Balancer is used to distribute incoming website traffic across multiple targets (EC2 instances) for fault tolerance and low latency. Listeners (the ports to listen on) are defined, rules (how to route requests based on content) are also defined, and then targets (such as EC2 instances) are registered so that requests can be routed to them. After the Application Load Balancer is configured, it distributes incoming client requests according to the rules that have been set. These rules help ensure that each registered target get its share of the request load.

1. From the menu to the left, under  **Load Balancing**, choose the **Load Balancers** link.

One load balancer is listed, and it’s named *my-load-balancer*.

1. **Copy edit:** Copy the URL value listed under **DNS name** and save it in a text editor because you refer to it later. Additionally, the URL is listed to the left of these instructions, where you can also copy it when you need to.

Notice the load balancer has three Availability Zones. Each EC2 instance has one Availability Zone.

1. Choose the **my-load-balancer** link.
2. Look at the **VPC** section. You can see that three Availability Zones are listed.
3. Review the **Listeners and rules** section. You can see that the **Protocol:Port** is set to **HTTP:80**.
4. Under the **Default action**, you can see it is set to **Forward to target group**.
5. In a new browser tab, open the **my-target-group** link.

The browser takes you to the *my-target-group* dashboard. Review this dashboard and note the following details:

* + *Targets:* The group has a total of three targets. The targets reference the same EC2 instances you viewed in the Amazon EC2 console.
  + *Health:* All three instances should show a status of *Healthy*. If an instance fails health checks, it is deregistered from the target group and the faulty instance is removed from the list of targets that the Application Load Balancer directs traffic to.

In the *Attributes* section, be aware of the following attributes:

* + *Traffic configuration*
    - The load-balancing algorithm is set to *Round robin*. Based on the round robin routing algorithm, the Application Load Balancer distributes the traffic equally across each of the three registered targets in its scope.
  + *Target selection configuration*:
    - *Stickiness* is set to *Off*. By default, an Application Load Balancer routes each request independently to a registered target, based on the chosen load-balancing algorithm. However, you can use the sticky session feature (also known as session affinity) to enable the load balancer to bind a user’s session to a specific target. A sticky session means that all requests from the user during the session are sent to the same target. This feature is useful for servers that maintain state information to provide a continuous experience for clients. To use sticky sessions, the client must support cookies.

**Task complete:** Congratulations, you have reviewed the EC2 instances used as web servers. You have also reviewed the Application Load Balancer that’s used to direct web traffic to each web server.

**Task 2: Test the link for the DNS name of the Application Load Balancer and update the User Data by using Run Command**

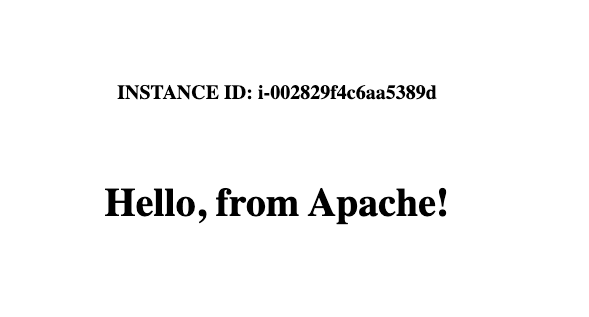
In this task, you paste the link for the DNS name of the Application Load Balancer into a new browser tab. By opening the link in the browser, you can see the response from the targets (EC2 web servers) that are registered with the Application Load Balancer. After viewing the web servers’ responses, you remove Apache HTTP Server and install the nginx web server by using the Systems Manager console and Run Command to run a shell script.

TASK 2.1: VISIT THE DNS NAME OF THE APPLICATION LOAD BALANCER IN A NEW BROWSER TAB

It’s time to verify that the Application Load Balancer properly directs web traffic (that’s bound for HTTP port 80) to one of the three registered Amazon EC2 targets.

1. Copy the DNS name of the Application Load Balancer (that you copied earlier, or to the left of these instructions) and in a new browser tab’s address bar, paste it and press Enter.

The browser opens a webpage with similar content to the following.



*The image shows a webpage that comprises a line that reads, “Instance ID: INSTANCE\_ID\_VALUE”, and a second line that reads, “Hello, from Apache!”.*

**Note:** If you are wondering how the instance ID and *Hello, from Apache!* text is being rendered, refer back to Task 1, which explains the User Data.

1. Refresh the page and notice how the value of the instance ID changes.

You might notice times when the ID might stay the same. However, as you keep refreshing the page, the requests should filter through the three different instance IDs. These IDs correspond to the three EC2 instances you saw in the Amazon EC2 console and in the target group for the Application Load Balancer.

This test confirms that the Application Load Balancer is redirecting web traffic bound for HTTP port 80 to the three web servers. The test also confirms that the instances are configured to run Apache HTTP Server.

TASK 2.2: USE THE SYSTEMS MANAGER RUN COMMAND FEATURE TO RUN A SHELL SCRIPT

In this subtask, you use Run Command. You remove Apache HTTP Server from all three instances, and install and configure the nginx web server on all instances with a shell script.

**Note:** It is important to realize what the shell script does and doesn’t do. Though the code snippet looks exactly like the User Data code snippet, the User Data for each instance does not get updated. Instead, Run Command runs a shell script that performs those actions on the instances by using Systems Manager. Then, Systems Manager invokes the SSM Agent on each instance to run those commands. Using the shell script in Run Command is an independent action that’s run outside the User Data.

1. At the top of the AWS Management Console, in the search bar, search for and choose

Systems Manager

.

1. From the menu to the left, under the  **Node Management** heading, choose **Run Command**.
2. Choose **Run a Command**.
3. In the search bar, enter

AWS-RunShellScript

 and press **Enter**.

1. Select the **AWS-RunShellScript** radio button.
2. Copy the following code snippet.
3. #!/bin/bash
4. # Creates token to authenticate and retrieve instance metadata
5. TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600"`
6. # Variable to store the instance ID
7. INSTANCE\_ID=$(curl -H "X-aws-ec2-metadata-token: $TOKEN" -v http://169.254.169.254/latest/meta-data/instance-id)
8. # Remove Apache and install nginx
9. yum remove -y httpd
10. yum install -y nginx
11. # Update default nginx index.html with instance ID
12. echo "<br><br><br><h4 align="Center">INSTANCE ID: $INSTANCE\_ID</h4>" > /tmp/index.html
13. cat /usr/share/nginx/html/index.html >> /tmp/index.html
14. mv /tmp/index.html /usr/share/nginx/html/index.html
15. # Update the nginx.conf file to turn off the if\_modified\_since property
16. sed -i 's/http {/http { \n if\_modified\_since off;/g' /etc/nginx/nginx.conf
17. systemctl restart nginx
18. # Enable and start nginx
19. systemctl enable nginx

systemctl start nginx

**Note:** This code snippet should look familiar, except for a few lines.

* + It creates a token to access the instance metadata service, which is used to retrieve and store the instance ID in a variable.
  + It removes Apache HTTP Server, and then installs and configures the nginx web server.
  + It updates the existing *index.html* file to add the *instance ID* value to the top of the *index.html* file.
  + It updates the /etc/nginx/nginx.conf file to turn the *if\_modified\_since* property to *off*.
    - This property defines how nginx handles the *If-Modified-Since HTTP header*. This header is mostly used by search engine spiders (such as Google web crawling bots). The robot indicates the date and time of the previous pass. If the requested file has not been modified since the earlier pass, the server returns a *304 Not Modified* response code without a body. From the visitor’s perspective, it returns a cached version of the file. If the modification date is earlier or later, the entire file is served normally (returning a 200 OK response). What does this behavior mean for you in this lab? If the *if\_modified\_since* value is not turned off, it appears that the Application Load Balancer constantly sends you to the same instance over and over when you refresh your browser. In this case, the browser refresh reloads the cached version of the webpage with the same instance ID, instead of loading an entirely new webpage that has a new instance ID each time. By turning this setting off, you avoid this situation and the possible confusion it might cause.

1. In the **Commands parameters** section, in the **Commands** text box, paste the code snippet.
2. In the **Target selection** section, select the **Specify instance tags** radio button, if it’s not already selected.
3. In the **Specify instance tags** text boxes, enter the following details:
   * For **Tag key**, enter

ServerType

.

* + For **Tag value (optional)**, enter **WebServer**.

1. Choose **Add** .

A new tag selection box is generated for *ServerType: WebServer*.

1. Keep all remaining default options and choose **Run** .

The **Run Command** console displays a similar message: *Command ID: SOME\_ID was successfully sent!*

The *Overall status* is set to *In Progress*. In the *Targets and outputs* section, you see a list of three EC2 instance IDs with a status of *In Progress*.

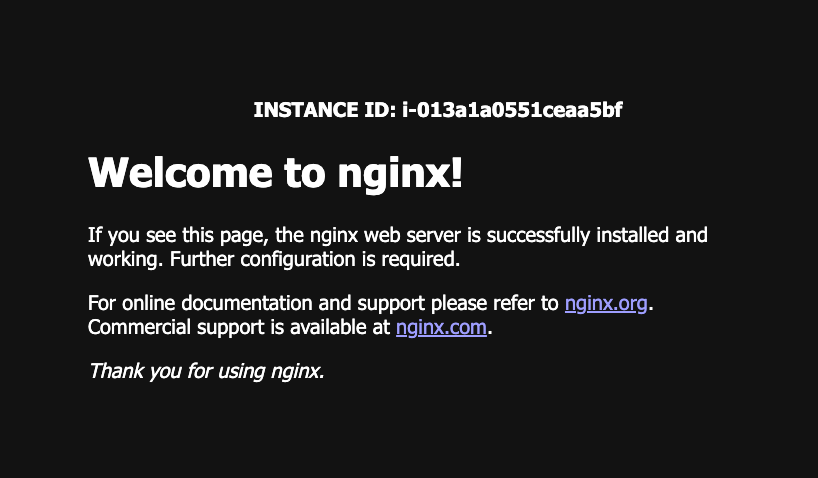
1. After a few moments, refresh the console by choosing the refresh button  .

All statuses should now be updated to *Success*, unless an error occurred.

**Note:** If you encounter an error, review the previous steps. Make sure the code snippet matches, and try again.

1. Switch to the browser tab that’s opened to the DNS name URL of the Application Load Balancer, and refresh the page.

The browser should render a new webpage. The top of the webpage should show *Instance ID:* followed by the ID. The webpage should also include additional lines with a *Welcome to nginx!* message, followed by the standard nginx boilerplate text, as shown in the following example.



*The image shows a webpage that comprises a first line that reads, “Instance ID: INSTANCE\_ID\_VALUE”, and a second line that reads, “Welcome to nginx!” These two lines are followed by standard nginx boilerplate text.*

If you refresh the browser, you see that it filters through different instance IDs as the HTTP traffic is routed to different instances.

**Task complete:** You have successfully tested the DNS name URL of the Application Load Balancer to view the Apache HTTP Server responses to web traffic. You have updated the EC2 instance User Data to remove the Apache HTTP Server, and to install and configure the nginx web server. The User Data also updates the existing default index.html file to display an instance ID at the top of the page. This instance ID belongs to instance that the HTTP traffic was sent to.

**Challenge: Update one instance to use Apache HTTP Server**

In this challenge, you use all the knowledge you have gained in this lab to update one instance to use Apache HTTP Server instead of nginx. You have already walked through the steps necessary to complete this challenge. Consider the logic that needs to be applied to update the Bash commands to complete the task.

**Knowledge check:** This challenge is meant to test your knowledge and understanding of the content covered in this lab. Try to complete the steps without using the hints or solution code.

If you find yourself struggling with the challenge and need assistance to make progress, clues (and a full solution) are provided so that you can complete this challenge.

1. To begin the challenge, you need to use the **Systems Manager** service. Refer to steps in **Task 2.2** if you need general assistance.

**Hint 1**

**Hint 2**

**Hint 3**

**Solution**

What do you expect to see when you refresh the browser that’s opened to the DNS name of the Application Load Balancer?

1. Switch to the browser tab opened to the Application Load Balancer DNS name, and refresh the page.

Did the browser respond with what you expected? You might need to refresh the page a few times, but you should start seeing that the pages alternate being returned from nginx or Apache. As the Application Load Balancer manages the web traffic, the pages update based on the web server you are directed to.

**Task complete:** Congratulations, you completed the challenge portion of the lab. You successfully updated the User Data for one EC2 instance to use Apache HTTP Server instead of nginx, while the other two instances continue to use nginx.

**Conclusion**

You now have successfully done the following:

* Reviewed the current infrastructure, which encompasses three EC2 instances and an Application Load Balancer.
* Tested the DNS name of the Application Load Balancer to see the Apache HTTP Server response that’s passed from one of the three instances in the Application Load Balancer target group.
* Launched a Run Command feature from the Systems Manager console with the *AWS-RunShellScript* command document to uninstall Apache HTTP Server and install the nginx web server.

**End lab**

Follow these steps to close the console and end your lab.

1. Return to the **AWS Management Console**.
2. At the upper-right corner of the page, choose **AWSLabsUser**, and then choose **Sign out**.
3. Choose **End lab** and then confirm that you want to end your lab.

For more information about AWS Training and Certification, see [*https://aws.amazon.com/training/*](https://aws.amazon.com/training/).

*Your feedback is welcome and appreciated.*  
If you would like to share any feedback, suggestions, or corrections, please provide the details in our [*AWS Training and Certification Contact Form*](https://support.aws.amazon.com/#/contacts/aws-training).