Securing VPC Resources with Security Groups

**SPL-255 - Version 1.0.9**

© 2023 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**

Security groups are virtual firewalls attached to Amazon Elastic Compute Cloud (Amazon EC2) instances. Security group rules define what traffic is allowed in or out of an instance. In this lab, you are a security monitor tasked with configuring access rules for Amazon EC2 instances. You must ensure that only authorized traffic is allowed into each instance. To accomplish this task, review what traffic should be allowed, and inspect the security group rules attached to an instance. Then, test connectivity and correct any misconfigured rules.

In this lab, an instance named **AppServer**, which acts as an application server, has been launched into a *private* subnet. This means that the instance is not accessible directly from the internet. To test the **AppServer** security group rules, you will first connect to an intermediary Amazon EC2 instance in a *public* subnet of the same virtual private cloud (VPC). The intermediary instance is known as a bastion host (or jump server) and therefore is named **BastionHost**. From the **BastionHost** connection, you will connect to and test the security group rules of the **AppServer** instance in the private subnet. Implementing a bastion host/jump server model is a common network security configuration to remotely administer private subnet resources.

There is a second Amazon EC2 instance named **PublicServer** in the public subnet. You will first use **PublicServer** to test and ensure that SSH traffic is only allowed into **AppServer** from **BastionHost**. This minimizes the risk of unauthorized SSH operations on **AppServer**. Later, you will duplicate security configuration from **BastionHost** to **PublicServer** to create a second bastion host for redundancy.

For simplicity, the agent for the AWS Systems Manager service has been installed on **BastionHost** and **PublicServer**. This allows you to use the Systems Manager Session Manager to instantly create an SSH session to either of those instances using a preconfigured URL in your browser. Toward the end of the lab, you will investigate more about Session Manager as an alternative to using a traditional bastion host.

Session Manager is a fully managed AWS Systems Manager capability that lets you manage your Amazon EC2 instances, on-premises instances, and virtual machines (VMs) through an interactive, one-click, browser-based shell or through the AWS Command Line Interface (AWS CLI). Session Manager provides secure and auditable instance management without the need to open inbound ports, maintain bastion hosts, or manage SSH keys. Session Manager also makes it easy to comply with corporate policies that require controlled access to instances, strict security practices, and fully auditable logs with instance access details, while still providing end users with simple one-click cross-platform access to your managed instances.

TOPICS COVERED

After completing this lab, you will be able to:

* Examine security groups and determine what traffic is allowed.
* Change which security groups are applied to Amazon EC2 instances.
* Update security groups to follow the principle of least privilege.
* Understand how security groups can reference other security groups.
* Understand how to leverage Session Manager to connect to instances.

PREREQUISITES

This lab requires:

* Access to a notebook computer with Wi-Fi and Microsoft Windows, macOS, or Linux (Ubuntu, SuSE, or Red Hat)

**Note** The lab environment is not accessible using an iPad or tablet device, but you can use these devices to access the student guide.

* For Microsoft Windows users: Administrator access to the computer
* An internet browser such as Chrome, Firefox, or Internet Explorer 9 (previous versions of Internet Explorer are not supported)
* Optional: An SSH client such as PuTTY

DURATION

This lab requires about 45 minutes to complete.

AWS SERVICES NOT USED IN THIS LAB

AWS services that are not used in this lab are disabled in the lab environment. In addition, the capabilities of the services used in this lab are limited to what the lab requires. Expect errors when accessing other services or performing actions beyond those provided in this lab guide.

**Start lab**

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

 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.

**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.

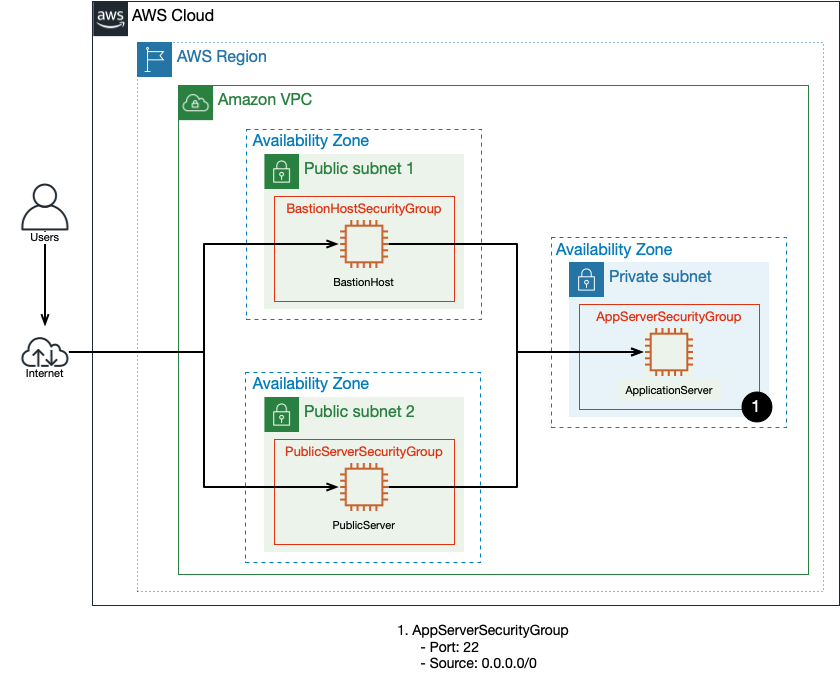
**Error: Choosing Start Lab has no effect**

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: Inspect VPC resources and the AppServer**

In this task, you review the VPC configuration and inspect the preconfigured security group attached to **AppServer**. The following diagram shows the VPC configuration.



Note that **BastionHost** and **PublicServer** are located in public subnets 1 and 2, so these instances can be accessed over the internet. **AppServer** is in a private subnet, which means that it is not directly accessible from the internet. You will connect to **BastionHost** and then connect from **BastionHost** to **AppServer**.

Your current corporate security policy states that all SSH traffic to instances in private subnets must connect through a bastion host.

Next, inspect the **AppServerSecurityGroup** security group rules to see what inbound traffic is currently allowed into the **AppServer** instance.

1. At the top of the AWS Management Console, to the right of **Services** menu, in the search bar, search for

**VPC**

 and then choose **VPC** from the list.

**Note** When you create a security group, it has no inbound rules. This means that by default security groups do not allow any traffic until you explicitly add inbound rules. Security groups are *stateful*. This means that if you send a request from your instance, the response traffic for that request is allowed in regardless of inbound security group rules.

1. In the left navigation pane, click **Security Groups**.
2. Click **Security group ID** with the Security group name **AppServerSecurityGroup**.
3. In the **Inbound rules** tab view the inbound traffic rules for the security group.

One inbound rule is listed:

* SSH via port 22 with source 0.0.0.0/0

This configuration allows inbound connections to **AppServer** from *any* IP address over port 22. All other ports deny access. Even though **BastionHost** should be able to connect to **AppServer**, unauthorized hosts from any IP address (source 0.0.0.0/0) can also currently connect to **AppServer**. ***This security group is noncompliant based on your corporate security policy.***

Security groups can be assigned to multiple resources. Any instance that has **AppServerSecurityGroup** attached to it will follow these same rules, allowing traffic over port 22.

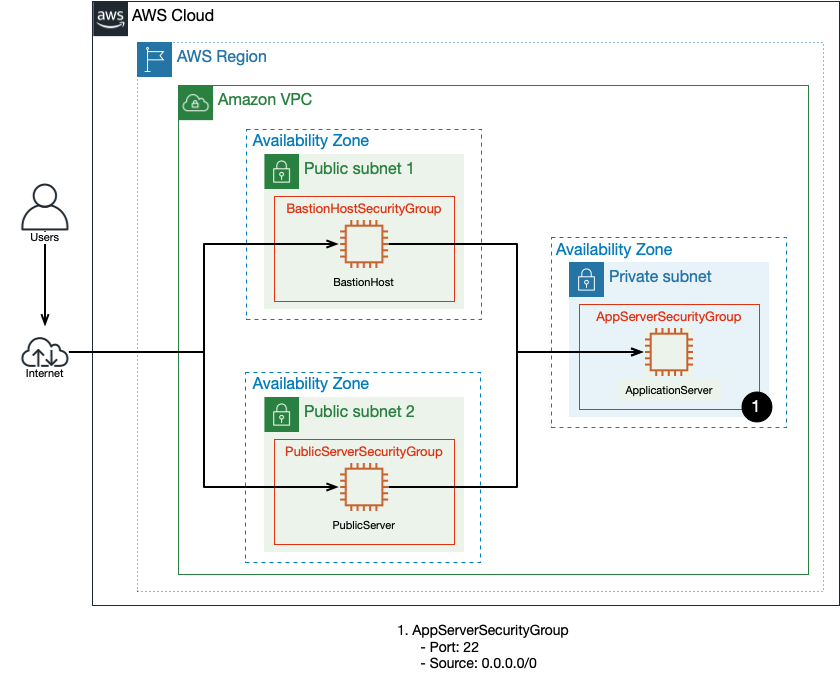
1. To view the outbound traffic rules for the security group, click the **Outbound rules** tab.

Traffic originating from an Amazon EC2 instance requires an outbound security group rule that allows the traffic to pass. Notice that the current rule allows *all* outbound traffic (0.0.0.0/0). You will not modify the outbound rule in this lab.

Now, it’s time to test the current security group rules for **AppServer**.

**Task 2: Test SSH connectivity to AppServer from public instances**

In this task, you test the current **AppServer** security group rules by attempting to SSH into **AppServer** from **BastionHost** and then from **PublicServer**.



To test SSH connectivity to **AppServer**, you first connect to **BastionHost** and then SSH to **AppServer**. Remember, this lab simplifies your first SSH connection by implementing Session Manager, so your initial connection into **BastionHost** or **PublicServer** works simply by pasting the URL into a new browser tab.

**Learn more** For more information about setting up Session Manager in your environment, see [Start a Session](https://docs.aws.amazon.com/systems-manager/latest/userguide/session-manager-working-with-sessions-start.html).

1. Copy the **BastionHostSessionManagementUrl** value from the left side of the lab page, and paste it in a **new** browser tab.

The command host terminal appears. You have now established a session to **BastionHost**.

**Note** If you encounter issues connecting to Session Manager, [click here](https://labs.skillbuilder.aws/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A470679935125%3Ablueprintversion%2Fspl-255%3A1.0.9-ff293c68/en-US#ssh) for help connecting to an Amazon EC2 instance using SSH.

1. In the **BastionHost** browser tab, run the following command to connect to the application instance. Replace **(AppServerPrivateIP)** with the value located to the left of these instructions.

ssh user-1@(AppServerPrivateIP)

1. When prompted, type

yes

 to continue connecting.

1. When prompted for a password, enter the **AppServerPassword** value located to the left of these instructions.

You are now connected to **AppServer**.

1. Run the following command:

hostname

This command displays the hostname of the instance you are currently connected to (AppServer), which includes the private IP address assigned to **AppServer**. This confirms you have successfully opened an SSH session to **AppServer** from **BastionServer**. ***This is compliant behavior.***

Now, attempt an SSH connection from **PublicServer** to **AppServer**.

1. Copy the **PublicServerSessionManagementUrl** value from the left side of the lab page, and paste it in a **new** browser tab.

The command host terminal appears. You have now opened a session to **PublicServer**.

Now, follow the same steps as above to connect to **AppServer** but from **PublicServer**.

1. In the **PublicServer** browser tab, run the following command to connect to the application instance. Replace **(AppServerPrivateIP)** with the value located to the left of these instructions.

ssh user-1@(AppServerPrivateIP)

1. When prompted type,

yes

 to continue connecting.

1. When prompted for a password, enter the **AppServerPassword** value located to the left of these instructions.

You are now connected to **AppServer**.

1. Run the following command:

hostname

This command displays the hostname of the instance you are currently connected to (AppServer), which includes the private IP address assigned to **AppServer**. This confirms you have successfully opened an SSH session to **AppServer** from **Public Server**. ***This is noncompliant behavior and must be corrected.***

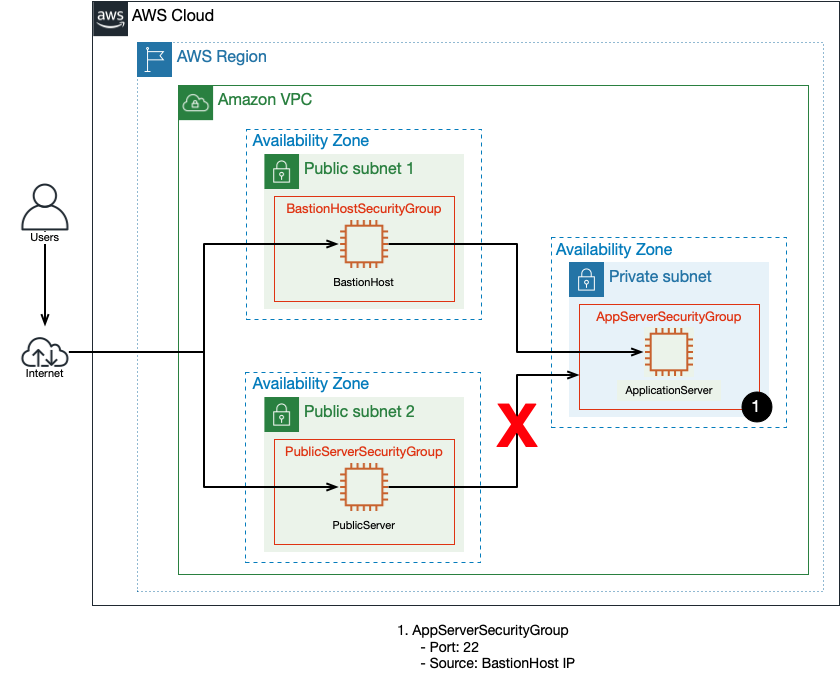
1. Close the browser tabs that are connected to **BastionHost** and **PublicServer**.

**Note** Leave the AWS Management Console tab open to use in later steps.

In the next task, you update the security group rules to make them compliant. Remember, your compliance requirements state that inbound SSH traffic can only originate from a bastion host.

**Task 3: Restrict SSH access to AppServer from a specific IP address**

To make the **AppServer** security group compliant, you need to modify the rules to only allow incoming SSH traffic from a bastion host. At present, **BastionHost** is your only bastion. In this task, you update the **AppServerSecurityGroup** to allow SSH access, specifying the internal IP address of **BastionHost** as the only possible source.



1. Return to the browser tab with the AWS Management Console. At the top of the AWS Management Console, to the right of **Services** menu, in the search bar, search for

**EC2**

 and then choose **EC2** from the list.

1. At the top left of the screen, if you see **New EC2 Experience** toggle to use the new UI, if it is not enabled by default.
2. In the left navigation pane, click **Instances**.
3. Select the **BastionHost** instance.
4. On the **Details** tab, find the **Private IPv4 addresses** item, and click the copy to clipboard button for the IP address of the instance.

**Note** This IP address should start with *10.0.1.*.

1. In the left navigation pane, click **Security Groups**.

**Note** You can access security groups from both the Amazon VPC and Amazon EC2 consoles.

1. Click **Security group ID** with the Security group name **AppServerSecurityGroup**.
2. Click **Edit inbound rules**.
3. For the **SSH** rule, click the X button for the 0.0.0.0/0 inbound IP address to remove it, and update the source to match the IP address you copied for **BastionHost**. Make sure to add */32* to the end of the IP address.
4. Click **Save rules**

You have now updated the security group attached to the application server so that it will *only* allow SSH traffic from **BastionHost**. Now it’s time to test your changes for compliance.

1. Copy the **PublicServerSessionManagementUrl** value from the left side of the lab page, and paste it in a **new** browser tab. The command host terminal appears.
2. In the **PublicServer** browser tab, run the following command to connect to the application instance. Replace **(AppServerPrivateIP)** with the value located to the left of these instructions.

ssh user-1@(AppServerPrivateIP)

The command hangs and does not complete. After about 2 minutes, you receive a notice saying:

ssh: connect to host 10.0.11.xxx port 22: Connection timed out

This confirms you were unsuccessful in opening an SSH session to **AppServer** from **PublicServer**. ***This is compliant behavior.***

Now, confirm SSH connectivity to **AppServer** from **BastionHost**.

1. Copy the **BastionHostSessionManagementUrl** value from the left side of the lab page, and paste it in a **new** browser tab. The command host terminal appears.
2. In the **BastionHost** browser tab, run the following command to connect to the application instance. Replace **(AppServerPrivateIP)** with the value located to the left of these instructions.

ssh user-1@(AppServerPrivateIP)

1. When prompted for a password, enter the **AppServerPassword** value located to the left of these instructions.

You are now connected to **AppServer**.

1. Run the following command:

hostname

This command displays the hostname of the instance you are currently connected to (AppServer), which includes the private IP address assigned to **AppServer**. This confirms you have successfully opened an SSH session to **AppServer** from **BastionHost**. ***This is compliant behavior.***

1. Close the browser tab with the **BastionHost** terminal.

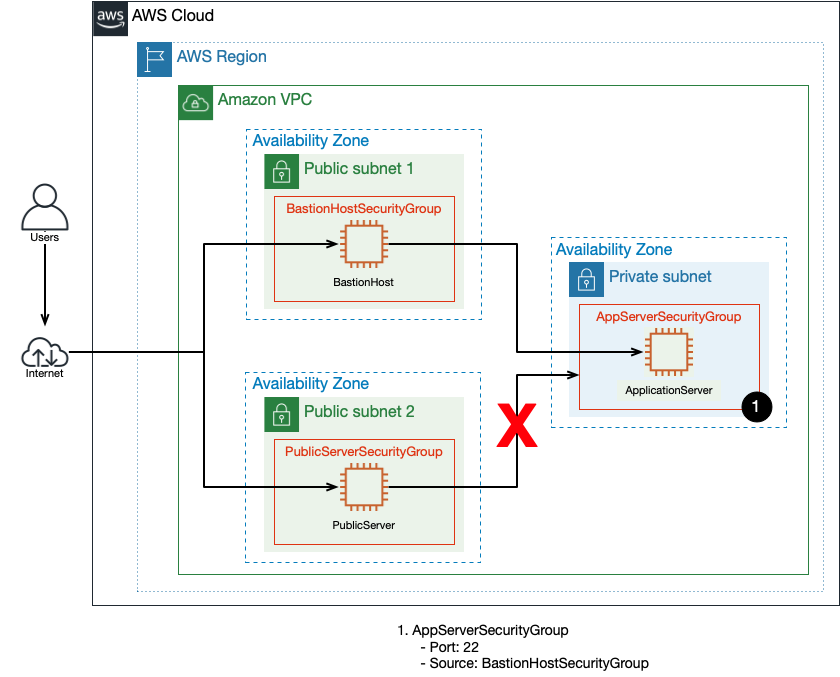
**Task 4: Restrict SSH access by referencing a security group as the inbound source**

Although you can list individual IP addresses as a source in a security group rule, as you did in the previous task, there is another important method to specify sources in security group rules. You can set the incoming source for traffic to be the name of a security group.

By updating the **AppServer** security group SSH rule and setting the source to be **BastionHostSecurityGroup**, any instance such as **BastionHost** that has **BastionHostSecurityGroup** attached to it can send SSH traffic successfully to **AppServer**. This is useful if you want multiple servers to have the same type of connectivity.

If using bastion hosts, best practice is to create at least two for redundancy. You have decided to turn **PublicServer** into another bastion host. What’s the simplest way to give the same **AppServer** access permissions currently assigned to **BastionHost** also to **PublicServer**? Attach **BastionHostSecurityGroup** to **PublicServer**.

In this task, you modify the **AppServer** security group rule to only allow inbound SSH connections from instances using the **BastionHostSecurityGroup**. Then, you change the security group assigned to **PublicServer** to be the **BastionHostSecurityGroup**. This means that **PublicServer** and **BastionHost** are assigned the same security group, and both instances are valid incoming traffic sources using SSH into **AppServer**. Lastly, you test connectivity to **AppServer** from both **BastionHost** and **PublicServer**.



1. Return to the browser tab with the AWS Management Console. At the top of the AWS Management Console, to the right of **Services** menu, in the search bar, search for

**EC2**

 and then choose **EC2** from the list.

1. In the left navigation pane, click **Security Groups**.
2. Click **Security group ID** with the Security group name **AppServerSecurityGroup**.
3. Click **Edit inbound rules**.
4. Delete the existing **SSH Rule**.
5. Add a new rule, choose type as **SSH** ,in the source text box, enter

BastionHostSecurityGroup

A pop up box appears with an option to select a single security group, click it.

1. Click **Save rules**

You have now updated the security group attached to **AppServer** so that it will *only* allow traffic from instances with the **BastionHostSecurityGroup** attached.

Now, change the security group assigned to **PublicServer** to be the same as **BastionHost**.

1. In the left navigation pane, click **Instances**.
2. Select the **PublicServer** instance.
3. Click **Actions** select **Security**, and select **Change security groups**.
4. Remove **PublicServerSecurityGroup**, and select **BastionHostSecurityGroup** from the list by clicking **Add security group**.
5. Click **Save**

Now it’s time to test the connectivity from the redundant bastion host (PublicServer).

1. Return to the open **PublicServer** browser tab. If you accidentally closed it, just copy the **PublicServerSessionManagementUrl** value from the left side of the lab page, and paste it in a **new** browser tab.
2. In the **PublicServer** browser tab, run the following command to connect to the application instance. Replace **(AppServerPrivateIP)** with the value located to the left of these instructions.

ssh user-1@(AppServerPrivateIP)

1. When prompted for a password, enter the **AppServerPassword** value located to the left of these instructions.

You are now connected to **AppServer**.

1. Run the following command:

hostname

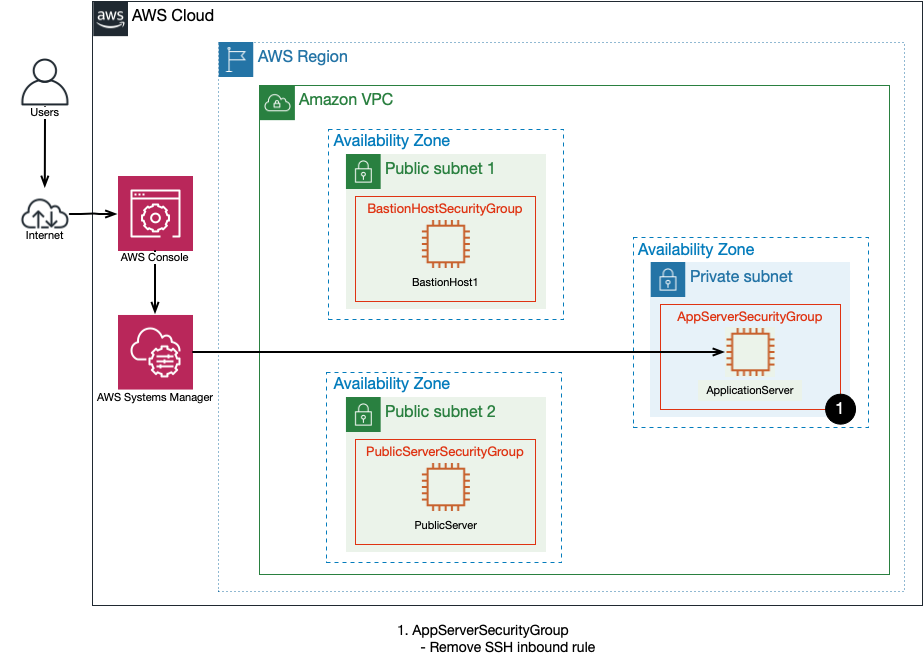
This command displays the hostname of the instance you are currently connected to (AppServer), which includes the private IP address assigned to **AppServer**. This confirms you have successfully opened an SSH session to **AppServer** from **PublicServer**. ***This is now compliant behavior since we made the PublicServer an additional bastion host by adding it to the Bastion Host Security Group***

1. Close the browser tab with the **PublicServer** terminal.

**Challenge: AWS Systems Manager Session Manager**

In this lab, you have connected to **BastionHost** and **PublicServer** using the AWS Systems Manager Session Manager. Systems Manager allows you to perform many different operations on Amazon EC2 instances after installing the service agent. The Session Manager feature allows you to connect directly to an instance without requiring you to open up port 22 for SSH. You can connect with Session Manager using either a URL through a browser, as you have been doing in this lab, or through the AWS CLI.

Using Session Manager is an alternative to setting up and maintaining bastion hosts. This lets AWS do the heavy lifting of securing the connection and managing a secure, redundant, bastion host configuration. This also gives you the ability to use AWS Identity and Access Management (IAM) to grant access to users and creates an auditable logging trail when integrated with AWS CloudTrail and Amazon CloudWatch.



Use what you have learned in this lab to do the following:

* Remove the inbound **SSH** rule from the **AppServerSecurityGroup**.
* Connect to **BastionHost** and attempt to SSH to **AppServer**.
* Use the **AppServerSessionManagementUrl** from the left side of these instructions to connect directly to **AppServer**.

**Note** For help modifying the **AppServerSecurityGroup**, refer to the steps in Task 1.

**Conclusion**

 Congratulations! You now have successfully:

* Examined security groups and determined what traffic is allowed.
* Changed which security groups are applied to Amazon EC2 instances.
* Updated security groups to follow the principle of least privilege.
* Understood how security groups can reference other security groups.
* Understood how to leverage Session Manager to connect to instances.

**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.

**Additional Resources**

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).

**Connection assistance notes**

CONNECT USING SYSTEMS MANAGER

Systems Manager allows administrators to grant IAM users the ability to create browser-based terminal sessions. This allows authenticated users to have one-click access to approved instances. This helps improve access by removing the need to download/share PEM keys and offering easy access to the server to anyone who does not have a terminal program (such as PuTTY).

**Note** You need to be signed in to the AWS Management Console before you can establish a session.

1. Copy the **BastionHostSessionManagementUrl**, **PublicServerSessionManager**, or **AppServerSessionManagementUrl** value from the left side of the lab page, and paste it in a new browser tab. The command host terminal appears.

**Note** If you are having difficulty using Systems Manager, ask your instructor for help or try one of the other connection options.

You can also alter your command prompt to make command output easier to read by exporting the PS1 variable. To do this, run the following command:

export PS1="\n[\u@\h \W] $ "

1. Once connected, [click here](https://labs.skillbuilder.aws/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A470679935125%3Ablueprintversion%2Fspl-255%3A1.0.9-ff293c68/en-US#ssh-after) to return to the lab instructions.

CONNECT TO A LINUX INSTANCE FROM WINDOWS USING PUTTY

**Note** Only perform the following steps if you are connecting from a Windows machine. If you are connecting from a macOS or Linux machine, [click here](https://labs.skillbuilder.aws/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A470679935125%3Ablueprintversion%2Fspl-255%3A1.0.9-ff293c68/en-US#ssh-osx) for instructions.

1. On the left side of the lab page, click  **Download PPK**. Save the PPK file to the directory of your choice.
2. Open PuTTY (from the **Start** menu, choose **PuTTY** > **PuTTY**).

**Note** If PuTTY is not already installed on your computer, download and install it from the following URL: <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>. If you already have an older version of PuTTY installed, we recommend that you download the latest version.

1. In the **Category** pane, choose **Session** and configure the following:

* For **Host Name**, enter

<user\_name>@<public\_dns\_name>

, where

<public\_dns\_name>

 is the **Public DNS (IPv4)** name found on the EC2 instance description tab for the instance you are attempting to connect to.

**Note** For Amazon Linux 2 or the Amazon Linux AMI, the user name is

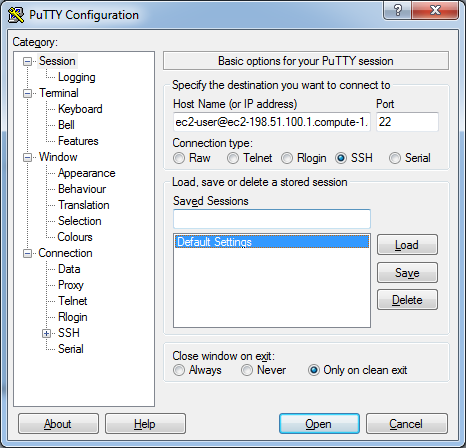
ec2-user

. For an Ubuntu AMI, the user name is

ubuntu

.

* For **Connection type**, select **SSH**
* Ensure that the **Port** value is **22**



1. (Optional) You can configure PuTTY to automatically send ‘keepalive’ data at regular intervals to keep the session active. This is useful to avoid disconnecting from your instance due to session inactivity. In the **Category** pane, choose **Connection**, and then enter the required interval in the **Seconds between keepalives** field. For example, if your session disconnects after 10 minutes of inactivity, enter 180 to configure PuTTY to send keepalive data every 3 minutes.
2. In the **Category** pane, expand **Connection**, expand **SSH**, and then choose **Auth**. Complete the following:

* Choose **Browse**.
* Select the .ppk file that you downloaded earlier, and choose **Open**.

**Note** This .ppk file is usually located in the **Downloads** folder on your PC.

* (Optional) If you plan to start this session again later, you can save the session information for future use. Under **Category**, choose **Session**, enter a name for the session in **Saved Sessions**, and then choose **Save**.
* Choose **Open**.

1. If this is the first time you have connected to this instance, PuTTY displays a security alert dialog box that asks whether you trust the host to which you are connecting. Choose **Yes**. A window opens and you are connected to your instance.

**Note** If you receive an error while attempting to connect to your instance, see [Troubleshooting Connecting to Your Instance](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/TroubleshootingInstancesConnecting.html).

You are now connected to your instance. For the remainder of the lab, use the SSH terminal window to enter the commands in the lab instructions.

1. Once connected, [click here](https://labs.skillbuilder.aws/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A470679935125%3Ablueprintversion%2Fspl-255%3A1.0.9-ff293c68/en-US#ssh-after) to return to the lab instructions.

CONNECT TO A LINUX INSTANCE FROM MACOS OR LINUX

**Note** Only perform the following steps if you are connecting from a macOS or Linux machine. If you are connecting from a Windows machine, [click here](https://labs.skillbuilder.aws/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A470679935125%3Ablueprintversion%2Fspl-255%3A1.0.9-ff293c68/en-US#ssh-windows) for instructions.

1. On the left side of the lab page, click  **Download PEM**. Save the file to the directory of your choice.
2. Open a terminal window on your local computer.

**Note** Your local computer most likely has an SSH client installed by default. You can check for an SSH client by typing

ssh

 at the command line. If your local computer doesn’t recognize the command, you can install an SSH client. For information about installing an SSH client on Linux or macOS X, see [https://www.openssh.com](https://www.openssh.com/).

Complete the remaining connection steps in the terminal window.

1. Change the directory to the folder where you downloaded the PEM file.

**Note** The PEM file is usually located in the **Downloads** folder on your computer. Access this directory by typing

cd ~/Downloads

1. Your key must not be publicly viewable for SSH to work. Change the permissions on the PEM file by running the following command. Replace

<PEM\_FILE>

 with the name of the PEM file you downloaded:

chmod 400 <PEM\_FILE>

1. Log in to the remote instance by running the following command. Replace

<PEM\_FILE>

 with the name of the PEM file you downloaded,

<user\_name>

 with the user name for the instance type you are connecting to, and

<public\_dns\_name>

 is the **Public DNS (IPv4)** name found on the EC2 instance description tab for the instance you are attempting to connect to:

ssh -i <PEM\_FILE> <user\_name>@<public\_dns\_name>

**Note** For Amazon Linux 2 or the Amazon Linux AMI, the user name is

ec2-user

. For an Ubuntu AMI, the user name is

ubuntu

.

1. If this is the first time you have connected to this instance, you see a response similar to the following:

The authenticity of host 'ec2-192-0-2-111.compute-1.amazonaws.com (192.0.2.111)'

can't be established.

RSA key fingerprint is 1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f.

Are you sure you want to continue connecting (yes/no)?

1. When prompted, enter

yes

You are now connected to your instance.

**Note** If you receive an error while attempting to connect to your instance, see [Troubleshooting Connecting to Your Instance](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/TroubleshootingInstancesConnecting.html).

You are now connected to your instance. For the remainder of the lab, use the SSH terminal window to enter the commands in the lab instructions.

1. Once connected, [click here](https://labs.skillbuilder.aws/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A470679935125%3Ablueprintversion%2Fspl-255%3A1.0.9-ff293c68/en-US#ssh-after) to return to the lab instructions.