Introduction

Creating an architecture that includes cloud components brings some challenges to the design that have traditionally been less important. In a traditional web application, the servers and storage are provided by machines that are connected by a tightly controlled network. The client machines typically connect to load-balancing machines that provide the only external connection to the public internet. In this architecture, securing the system involves managing the network security and firewall rules at the entry point.

In cloud-based architectures, some of the machines that provide the service are provisioned from cloud providers. This can include the load balancers, origin servers, database servers, and monitoring systems. It is possible, especially when using a single vendor, to create a virtual network in the cloud that connects all of your machines. In this case, security can be managed in a fashion similar to the traditional model. The virtual network is treated as a secure zone, and external access to it is tightly controlled. However, even in this scenario, you are placing considerable trust in a single cloud vendor. This can lead to problems in the future if the vendor does not support changes that you want to make.

An alternative approach for designing cloud-based systems is to use a zero-trust model. In this model, all cloud-based resources require credentials to gain access. All communication channels that pass through the public internet are encrypted. This creates some unique challenges. Requiring credentials for access means that you need to have a method of managing those credentials. The choice of an identity and access management system will depend on your environment. For the purposes of this lab, this component of the design has been left out. Some of the tools for securing access in the cloud network will be covered in a subsequent lab.

You are a member of a team that is designing the initial transition from a traditional server cluster to a cloud-based architecture. The primary motivation is to be able to scale the number of client-facing servers with demand. The existing application is suitable for running in Docker containers, so that has been selected as the initial transition point. You will provision machines from various cloud providers and use them to run the Docker containers. You will also have a private Docker registry that holds the images to be used for the containers. The registry and persistent storage for the application will be kept in-house for the first stage.

Lab Overview

This lab has **three** parts, which should be completed in the order specified.

1. In the first part of the lab, you will create a diagram of the basic network architecture.

2. In the second part of the lab, you will use SCAP Workbench to scan a machine.

3. In the third part of the lab, you will set up Wazuh to scan machines automatically.

Finally, you will explore the virtual environment on your own. You will answer questions and complete challenges that allow you to use the skills you learned in the lab to conduct independent, unguided work – similar to what you will encounter in a real-world situation.

Learning Objectives

Upon completing this lab, you will be able to:

1. Create cloud architecture diagrams.

2. Scan machines using SCAP documents.

3. Customize a SCAP profile.

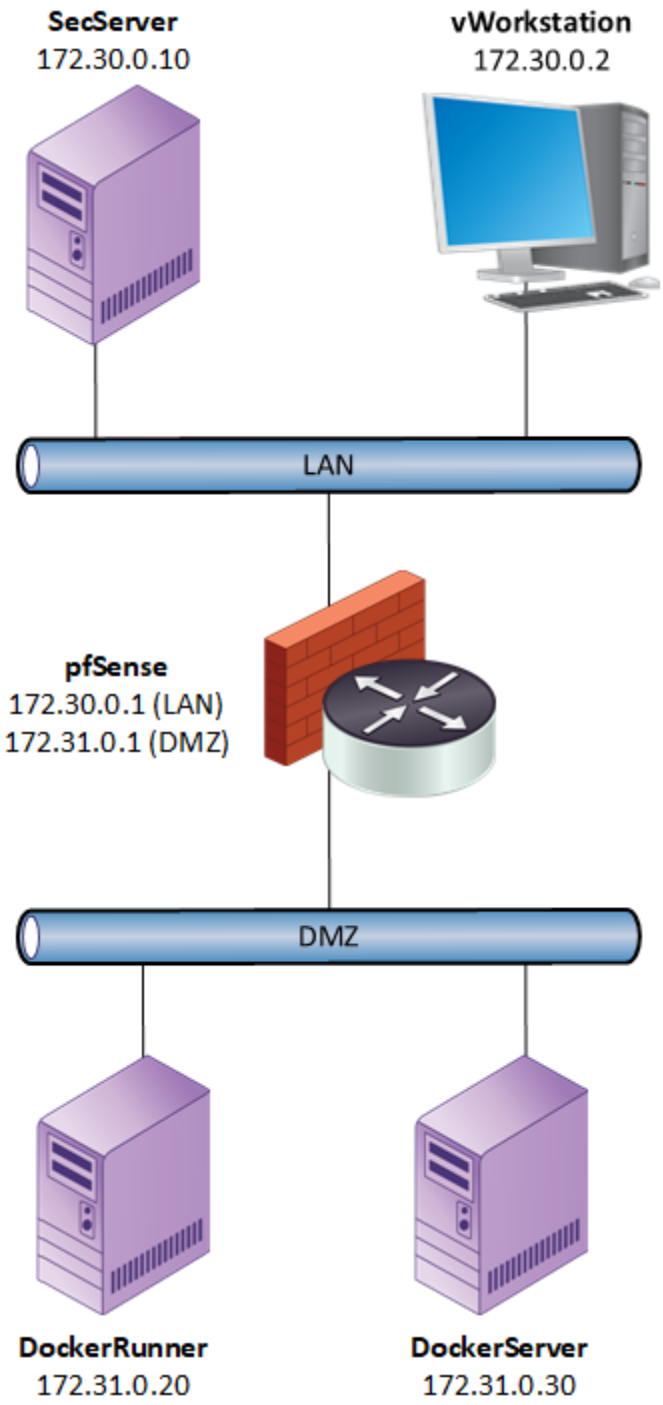
4. Enable SCAP scans in Wazuh.

5. Enable Docker scans in Wazuh.

Topology

This lab contains the following virtual machines. Please refer to the network topology diagram below.

- vWorkstation
- SecServer
- pfSense
- DockerServer
- DockerRunner



Tools and Software

The following software and/or utilities are required to complete this lab. Students are encouraged to explore the Internet to learn more about the products and tools used in this lab.

- Dia
- SCAP Workbench
- Docker
- Wazuh

Deliverables

Upon completion of this lab, you are required to provide the following deliverables to your instructor:

Hands-On Demonstration

1. Lab Report file, including screen captures of the following:

- the network diagram for the system
- the Disable SSH Access via Empty Passwords test failed
- the Disable SSH Access via Empty Passwords test passed
- an event in Wazuh for Disable SSH Access via Empty Passwords failing on DockerRunner
- an event about the alpine image being pulled
- an event about a container being started

2. Any additional information as directed by the lab:

- None

Challenge and Analysis

1. Lab Report file, including screen captures of the following:

- the results in OpenSCAP Workbench after using your new profile to scan DockerRunner
- the new group configuration file for the default group in Wazuh

2. Any additional information as directed by the lab:

- None

Hands-On Demonstration

Note: In this section of the lab, you will follow a step-by-step walk through of the objectives for this lab to produce the expected deliverables.

1. **Review** the **Tutorial**.

   Frequently performed tasks, such as making screen captures and downloading your Lab Report, are explained in the Cloud Lab Tutorial. The Cloud Lab Tutorial is available from the User menu in the upper-right corner of the Student Dashboard. You should review these tasks before starting the lab.

2. **Proceed** with **Part 1**.

Part 1: Planning a Design

Note: A secure cloud infrastructure does not happen by accident. It takes planning. There is also no single recipe for all deployments. The requirements of your situation will guide your choices. However, there are some global principles that should be remembered when designing the system.

- Access to all resources must be controlled.
- Access to all resources should be monitored to allow auditing.
- Systems should be constantly monitored.
- All data needs to be protected at rest and in transit.
- Configuration of on-demand machines needs to be managed.

While these are all good principles for a traditional self-managed data center, several of them take on new significance in a cloud environment. The traditional approach creates a secure perimeter that allows machines inside the perimeter to be treated as "trusted" to a certain extent. With cloud-based systems, creating and maintaining a secure perimeter can be very complex. It is often simpler to adopt a zero-trust strategy in which all resource access requires credentials and all communication is encrypted.

It is also important to be able to record and report design choices made for your system. One very useful method for designing networked systems is a network diagram. The open-source program Dia

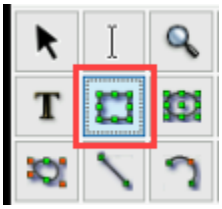
is a tool that can be used to create diagrams describing the architecture.

1. On the vWorkstation desktop, **double-click** the **Dia icon** to open the diagramming tool.

Dia icon

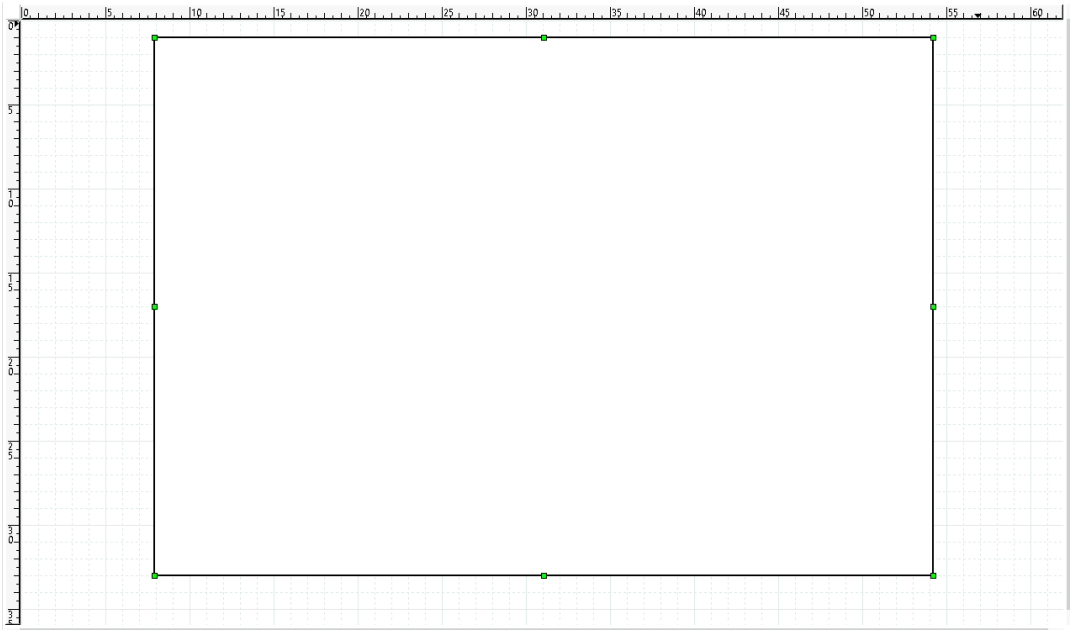
Note: The tool has a drawing area with a grid and a left-side panel with a tool palette and a symbol palette. There is a drop-down menu of the available symbol pallets. In this case, you will use the Network palette.

2. In the tool palette, **click** the **Box icon** to activate the box tool.



Box icon

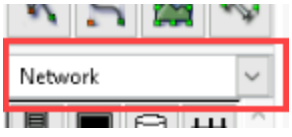
3. In the drawing area, **click-and-drag** to create a rectangle that covers about two-thirds of the width of the drawing area and most of the height.



Draw a box to indicate a logical network boundary

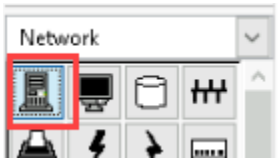
Note: This rectangle will represent a logical boundary in the system design, and will serve as a starting point for your architectural blueprint. Machines under your control will be inside the boundary, and machines outside your control will be outside.

4. In the symbol palette drop-down in the left side panel, **select Network** to use the network symbols.



Select Network in the symbol palette drop-down

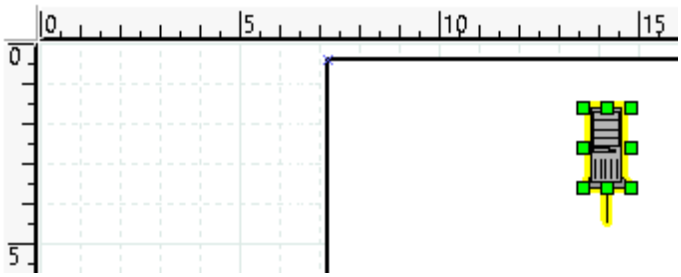
5. In the network symbol palette, **click** the **computer icon** to select it for placement.



Computer icon

6. In the drawing area, **click a point near the upper left of the rectangle** to place a computer icon.

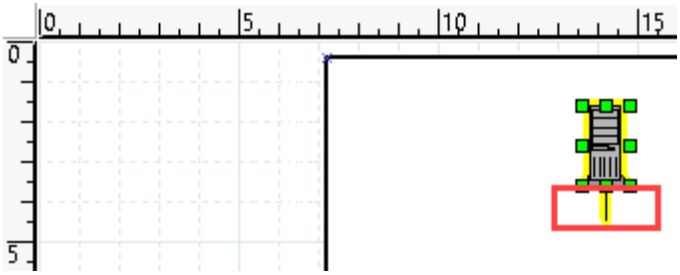
   Leave enough room between this computer and the left edge of the rectangle to place another machine, which you will do in a later step.



Place a computer icon near the upper left of the rectangle in the Dia drawing area

Note: There should be a cursor below the icon after it is placed. If you do not see the cursor, you can select the icon in the drawing area and press Enter.

7. In the label for the computer icon, **type** `Registry` to label the symbol.

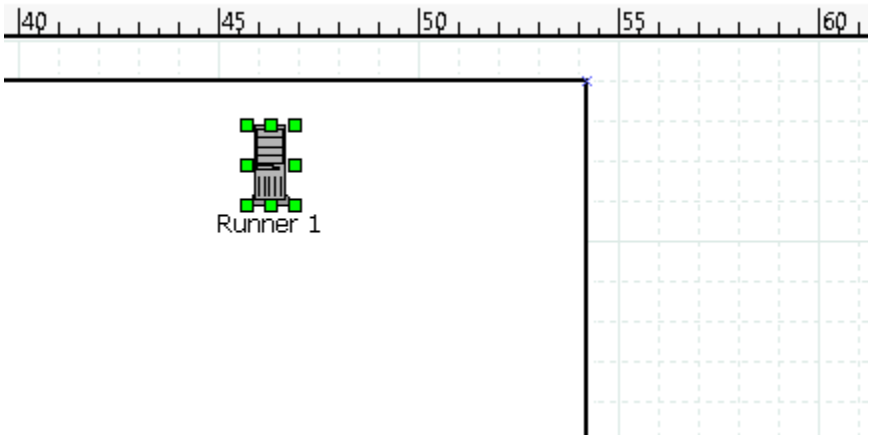


Type Registry to label the computer icon

Note: This computer represents the registry that stores the Docker images for your system. Placing it inside the rectangle means that it is within your domain of control.

In the following steps, you will place three additional machines within this domain of control. Leave enough room between these machines and the right edge of your rectangle to add additional machines, which you will do in a later step.

8. **Repeat steps 5 to 7** to place a computer icon labeled Runner 1 near the upper right of the rectangle.



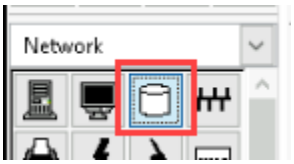
Add Docker Runner 1 near the upper-right of the logical network rectangle

9. **Repeat steps 5 to 7** to place a computer icon labeled Runner 2 below Runner 1.

10. **Repeat steps 5 to 7** to place a computer icon labeled Runner N below Runner 2.

Note: These symbols represent machines that will run Docker containers. This is the part of the system that you want to be dynamically scalable. This is indicated on the diagram by the choice of labeling.

11. In the network symbol palette, **click** the **Storage icon** to select it for placement.



Storage icon

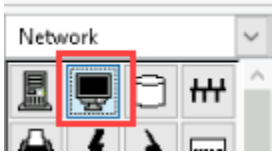
12. In the drawing area, **click a point below the Registry icon** to place the Storage icon.

    Leave enough vertical space to place another machine between these two, which you will do in a later step.

13. Below the Storage icon, **type** `Storage` to label the symbol.

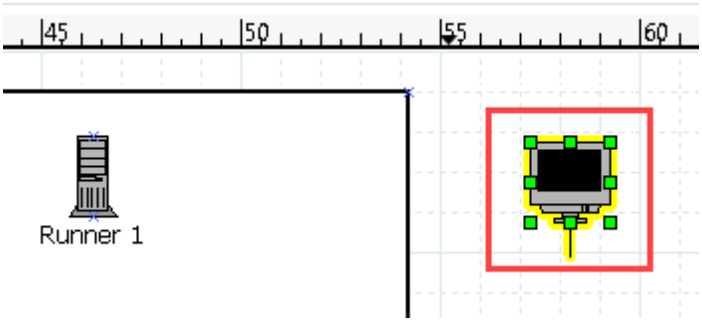
Note: The persistent storage is going to be kept in the current private network. Thus, this symbol represents a machine (or cluster of machines) that will provide persistent storage to the Docker containers.

14. In the network symbol palette, **click** the **Monitor icon** to select it for placement.



Monitor icon

15. In the drawing area, **click a point to the right of the rectangle** to place the icon.

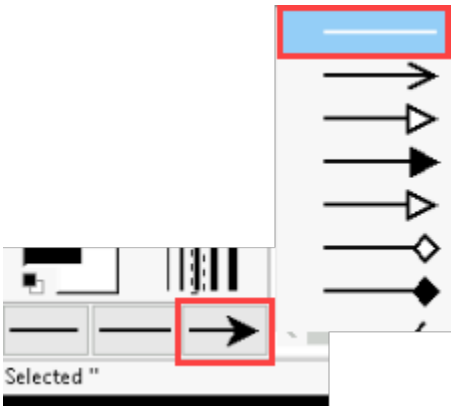


Place the monitor icon to the right of the logical network rectangle

16. Below the Monitor icon, **type** `Client` to label the symbol.

Note: Here, you use a different symbol to represent a machine that typically has an active user. The clients of your application will be served by one of the Docker runners. So far the diagram just has symbols to represent machines and a single boundary. You also want to show important connections.

17. At the bottom of the side panel, **click** the **end arrow style icon** and select the None option to set the arrow style for new lines.

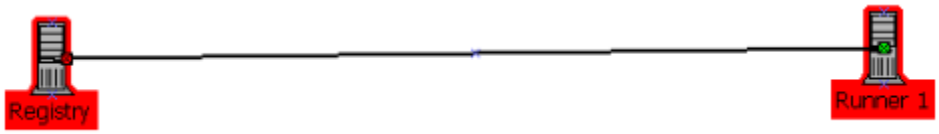


Click the end arrow style icon and select the None option

18. In the tool palette, **click** the **Line icon** to activate the line tool.

Line icon

19. In the drawing area, **click-and-drag from the Registry symbol to the Runner 1 symbol** to draw a line between the two symbols.



Connect the Registry and Runner 1 symbols by clicking-and-dragging the Line icon between them

20. **Repeat steps 18 and 19 two times** to draw lines from Registry to Runner 2 and Runner N.

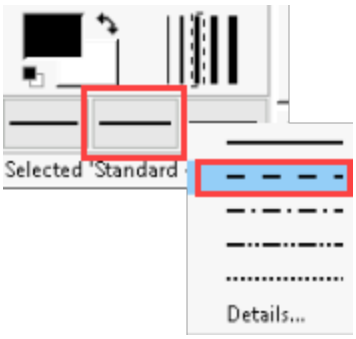
Note: These lines represent the communication that must occur between the Docker runners and the Docker registry.

21. **Repeat steps 18 and 19 three times** to draw lines between Storage and each of the Docker runners.

Note: These lines represent the communication between the Docker containers and the persistent

storage.

22. At the bottom of the side panel, **click** the **line style selector** and **select** the **dashed line** to set the line style for new lines.



Click the line style selector and select the dashed line option

23. **Repeat step 18 and 19 three times** to draw dashed lines between the client and the Docker runners.

Note: The diagram now communicates several important pieces of information.

- There is a network boundary containing machines under your control.
- There is a single registry server for Docker images.
- There is a single storage server for persistent storage.
- There are multiple machines for running Docker containers.

- Clients can be served by any of the Docker runners.

However, there is currently little in the diagram, except perhaps the boundary, that outlines the security aspects of the design. Another important part of security architecture is monitoring. In the following steps, you will add a monitoring server within your domain of control, along with additional lines to indicate the machines with which it needs to communicate.

24. In the network palette, **click** the **Computer icon** to activate it for placement.

25. In the drawing area, **click** a **point between Registry and Storage** to place the symbol.

26. In the label below the icon, **type** `Monitoring` to label the symbol.

Note: This symbol represents a machine to be used for monitoring all the other machines in your architecture. There should be a connection shown to each machine that is monitored. To make it clear that the connection is different from the application data paths, you will use a different color.

27. In the sidebar, **click** the **color selector** to open the Select foreground color dialog.

Color selector button

28. In the color dialog, **click** the **bright green box** and **click OK** to set the color for new objects.

Color dialog

29. In the sidebar, **click** the **line style selector** and **select** the **solid line**.

30. In the drawing area, **use** the **line tool** to draw lines between Monitoring and every other machine in the rectangle.

Note: This is about the extent of what the lab environment models, except for the storage machine. You will use the other machines later to see an example of how monitoring can work. However, this is far from a complete architecture for our system. What about development? For your current use case, the developer's task is primarily creating images to be run on the Docker runners. You will use a single symbol to represent a developer machine. It is also good to use a different color for the connection used for deploying images.

31. In the sidebar, **use** the **foreground color selector** to set the foreground color back to black.

32. In the drawing area, **use** the **Monitor Icon** to place a Monitor Symbol in the upper-left corner of the rectangle and label it Developer.

33. In the sidebar, **use** the **foreground color selector** to set the foreground color to blue.

34. In the drawing area, **use** the **line tool** to draw a line between Developer and Registry.

Note: Your system design also requires a load balancer between the clients and the Docker containers to distribute client requests. This should be indicated in the diagram.

35. In the sidebar, **use** the **foreground color selector** to set the foreground color back to black.

36. In the drawing area, **use** the **Computer icon** to place a Computer Symbol between the Docker Runners and the Client machine and label it Load Balancer.

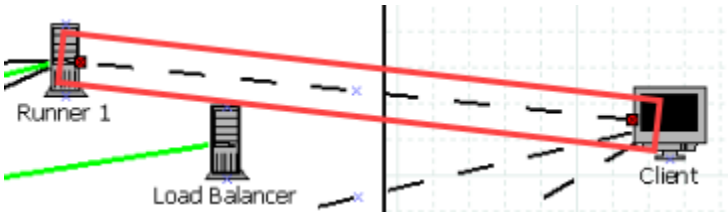
Note: This load balancer should also be monitored actively.

37. In the sidebar, **use** the **foreground color selector** to set the foreground color to green.

38. In the drawing area, **use** the **line tool** to draw a line between the Monitor machine and the Load Balancer.

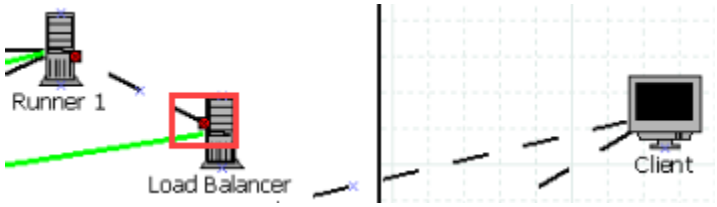
Note: You also need to rearrange the connections from the client.

39. In the drawing area, **click** on the **line from Runner 1 to Client** to select it.



Select the dashed line connecting Runner 1 and Client

40. **Click-and-drag** the **red box from Client to Load Balancer** to change the line connection.



Runner 1 symbol is now connected by a black dashed line to Load Balancer instead of Client

41. **Repeat steps 39 and 40** for the lines between the other Docker runners and the client machine.

42. In the sidebar, **use** the **line style selector** to set the line style to dashed.

43. In the drawing area, **use** the **line tool** to draw a dashed line between Load Balancer and Client.

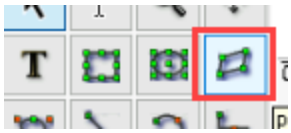
Note: Now the diagram more accurately reflects the actual deployment. Client machines will connect to the load balancer. The load balancer will use the Docker runners as origin servers. At this point, the diagram does not really indicate any use of cloud-based resources.

Your team has decided to start by using cloud-provisioned machines for running Docker containers. To update the diagram to indicate the use of cloud resources, you should change how the boundaries are drawn.

44. In the drawing area, **shrink** the **rectangle** to contain only the Docker runner machines.

Note: This rectangle now represents the cloud-based resources. Since your initial foray into cloud-based services is keeping the rest of the parts inside your current private system, you need to draw a border around all the other machines that were in the rectangle.

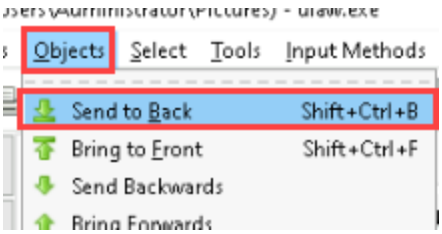
45. In the tool palette, **select** the **polygon tool** to activate it.



Polygon tool icon

46. In the drawing area, **click** a **point above and to the left of the Developer machine** to place the polygon.

47. In the menu bar, **select Objects > Send to Back** to make the polygon go behind all the other objects.

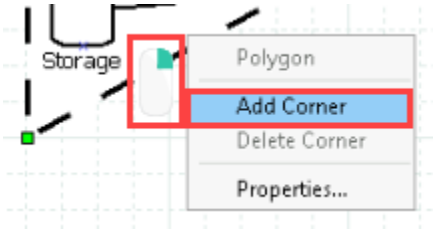


Objects menu

48. In the drawing area, **click-and-drag** the **rightmost green box of the polygon horizontally** so that it is to the right of the load balancer location.

49. In the drawing area, **click-and-drag** the **bottom green box of the polygon vertically** so that it is below all the machines.

50. In the drawing area, **right-click** on **the diagonal edge of the polygon** and **select Add Corner** to add a new corner to the polygon.



Polygon context menu

51. In the drawing area, **click-and-drag** the **new corner** so that the polygon forms a rectangle around all the machines except the client machine.

52. In the drawing area, **use Add Corner four more times** and **position the corners** so that the polygon is shaped like an upside-down U around the Docker Runners and contains all the non-Runner machines except the client machine.

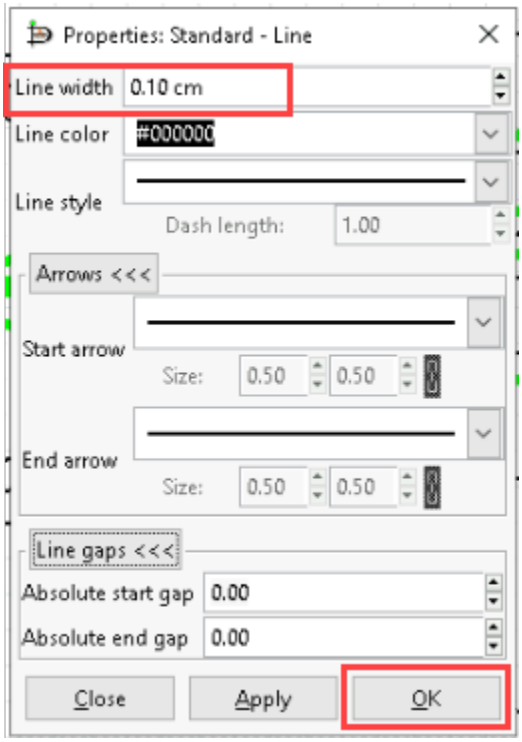
Note: The diagram now has features that show some more important points.

- The machines running the Docker containers are in a separate group.
- The clients will talk to a load-balancing machine.
- The other machines are organized into a single group as well.

You still need to provide a way of indicating that some connections need access control and data protection.

53. In the drawing area, **double-click** on the **line between Registry and Runner 1** to open the Properties dialog.

54. In the Properties dialog, **change** the **line width** to 0.30 cm to make it thicker and then click OK.



Line Properties dialog

55. **Repeat steps 53 and 54** to make the lines between the Docker runners and Registry, Monitoring, and Load Balancer thicker.

Note: While you may know what the different colors and line weights mean, the diagram should have notes to indicate this to a reader.

56. In the tool palette, **click** the **text tool** to activate it.

Text tool icon

57. In the drawing area, **click** a **point below the rectangle** to place a text label.

58. In the label, **type** `Cloud Provisioned` to label the region containing the cloud machines.

59. **Repeat steps 56 through 58** to label the polygon Private Network.

60. **Repeat steps 56 through 58** to create a label to the left of the diagram using black text that says “Data flow.”

61. **Use** the **color selector** to change the foreground color to bright green.

62. **Repeat steps 56 through 58** to create a label below Data flow that says “Monitoring” using green text.

63. **Repeat steps 61 and 62** to create a label below Monitoring that says “Deployment” using blue text.

64. **Repeat steps 61 and 62** to create a label below the others that says the heavy lines represent connections that require access control and encryption.

65. **Make a screen capture** showing the network diagram for the system with the labels.

66. **Close** the **Dia window**.

Note: This is a reasonable point to work from for the rest of this lab. However, it is not complete for a full production environment. You have not addressed production versus staging. For the final system, this diagram could serve as the staging diagram and the clients would become test clients and test attack systems. Then a second, very similar diagram would show the production network. The production network would not contain a developer machine. Instead, there would be a connection from the staging registry to the production registry.

It also does not include anything about identity and access management. While this is a key component of a secure design, the choice of a system is strongly dependent on the details of the services that must be supported. Many times, enterprise-level systems will be purchased from a vendor, which would also come with support for installing and managing the system. There are also some open-source IAM systems available, including Open Identity Platform, OpenIAM, and gluu.

Part 2: Automating Security

Note: While having a well-designed architecture is important, it needs to be implemented in a way that allows security policies to be enforced and monitored. A key component of a secure architecture is including ways to automatically check the active systems to make sure that the policies are being followed. This requires a method of specifying the policies as machine-readable files that can be used by a monitoring tool.

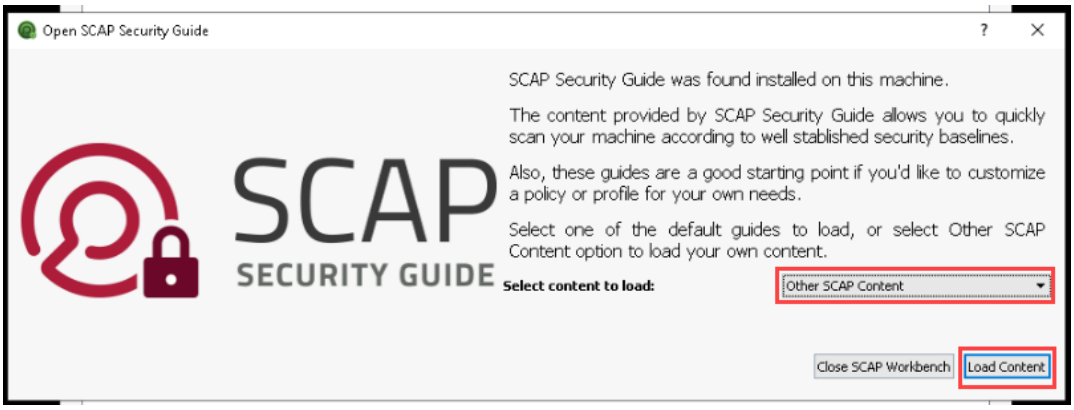
The OpenSCAP system provides a suite of tools that can help with the automated monitoring of systems. The Security Content Automation Protocol (SCAP) is a NIST-certified standard for automated configuration, vulnerability and patch checking, and compliance checking. One of the products of this system is the SCAP Security Guide, which provides files that define many security policies. The files define tests that perform a single check on a machine and that can be organized into sets with an identifier. It is also possible to define custom sets of tests.

The SCAP Workbench is a desktop program that can be used to perform scans defined in SCAP files.

1. On the vWorkstation desktop, **double-click** the **SCAP Workbench Icon** to open the program.

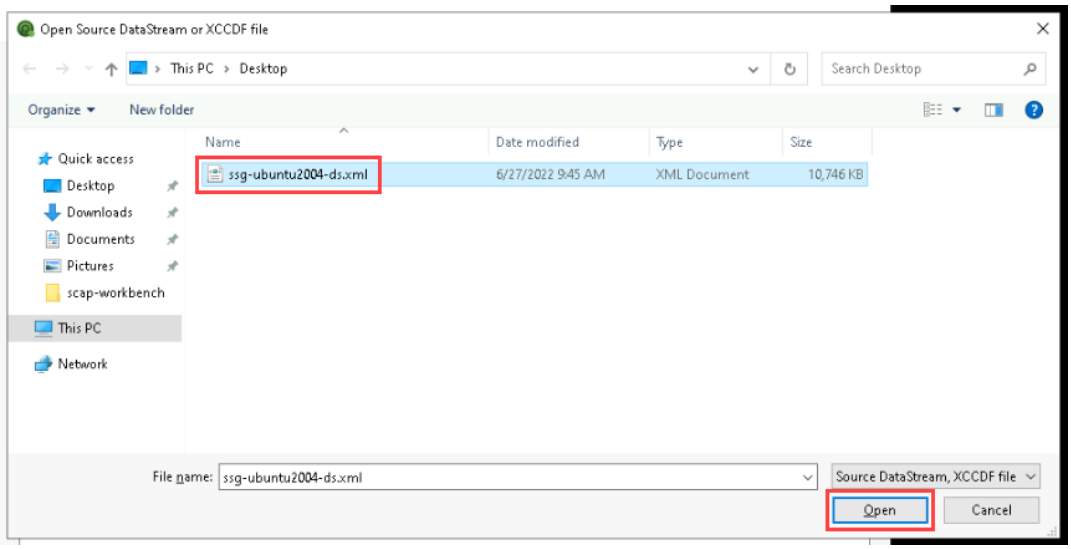
SCAP Workbench icon

2. In the Open SCAP Security Guide dialog, **choose Other SCAP Content** and **click Load Content** to load the Open Source dialog.



Open SCAP Security Guide dialog box

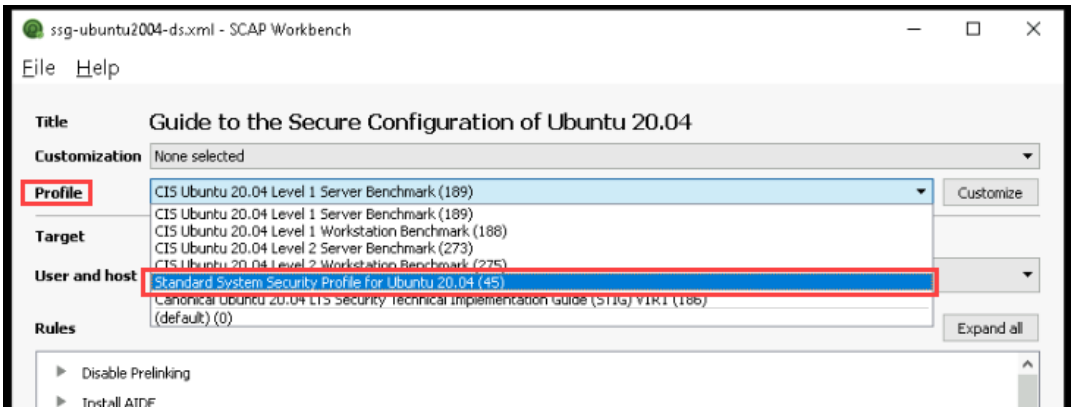
3. In the Open Source dialog, **choose ssg-ubuntu2004-ds.xml** from the Desktop and **click Open**.



Open Source Datastream dialog box

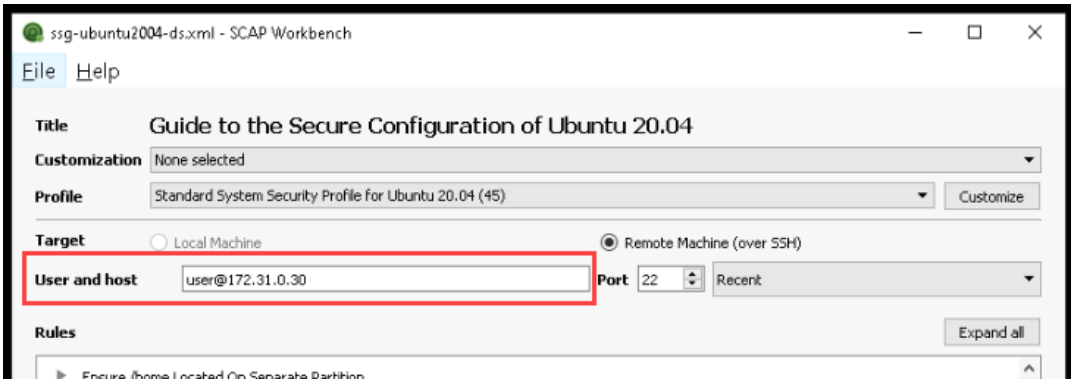
Note: The SCAP content was created by the Open SCAP project as part of the Open SCAP Security Guide. The file is specialized for Ubuntu 20.04 and has multiple profiles defined with sets of scans to be used. For testing purposes in this lab, you will use a simple standard profile.

4. In the Profile drop-down, **select Standard System Security Profile for Ubuntu 20.04**.



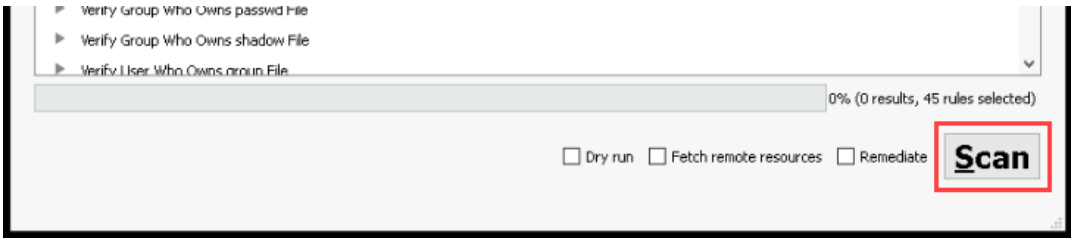
SCAP Workbench dialog box Profile drop-down

5. In the User and host field, **type** `user@172.31.0.30` to give the SSH connection information.



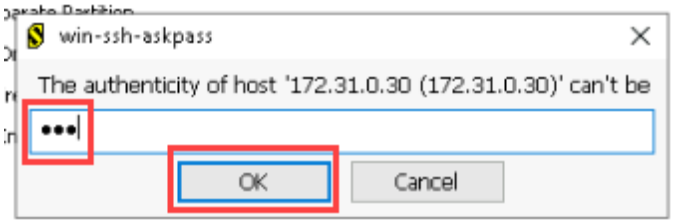
SCAP Workbench dialog box User and host field

6. At the bottom of the window, **click Scan** to start the SCAP scan.



SCAP Workbench dialog box Scan button

7. In the first dialog, **type** `yes` and **press Enter** to accept the host key.



win-ssh-askpass first dialog box

Note: During the scan, the workbench connects to the target machine using SSH. The windows version of SCAP Workbench comes with a built-in SSH client program. This program needs to be told to accept the host key and also needs to be given the password for each scan.

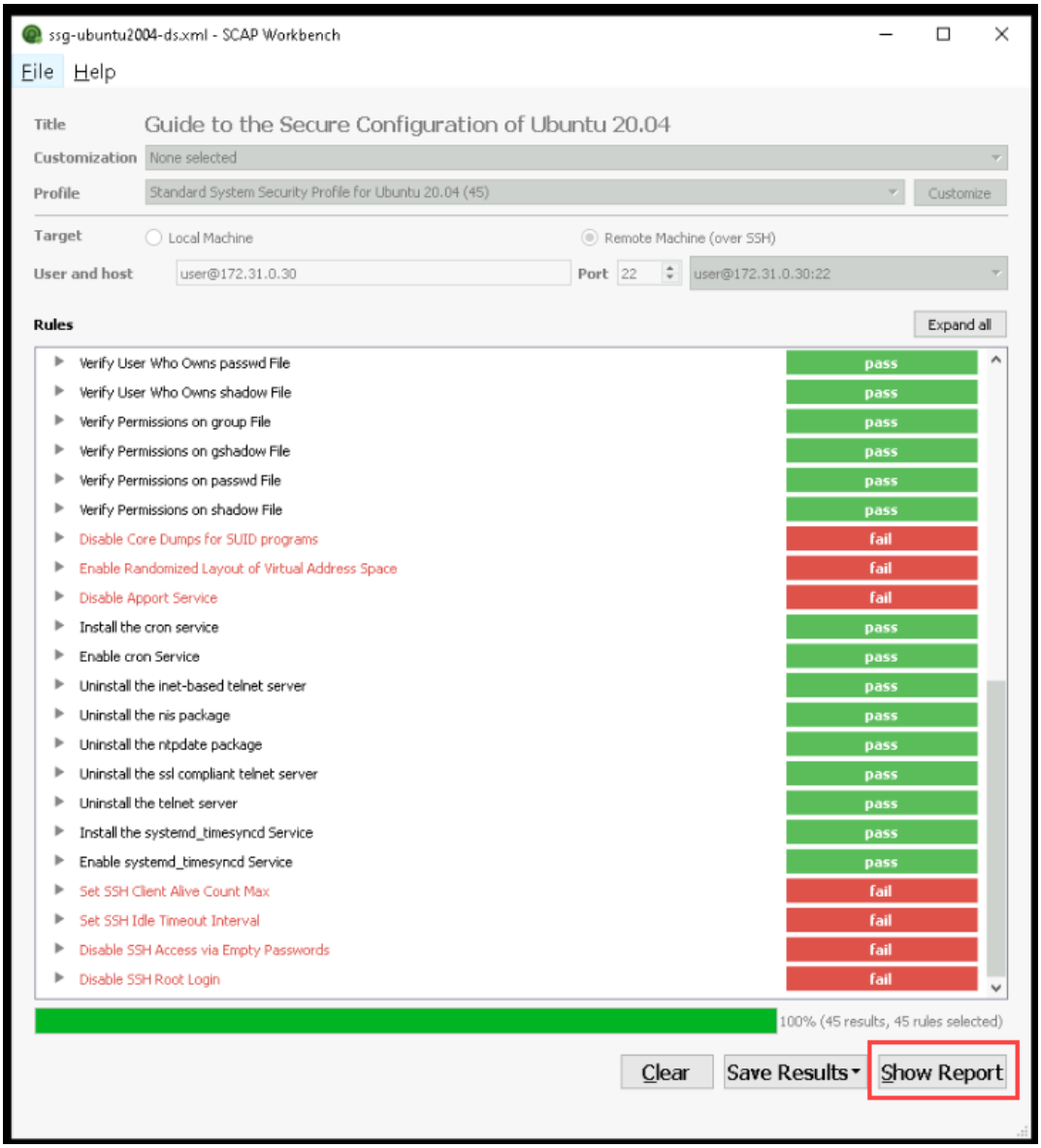
8. In the second dialog, **type** `password` and **press Enter** to provide the password for user.

win-ssh-askpass second dialog box

Note: The results will populate as the scan is running. When the scan is done, a diagnostics window will be shown with log entries from the process. For your current testing, the diagnostics are not important.

9. **Close** the **Diagnostics window**.

10. At the bottom of the SCAP Workbench window, **click Show Report** to open the report in a browser.



SCAP Workbench dialog box Show Report button

Note: You will see quite a few failed items from this scan. To see what caused a failure, you can view the details from each test in the report.

11. In the browser window, **scroll down** to find **Disable SSH Access via Empty Passwords**.

Disable SSH Access via Empty Passwords rule

12. **Click** the **name** to view details about the result.

Note: You should see a popup dialog in the browser with details about the rule. In the OVAL Details section, there is a table with columns Filepath and Pattern. The Pattern is a regular expression used to check the sshd_config file. If the pattern matches, then the test passes. If the pattern does not match, then the test fails.

In this case, the regular expression is looking for a line with "PermitEmptyPasswords no" in the file /etc/ssh/sshd_config. You can also do a similar search manually.

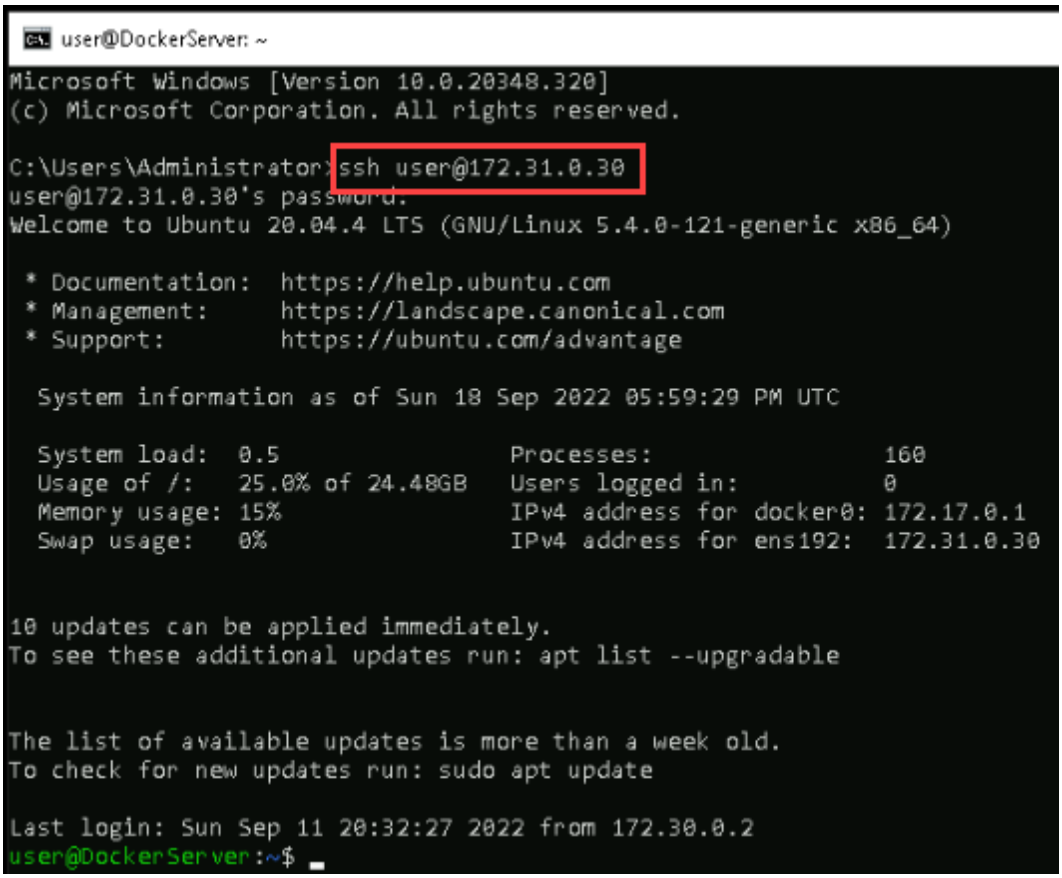
13. **Make a screen capture** showing the **OVAL details of the PermitEmptyPasswords failure** from the manual scan.

14. On the vWorkstation desktop, **double-click** the **Command Prompt icon** to open a new command prompt window.

Command Prompt icon

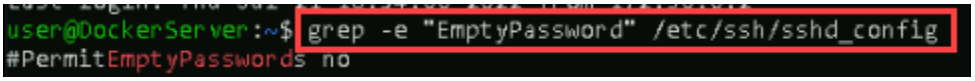
15. At the command prompt, **type** `ssh user@172.31.0.30` and **press Enter** to connect to the machine.

    When prompted, **type** `password` and **press Enter**.



Type ssh user@172.31.0.30

16. At the shell prompt, **type** `grep -e "EmptyPassword" /etc/ssh/sshd_config` and **press Enter**.

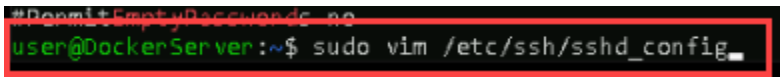


Type grep -e "EmptyPassword" /etc/ssh/sshd_config

Note: You should see a line #PermitEmptyPasswords no. The configuration line is commented out. This makes it fail the test, which is a bit of a false positive. The standard sshd_config file shows default configurations as commented-out lines. In this case, the default configuration is no, which is what the test requires. However, the test requires the configuration to be explicit. You can make the test pass by removing the # in the line.

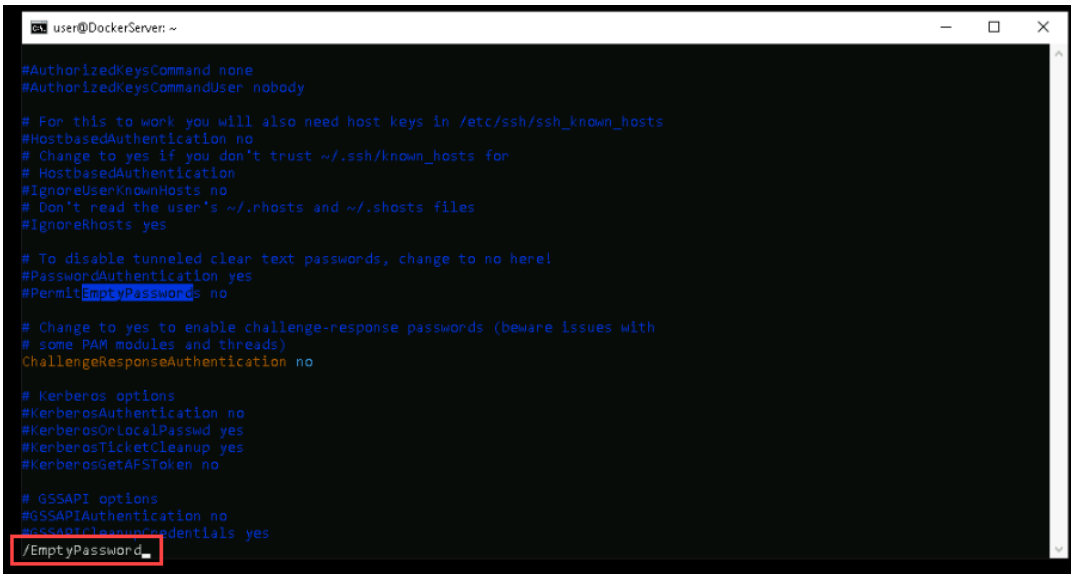
17. At the shell prompt, **type** `sudo vim /etc/ssh/sshd_config` and **press Enter** to edit the configuration file.

    If prompted, **type** `password` and **press Enter**.



Type sudo vim /etc/ssh/sshd_config

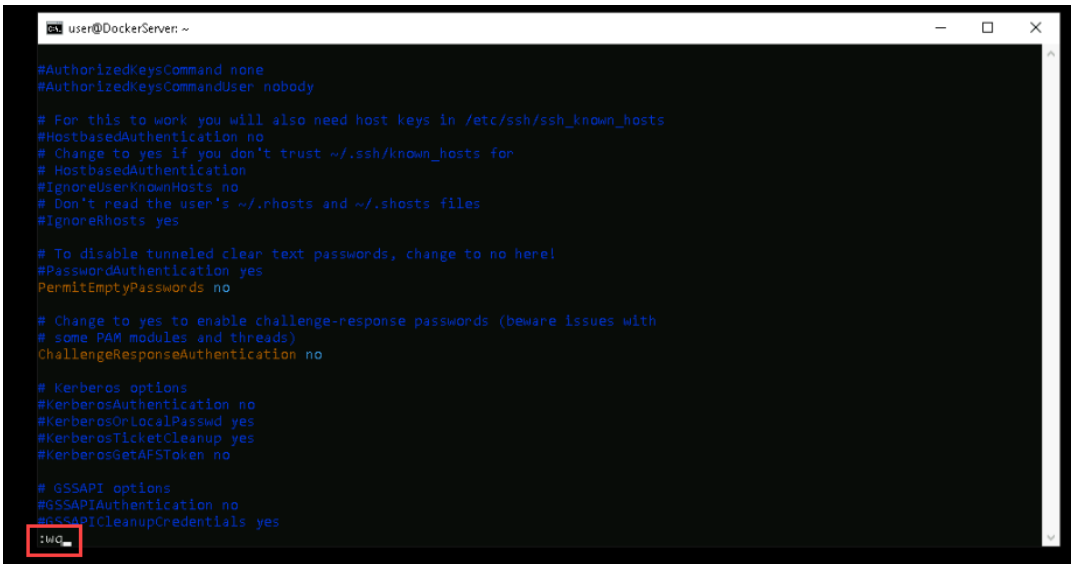
18. In the editor, **type** `/EmptyPassword` and **press Enter** to find the line.



Type /EmptyPassword

19. In the editor, **type ^ then x** to go to the beginning of the line (^) and delete (x) the # character.

20. In the editor, **type `:wq`** to save (w) and exit (x) the editor.



Type :wq

21. At the shell prompt, **type** `exit` and **press Enter** to log out.

22. **Close** the **command prompt window**.

23. **Close** the **browser window** with the SCAP workbench report.

24. In the SCAP Workbench window, **click Clear** and then **click Scan** to run the scan again.

    When prompted, **answer** `yes` and **enter the password**.

25. When the results are done, **close** the **diagnostics window** and **click Show Report**.

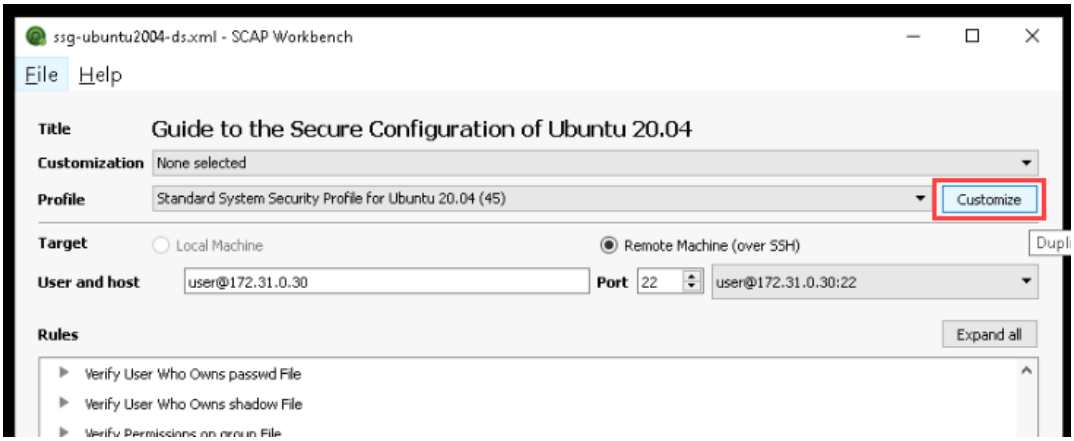
26. **Make a screen capture** showing that **the Disable SSH Access via Empty Passwords test passed**.

Note: Now you should see that the same test passes. As the system designer, you need to choose how to proceed. Do you want to enforce an explicit configuration using the test or do you just want to eliminate the test? It is possible to modify the profile to customize it for your needs.

27. **Close** the **browser window**.

28. In the SCAP Workbench window, **click Clear** to remove the results.

29. In the SCAP Workbench window next to the profile drop-down, **click Customize**.



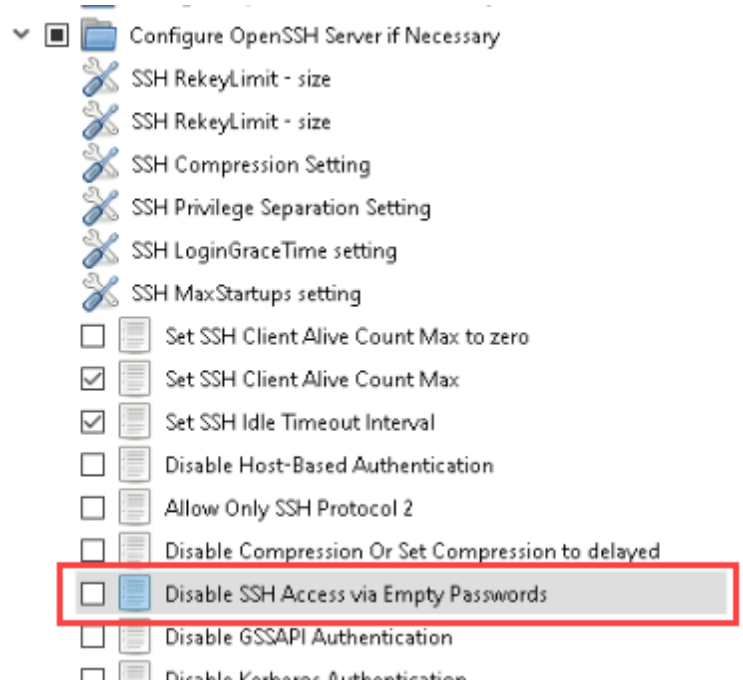
SCAP Workbench Customize button

Note: This will open a Customize Profile dialog asking for an ID. In production, when you have multiple machine types with different requirements, you will need to have a method of managing these profiles. Here, you can just accept the generated ID.

30. In the Customize Profile dialog, **click OK** to accept the generated ID.

Note: Now you will see a new window labeled Customizing "Standard System Security Profile for 20.04 [CUSTOMIZED]". The left pane has a list of all the tests in the ssg-ubuntu2004-ds.xml file that was loaded. Each test can be enabled or disabled using the checkbox next to it.

31. In the Customizing window, **scroll down** and **uncheck the box next to Disable SSH Access via Empty Passwords** to disable the test.



Customize Profile window Disable SSH Access via Empty Passwords rule checkbox

32. At the bottom of the Customizing window, **click OK** to save the settings.

33. In the SCAP Workbench window, **click Scan** to start a new scan.

    When prompted, **answer yes** and **enter the password**.

34. When the diagnostics window appears, **click Close** and then **click Show Report** to view the report.

Note: Now you can see that the test does not show up in the report at all -- where it was previously displayed under *Services > SSH Server > Configure OpenSSH Server if Necessary*, you should now see only three tests: *Set SSH Alive Count Max*, *Set SSH Idle Timeout Interval*, and *Disable SSH Root Login*.

35. **Make a screen capture** showing the **three tests run under *Services > SSH Server > Configure OpenSSH Server if Necessary***.

Note: You can use File > Save Customization Only to save a file with your changes to the profile. For now you can abandon the change.

36. **Close** the **SCAP Workbench window** and **click Yes** to discard changes when prompted.

Note: As the system designer, you need to make some choices here. Part of your task involves choosing the tests that are required for machines that are in production. The SCAP Security Guide provides a way of doing this that uses machine-readable files that are customizable. While the default scans may provide a suitable baseline, it is important that you review each test that is being used to verify that it is required and sufficient for its intended purpose. One of your goals as the designer is to minimize the number of false positives that show up in event logs while also ensuring that all important events are captured.

Part 3: Continuous Monitoring

Note: So now you have a design architecture and you have specified some security policies for the various machines in the design. You even have policies that can be machine checked. However, just having them does no good. The policies need to be enforced continuously. This is where monitoring comes in. It is not feasible to require that manual scans of machines be done by operators. Instead, you will use tools designed to automate the scans and generate reports based on the results.

The open-source project Wazuh provides some tools that can be used for automatically performing tasks for scanning endpoints and collecting the results on a central server. The central server has an indexing system and a dashboard web application that can be used for viewing results from scans. The basic architecture of the system is as follows:

- A server node runs a manager program to coordinate the agents.
- Each endpoint to be monitored runs an agent program (the manager node also runs an agent).
- The manager node can also run scans of agentless endpoints.
- The agents report results to the manager node.
- An indexer process on the manager node organizes and stores the results.

- A dashboard server provides a web interface to view the results.

The system also has methods for creating alerts and sending them via email and other methods as needed. Your team has already installed the Wazuh manager, indexer, and dashboard on SecServer. A Wazuh agent has already been installed on the DockerServer and DockerRunner.

1. On the vWorkstation desktop, **double-click** the **Firefox icon** to open the browser.

Firefox icon

2. In the browser toolbar, **click** the **Wazuh icon** to navigate to **`https://172.30.0.10.`**

3. On the login page, **enter** the **credentials below** and **click Log In** to access the server.

   Username: **`admin`**
   Password: **`admin`**

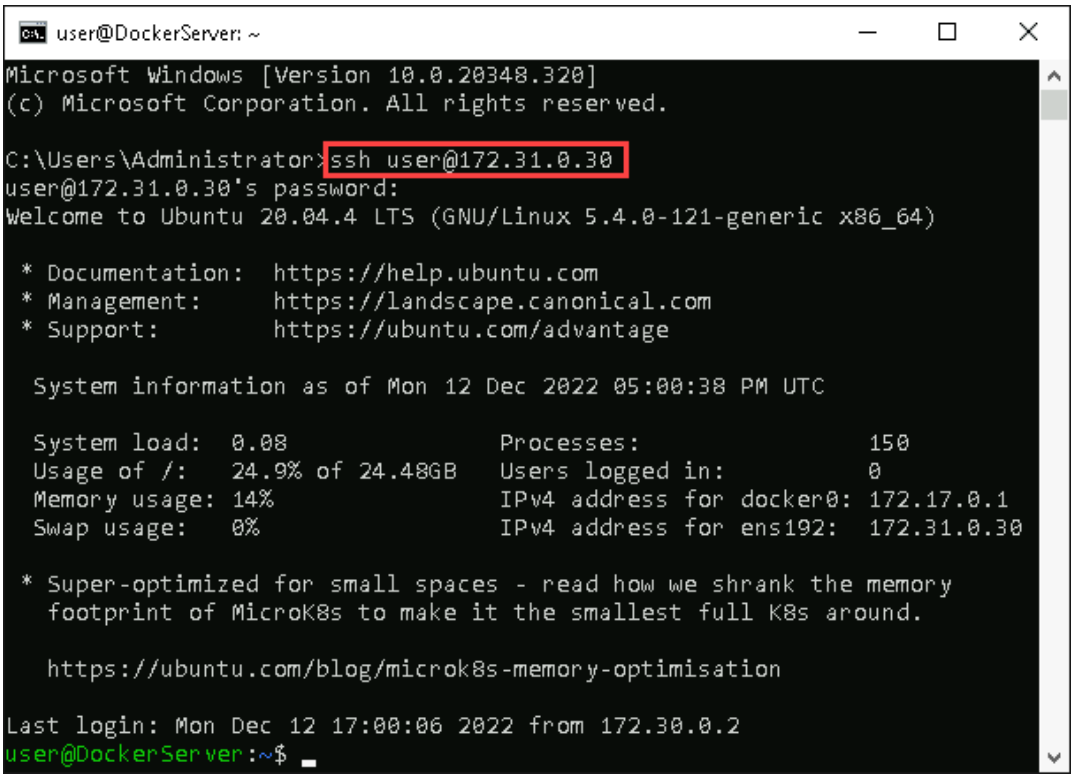
Wazuh login page

Note: The first page you see is the dashboard. At the top is a summary of the agents that are connected to the server. In this case, you should see two total agents and two disconnected agents. The agents have been installed but are not active.

4. On the vWorkstation desktop, **double-click** the **Command Prompt icon** to open a new prompt.

5. At the command prompt, **type** `ssh user@172.31.0.30` and **press Enter** to connect to DockerServer.

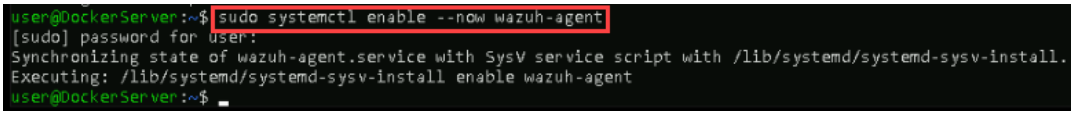
   When prompted, **type** `password` and **press Enter**.



Type ssh user@172.31.0.30

6. At the shell prompt, **type** `sudo systemctl enable --now wazuh-agent` and **press Enter** to enable the Wazuh agent.

   If prompted for a sudo password, **type** `password` and **press Enter**.



Type sudo systemctl enable --now wazuh-agent

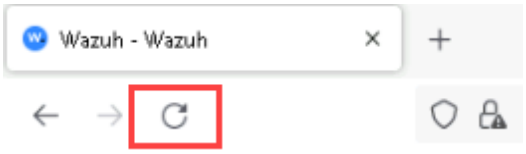
7. At the shell prompt, **type** `exit` and **press Enter** to disconnect from DockerServer.

8. **Repeat steps 5 through 7** to enable the Wazuh agent on DockerRunner at 172.31.0.20 with the same credentials.

9. **Close** the **command prompt window**.

Note: Now the Wazuh agents are running on the two machines. These agents will connect to the Wazuh server and perform actions that are configured. They should also appear as active on the dashboard.

10. In the Firefox window, **click** the **reload button** to reload the dashboard page.



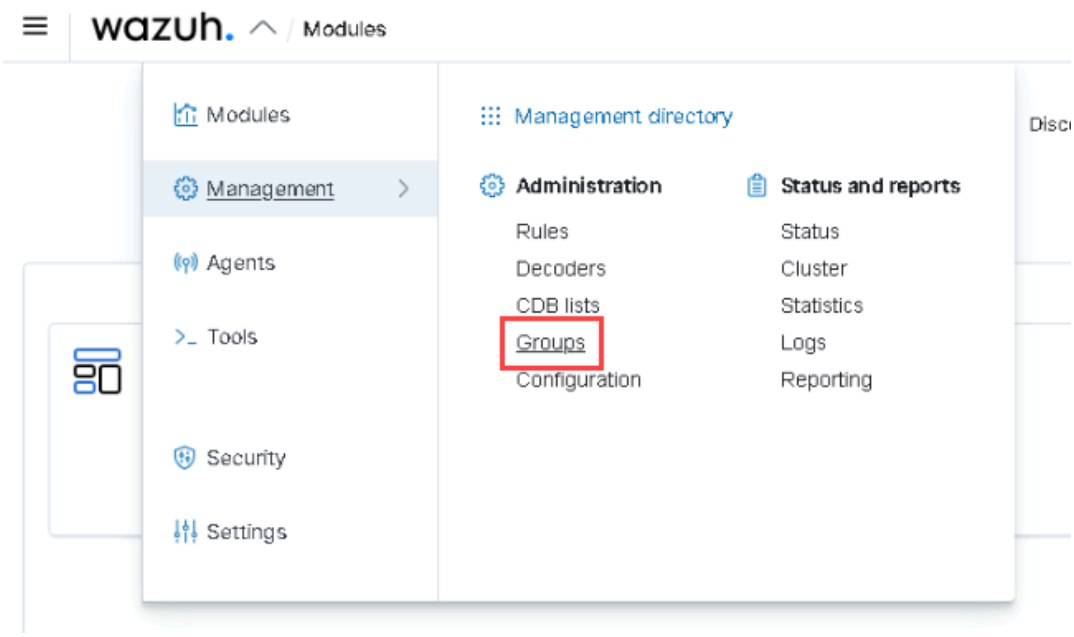
Reload button

Note: The top of the page should now show two active agents and 0 disconnected agents. If it does not, wait a few minutes and try again.

Wazuh allows agents to be placed into groups. When your team set up the Wazuh agents, they used the default group, so both agents are current in the group named "default."

We want to enable the SCAP scanning for the agents. They are both in the default group. By enabling SCAP scanning for the group, we can enable it for both agents in one step. In a real system, we would probably be making groups to better organize these configurations. While we are there, we will also enable the Docker integration.

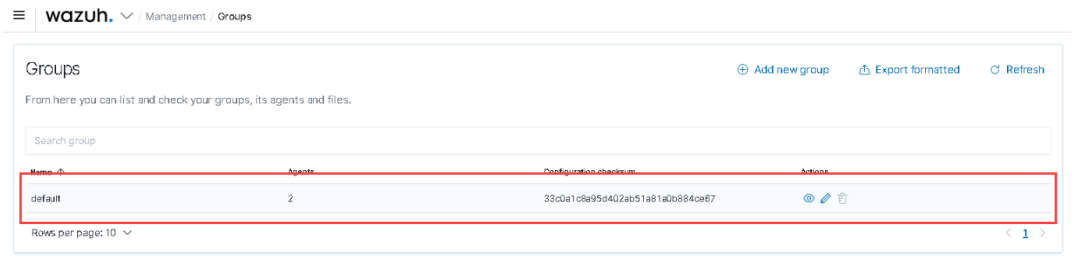
11. On the dashboard page, **navigate** to **Wazuh > Management > Groups** to view the groups page.



Wazuh Management menu

Note: You should see a list of groups with only one entry, default. It should also show that there are two agents in this group. Wazuh allows you to set configurations per group so that you can manage the settings for multiple machines with changes in a single place.

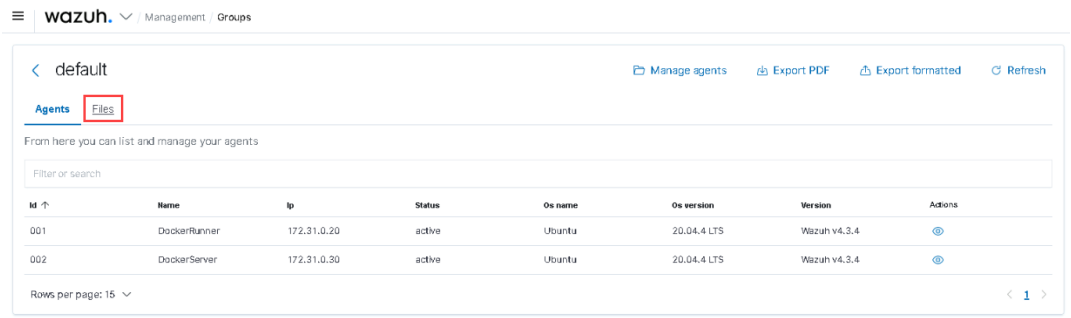
12. On the groups page, **click** the **default row** to view the details for the group.



Groups page default row

Note: Now you should see the agents tab, which shows a list of the two machines in the group. There is also a Files tab. The Wazuh agent can transfer files from the server to each machine in the group as part of the configuration.

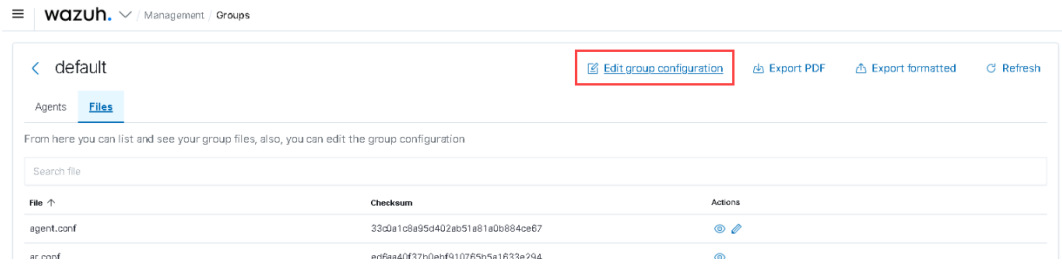
13. On the default group page, **click** the **Files tab** to view the files for the group.



default group page Files tab

Note: There are several pages of files listed. On the first page, at the top, is the file agent.conf. This file is what controls the Wazuh agent on the machines in the group. When the file is changed on the server, the new version is sent to each machine in the group and the agent on the machine reloads its configuration. In the group machines, the files are stored under /var/ossec/etc/shared/default. Other groups would use a similar path, ending with the group name instead of default.

14. On the default group page, **click** the **Edit group configuration link** to edit the agent.conf file.

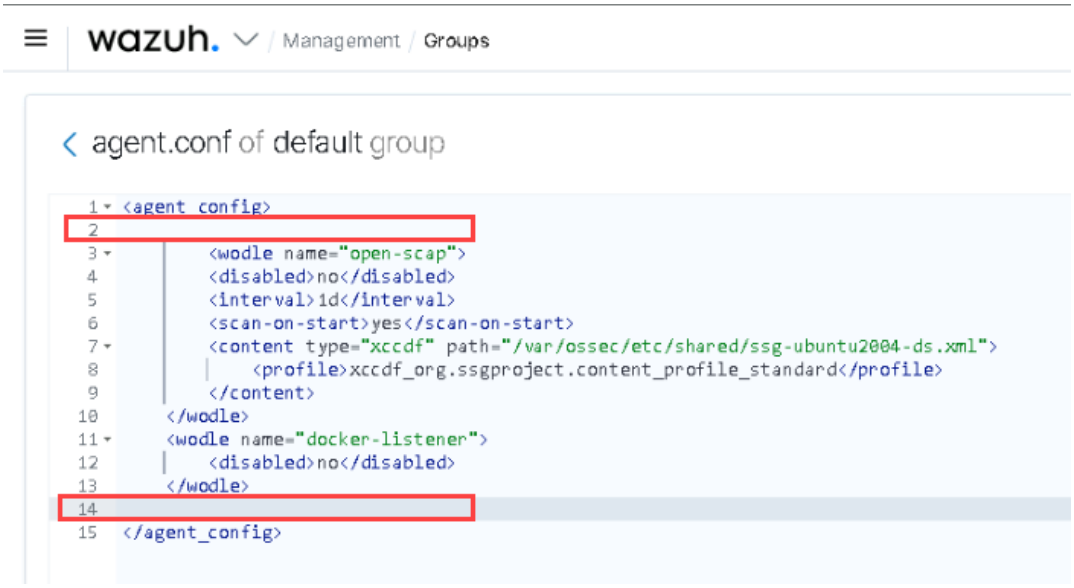


default group page Edit group configuration link

Note: The agents.conf file is an XML file. You can see information about the possibilities for this file in the documentation here. Currently, there are two elements that are commented out. The first wodle (Wazuh Module) element configures an OpenSCAP scan. The content element identifies the file with the test specifications, which is the same file that was on the vWorkstation desktop. This file has already been added to the group configuration by your team. The profile element selects the test set by ID. Here it is configured to use the same test that was used in the manual scan. The file and profile mentioned in the block are the same as the ones you used for the manual scan.

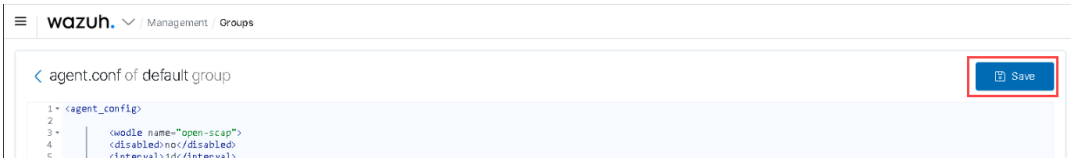
Once the agents are updated, they will do the scans, and more things should start to show up in the Wazuh database. You can navigate in Wazuh and see reports of the same errors you saw with the manual scan.

15. On the agent.conf page, **remove line 2 and line 14** so that the two wodle elements are no longer commented.



Remove line 2 and line 14

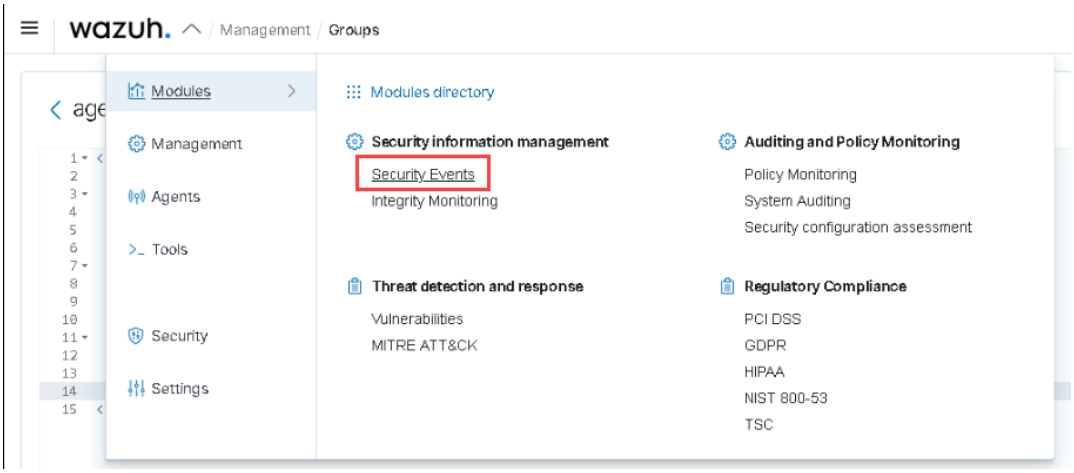
16. On the agent.conf page, **click** the **Save button** to save your changes.



Edit agent.conf page Save button

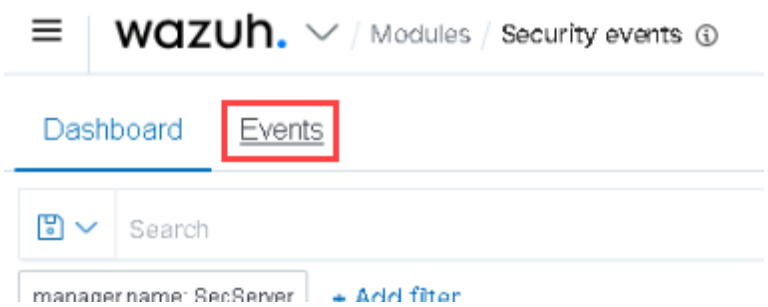
Note: You have also uncommented a wodle element that enables a Docker listener. Once the file is saved, it will automatically be updated on the agent machines and then the new tests will be started. This can take a couple of minutes, however.

17. In the Wazuh menu, **navigate** to **Wazuh > Modules > Security Events** to open the security events page.



Wazuh Modules menu

18. On the security events page, **click** the **Events tab** to view the list of events.



Security events page Events tab

Note: Once the SCAP scan has completed, you should see some events here. You can scroll through the list to see the rule descriptions. If there are not many events yet, wait a little longer and then use the Refresh button to reload the list.

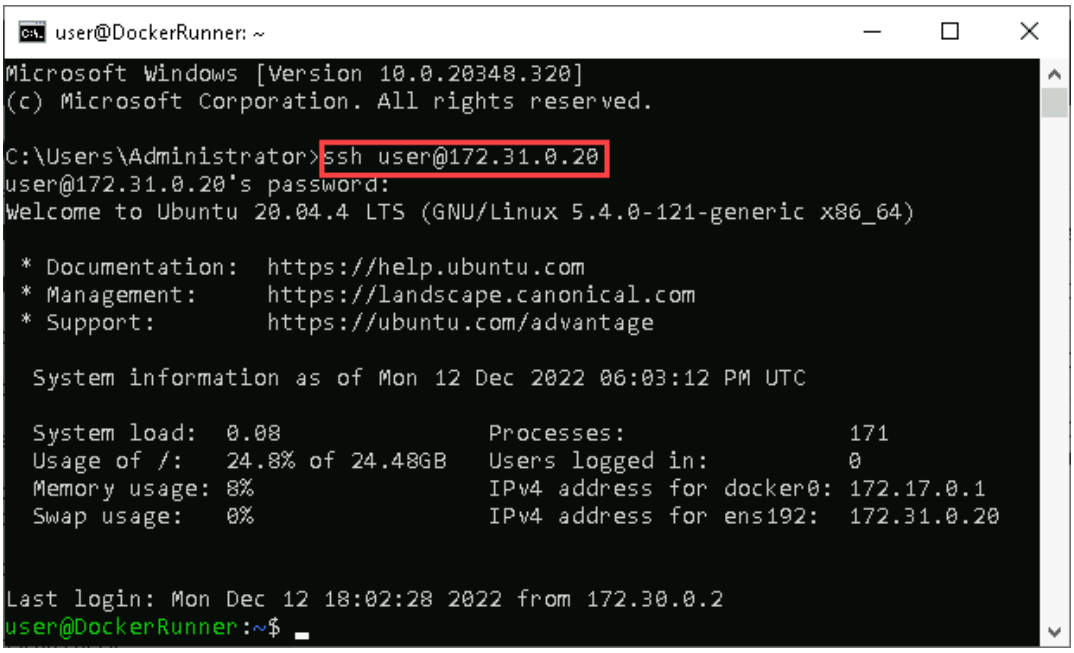
19. **Make a screen capture** showing an **event in Wazuh for Disable SSH Access via Empty Passwords failing** on the **DockerRunner machine**.

Note: The second wodle that was enabled will listen for Docker-related events on the two machines. By default, this includes events like getting an image from the registry and running a container.

In the next steps, you will pull a Docker image from the local registry, run a container based upon it, and execute a command within that container, in order to generate events for the Docker-Listener agent to pick up on.

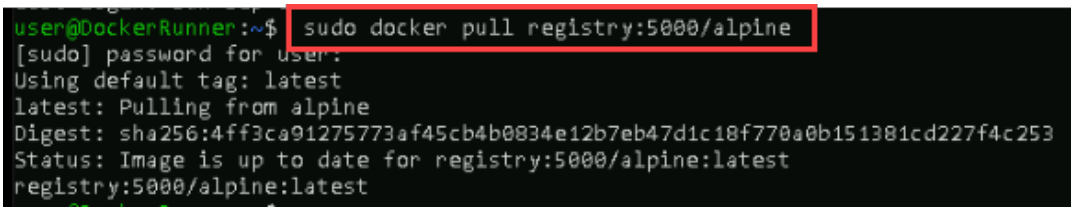
20. On the vWorkstation Desktop, **double-click** the **Command Prompt icon** to open a new prompt.

21. In the command prompt, **type** `ssh user@172.31.0.20` and **press Enter** to log in to the DockerRunner.



Type ssh user@172.31.0.20

22. At the shell prompt, **type** `sudo docker pull registry:5000/alpine` and **press Enter** to get an image from the registry.

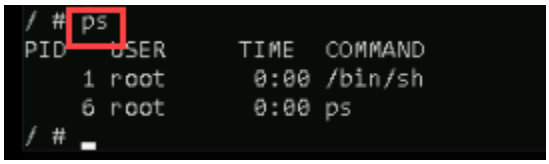


Type sudo docker pull registry:5000/alpine

23. At the shell prompt, **type** `sudo docker run -it registry:5000/alpine` and **press Enter** to get a shell in a running Docker container.

Type sudo docker run -t registry:5000/alpine

24. At the Docker container prompt, **type** `ps` and **press Enter** to view the processes running in the container.



Type ps

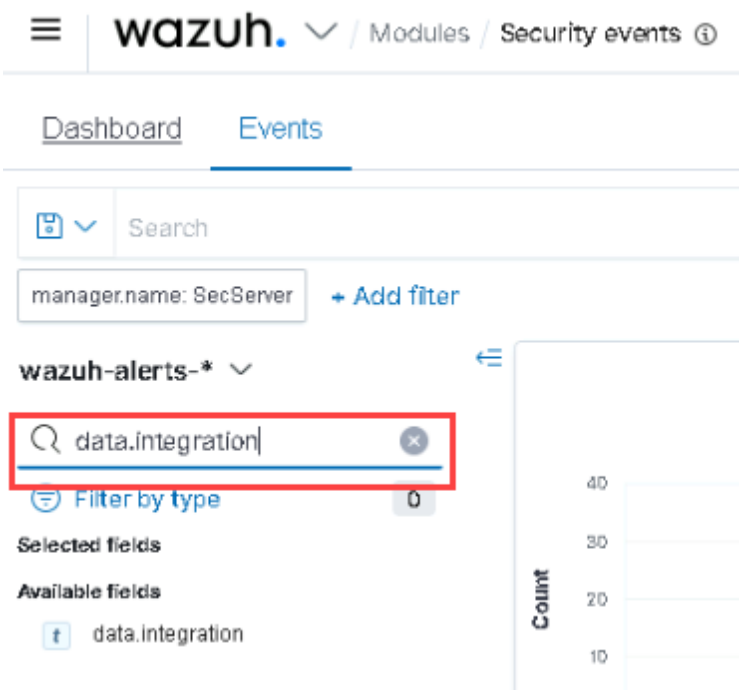
25. At the Docker container prompt, **type** `exit` and **press Enter** to exit the container shell.

26. At the shell prompt, **type** `exit` and **press Enter** to exit the DockerRunner shell.

27. **Close** the **command prompt window**.

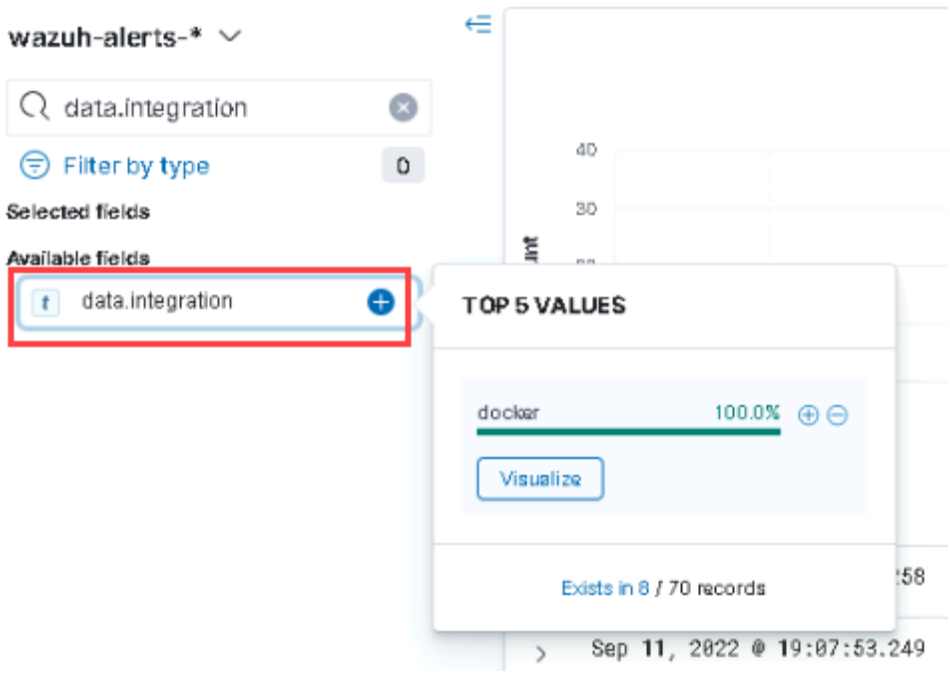
Note: These steps have included a few actions that should be logged by the Docker-Listener agent for Wazuh. You should be able to see new events related to these actions.

28. On the events table page, **type** `data.integration` into the **Search field names area**.



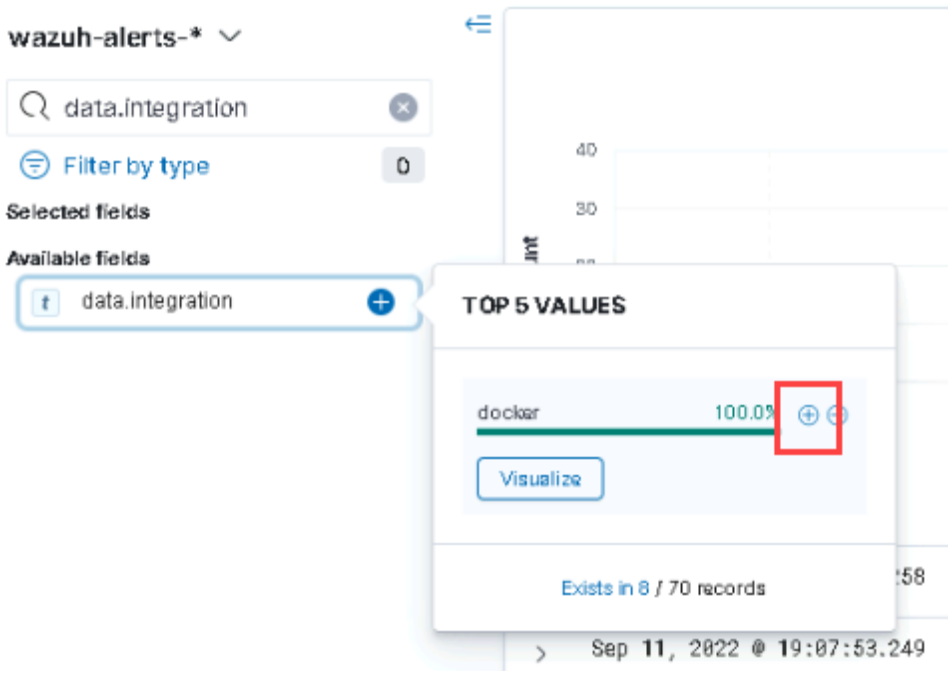
Type data.integration in the Search field

29. Below the search area, **click** on **data.integration** to reveal a popup.



data.integration popup window

30. In the popup, **click** the **plus symbol** next to docker to filter the events.



Click the plus symbol next to docker

Note: When the page reloads, you should see only events that include the field data.integration with the value "docker," which comes from the Docker-Listener wodle.

31. **Make a screen capture** showing **an event about the alpine image being pulled**.

32. **Make a screen capture** showing **an event about a container being started**.

Note: These steps have given only an indication of how things can be monitored with Wazuh. As a part of your design phase, you need to consider all the events that would be important in the production environment. There are many ways to customize how Wazuh reports things. You can use rule sets to create alerts automatically from events. You can also create a custom dashboard to view events and alerts.

Implementing the precise rules to be used in a production environment is beyond the scope of this lab. As the system designer, you would need to indicate what monitoring system is going to be used. Then you would need to indicate the different event-gathering methods that would be needed for each type of endpoint in the system. You would also need to determine what the reporting and alerting

requirements are for the system.

Challenge and Analysis

Note: The following scenario is provided to allow independent, unguided work, similar to what you will encounter in a real situation.

Part 1: Make a New SCAP Profile

Your team is conducting research to determine which SCAP tests should be included in scans of the Docker registry and runners. To deploy the results, you will need to be able to place custom definitions on the machines. To determine the procedure, you will create a simple profile that includes a single test.

You have already conducted a scan on DockerServer using a custom profile, so you should be well-equipped for this new task. However, this time you will be making significant changes to the default profile. Your plan is as follows:

1. **Create a new OpenSCAP custom profile** based on the *Standard System Security Profile for Ubuntu 20.04* (as you did in Part 2 of this lab), this time removing all tests except for "Disable SSH Access via Empty Passwords."

2. Once you have finished customizing the standard Ubuntu 20.04 profile, **save your customization** from the OpenSCAP File menu (File > Save Customization Only) to the vWorkstation desktop using the default name provided.

   The suggested File name should be *ssg-ubuntu2004-ds-tailoring.xml*.

3. **Run an OpenSCAP scan** against the DockerRunner machine (172.31.0.20) using your custom profile.

4. **View your result within OpenSCAP Workbench** (i.e., do not click Show Report to display the results in the browser).

Make a screen capture showing the **results in OpenSCAP Workbench after using your new profile to scan DockerRunner**.

Part 2: Update the Group Files to Use the New Profile

Now that the custom profile is created and working locally, you will need to be able to update the configuration for the agents. There are two steps required to update the agents:

1. Add your new profile to the /var/ossec/etc/shared/default/ folder on SecServer. You will do this using scp, or secure copy, which uses SSH to transfer files between remote workstations. On the vWorkstation, **open the Command Prompt and execute** the following command:
   `scp C:\Users\Administrator\Desktop\ssg-ubuntu2004-ds-tailoring.xml root@SecServer:/var/ossec/etc/shared/default/`

   When prompted for a password, **type `password`** and **press Enter**. Note that allowing root login over SSH using a password is a security concern in itself -- something likely to trigger an OpenSCAP rule for most default profiles -- and as such, is only used in this lab for simplicity.

2. Update the agent group configuration file using the Wazuh dashboard.

   You have already edited this file once in Part 3, where you removed the comment markers to enable two Docker wodles. You will **navigate back to this agent group configuration file** and **change the *path* value** to point to the custom profile you uploaded above.

Make a screen capture showing the **new group configuration file for the default group in Wazuh**.