jamf | PRO

Managed Service Configurations with Blueprints in Jamf Pro

Rev. 20251210

Contents

Preface

Managing modern Apple fleets has evolved far beyond simply pushing profiles or enforcing a handful of payloads. As organizations grow, so does the need for a more adaptive, resilient, and secure management framework that can enforce policy at scale while remaining flexible enough to support real-world workflows. Declarative Device Management (DDM) represents Apple's answer to this evolution. When paired with managed service configuration files, Declarative Device Management (DDM), introduces a new level of precision and predictability in controlling sensitive system components across macOS devices.

One of the most valuable applications of this capability is the management and restriction of files like sudoers on macOS. For years, limiting access to sudo, whether by user, group, or role, has been a cornerstone of macOS hardening. The challenge, however, has always been enforcing these restrictions without completely removing a user's administrative capabilities. Many organizations still depend on allowing users to handle everyday admin-level tasks such as installing printers, capturing Wi-Fi settings, or running specialty applications. Declarative Device Management (DDM), provides a modern and tamper-resistant way to accomplish this. Users can have administrative rights where appropriate, while the system remains protected from unauthorized privilege escalation.

This guide will focus on two key workflows: customizing sudo access by modifying the sudoers file, and enabling Touch ID authentication for sudo through a Pluggable Authentication Module (PAM) file. Together, these configurations demonstrate how Declarative Device Management (DDM) can enforce privilege controls in a consistent, tamper-resistant way across your macOS fleet. Apple has expanded this managed-service architecture to cover several core system components. Beginning with macOS 14, the following built-in services automatically check for managed service configuration files, which, when present, override the standard local configuration on the device:

- sudo
- sshd
- PAM
- CUPS
- Apache
- zsh (/private/etc/zprofile)
- bash (/private/etc/profile)

This shift represents a meaningful step in Apple's long-term strategy: reducing reliance on traditional one-way MDM commands and empowering devices to remain in a consistent, compliant state even when intermittent or offline. If you want to better understand the foundational concepts behind this architecture, Apple outlines the entire model in their Platform Deployment guide:

https://support.apple.com/guide/deployment/service-configuration-files-declarative-depdac2c8d89/1/web/1.0

The following tools and prerequisites are essential for completing the sections outlined in this guide:

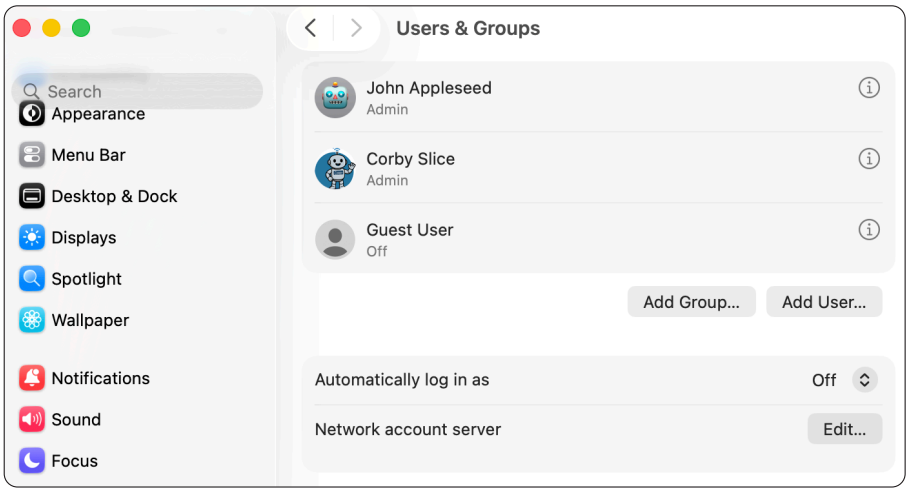
Requirements:

- A Jamf Pro server with version 11.23 or later.
- Two non-production Mac computers with macOS Sonoma 14.0 or later. Touch ID enabled with two macOS administrator accounts, and enrolled in your Jamf Pro server. One Mac computer will be used to follow along with the steps in this guide, the other will be used to test our work.
- A GitHub repository hosting publicly available files. Content must be accessible over HTTPS and no authentication or token required for access. A web server can be used instead of using GitHub to store the files publicly. This guide will NOT cover setting up a web server.
- A device management service that supports managed service configuration files. In Jamf Pro, these appear as Service Configuration Files within Blueprints
- Working knowledge of the Terminal and the zsh command-line environment.

This guide will use the following two macOS administrator accounts with Touch ID enabled for both accounts:

Full Name: **John Appleseed**
Account Name: **jappleseed**
Password: **Apple321!**

Full Name: **Corby Slice**
Account Name: **cslice**
Password: **Apple321!**



Optional: Consider making a Static Computer Group in Jamf Pro named "Proof of Concept Static Group" and add a non-production Mac for testing the workflow we create in this guide. This guide will use a pre-configured Proof of Concept Static Group and add our non-production test Mac computer to it so we can test our workflow in the last section of this guide.

Section 1: Configure sudo Access and Touch ID for sudo Authentication

What You'll Need:
Learn what hardware, software, and information you'll need to complete the tutorials in this section.

Hardware and Software:
Requirements for following along with this section:

- A non production Mac computer with macOS 14 or later with Touch ID Enabled.
- A working knowledge of the Terminal and the zsh command-line environment.

In this section, we will walk through two critical privilege-management workflows that strengthen macOS security while maintaining a streamlined user experience. This guide will use jappleseed and cslice as the macOS administrator accounts as discussed in the preface of this guide.

First, we will customize the sudoers configuration so that only a single designated local account is permitted to run the sudo command. We will remove sudo privileges from all other users, including those in the traditional admin group. By the end of this process, you will have a properly structured and deployable sudoers file, ready to be enforced through managed service configuration files to ensure consistent, hardened privilege controls across your macOS environment.

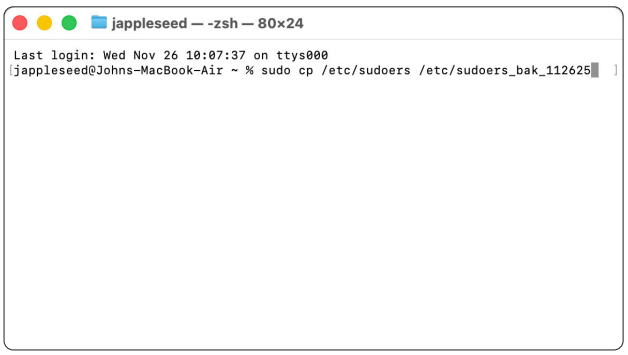
Next, we will create a Pluggable Authentication Module (PAM) configuration to enable Touch ID authentication for sudo in the Terminal. With macOS Sonoma, Apple introduced a native and repeatable method for using Touch ID to authorize sudo commands replacing older script-based approaches that were difficult to maintain and often required re-deployment after macOS updates were applied. With Declarative Device Management (DDM) and Jamf Pro Blueprints, this functionality can now be deployed reliably and automatically. When used together, these workflows provide a robust and secure model for controlling elevated privileges on macOS devices using Declarative Device Management (DDM) and Apple's modern managed-service framework.

Configure sudo access:

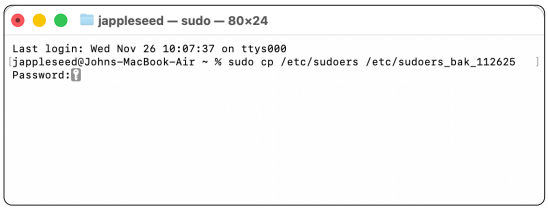
1. Open the /Applications/Utilities folder.
2. Open Terminal.

3. Make a backup of your existing sudoers file. Adjust the date at the end of the command to today's date. Enter the following command and press Return:

```
sudo cp /etc/sudoers /etc/sudoers_bak_<Today's Date>
```

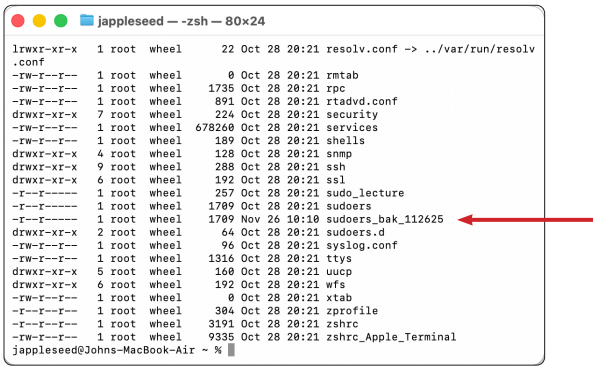


4. Enter the administrator's password.



5. Enter the following command and press Return:
   `ls -l /private/etc`

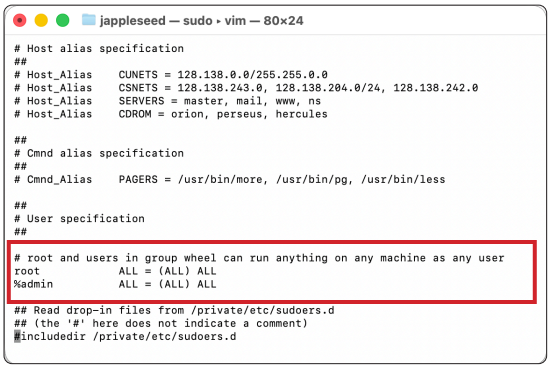
6. Confirm the file "sudoers_bak_<Today's Date>" has been created.



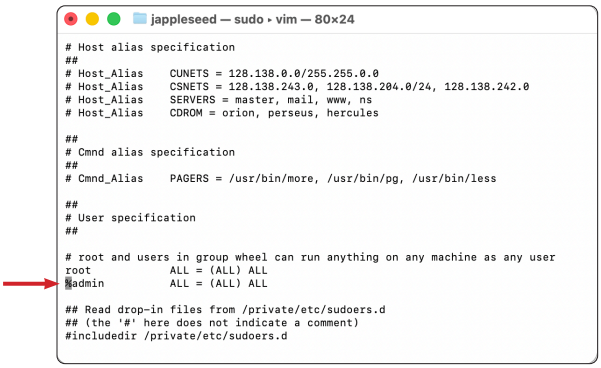
7. The sudoers file must be edited with the visudo command. Enter the following command and press Return:
   `sudo visudo /etc/sudoers`

8. Enter the administrator's password.

9. Scroll down to the section that says "#root and users in the group wheel can run anything on any machine as any user"



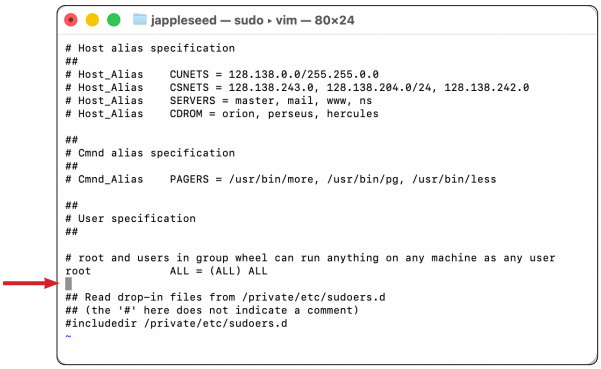
10. Move your mouse cursor to the line beginning with %admin.



11. Enter the following to delete the line:

dd

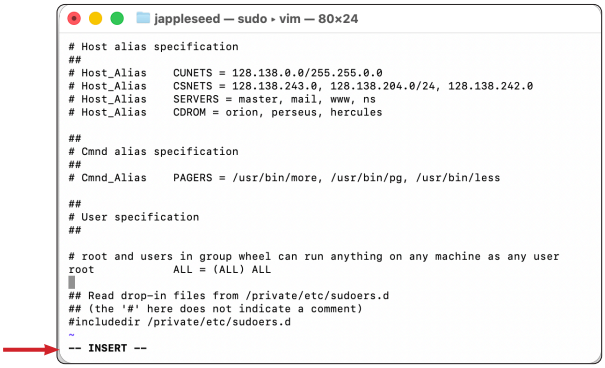
Confirm the line has been deleted



12. Enter a lowercase (**i**) to enter insert mode:

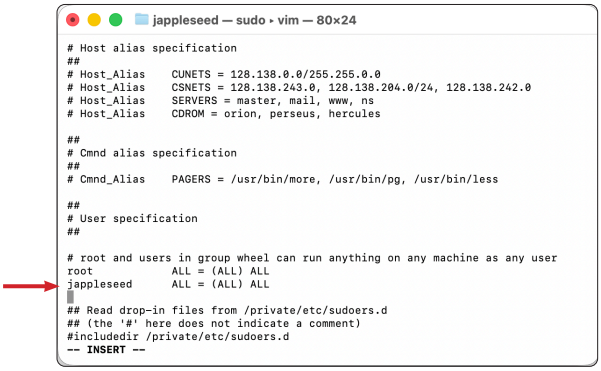
i

Confirm –- INSERT – at the bottom of the Terminal window.

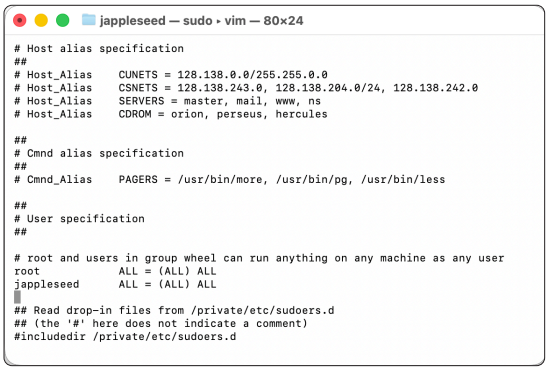


13. In this guide, we are only allowing the jappleseed user to use sudo. Enter the following and press Return:

jappleseed ALL = (ALL) ALL

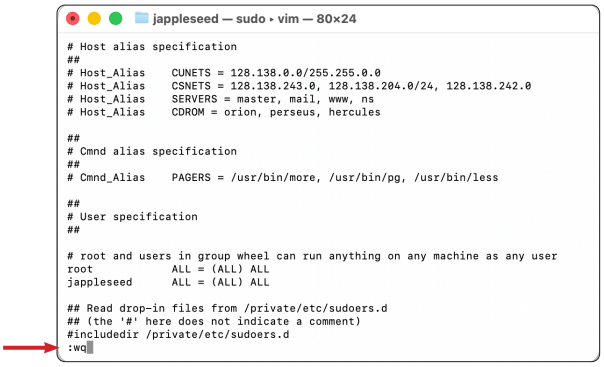


14. Press the escape key to exit insert mode.

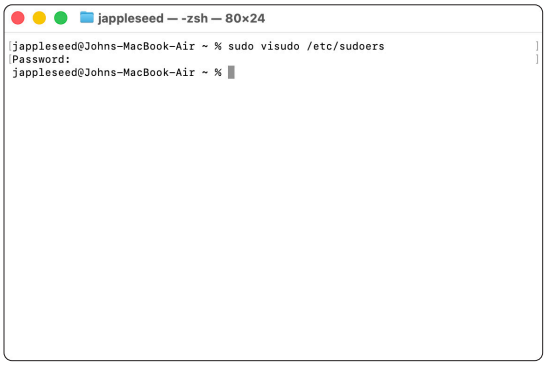


15. Enter the following:

```
:wq
```



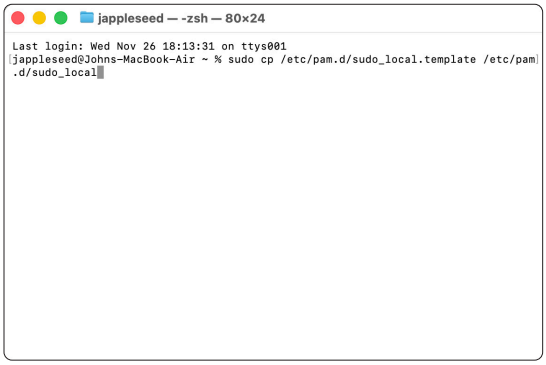
16. Press Return to write the change and exit visudo.



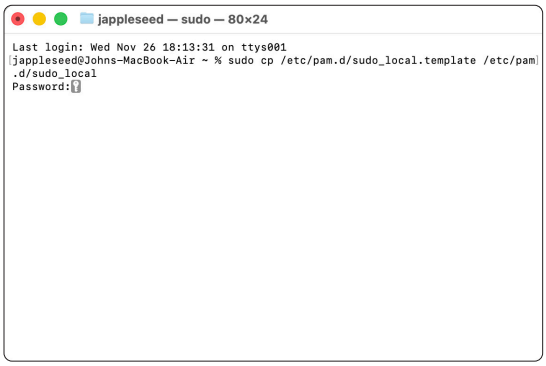
Enable TouchID for sudo authentication

17. Make a copy of the PAM file named sudo_local.template and name it sudo_local. Enter the following command and press Return:

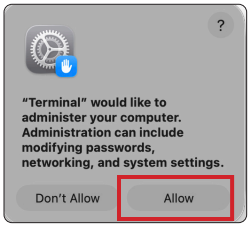
```
sudo cp /etc/pam.d/sudo_local.template /etc/pam.d/sudo_local
```



18. Enter your administrator password and press Return.



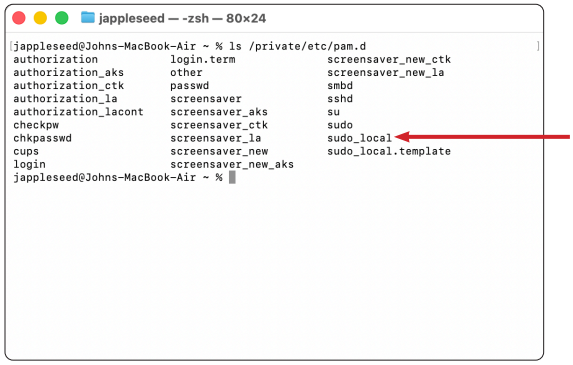
19. If a window appears asking you to allow Terminal to administer your Mac computer, click Allow.



20. Enter the following and press Return:

```
ls /private/etc/pam.d
```

Confirm the sudo_local file was created.

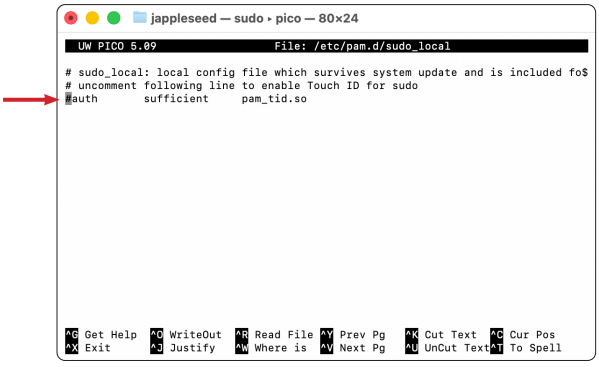


21. Run the following command to open the sudo_local file. Enter your administrator password if prompted.

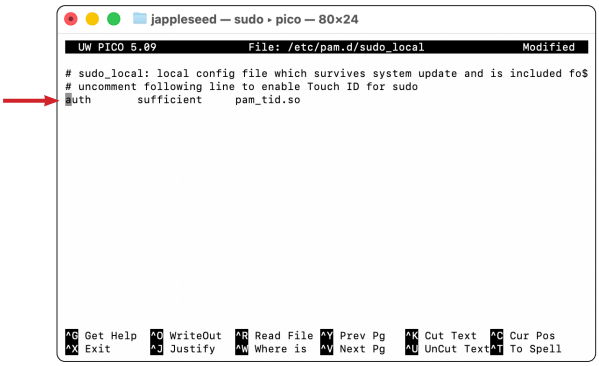
```
sudo nano /etc/pam.d/sudo_local
```

NOTE: nano is a command line text editor.

22. Using your arrow keys, navigate to the #auth line.

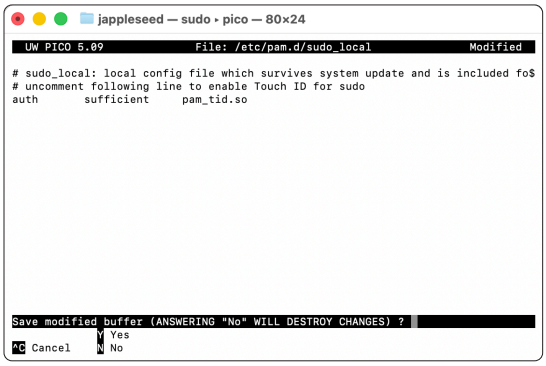


23. Delete the octothorp (**#**) at the beginning of #auth. This will uncomment the line so Touch ID can be enabled.

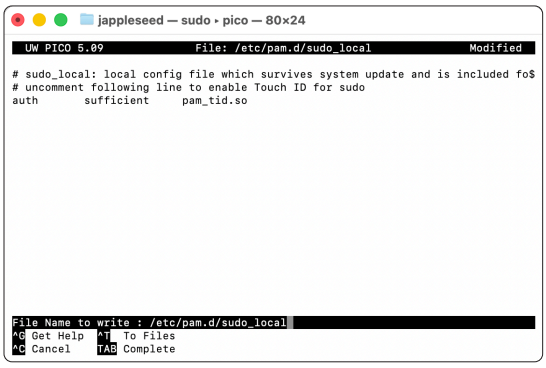


24. Press Control (^) and X keys.

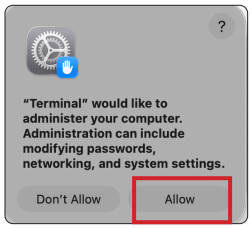
25. Enter **Y** to save the changes



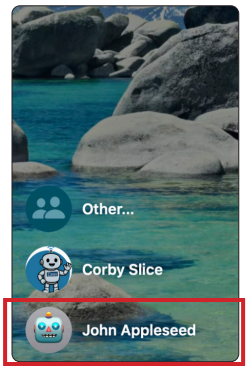
26. Press the Return key



27. If presented with the message below, click Allow.

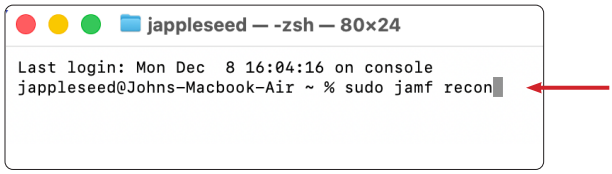


28. Let's test our work. If necessary, log in as John Appleseed.



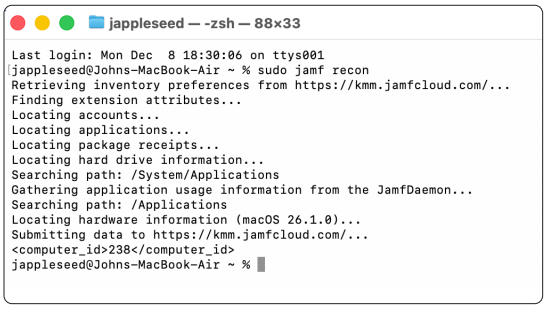
29. Run the following command:

```
sudo jamf recon
```

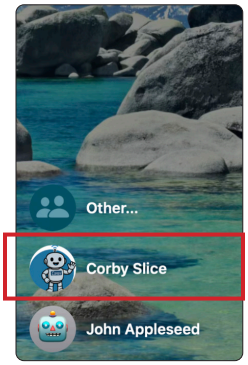


30. You will be prompted to use Touch ID or enter your password. Use Touch ID.

31. The command executes because jappleseed is listed in the sudoers file.



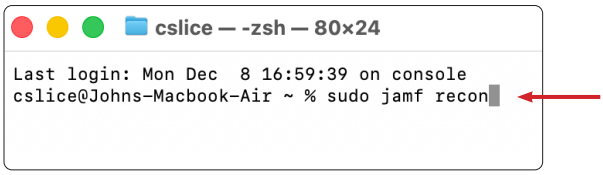
32. Logout out of the John Appleseed account and login as Corby Slice.



33. Open Terminal located in /Applications/Utilities.

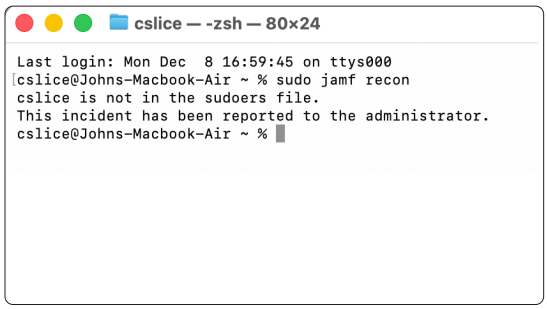
34. Run the following command:

```
sudo jamf recon
```



35. You will be prompted to use Touch ID or enter your password. Use Touch ID.

36. The command will fail because the user cslice is not in the sudoers file. This is the expected result.



This completes this section. In the next section, we will create the sudoers configuration and the sudo_local configuration zip files so they are ready for deployment.

Section 2- Package Configuration Files and Prepare Them for Deployment

What You'll Need:
Learn what hardware, software, and information you'll need to complete the tutorials in this section.

Hardware and Software:
Requirements for following along with this section:

- A non production Mac computer with macOS 14 or later and administrative privileges
- A user with sudo access. This guide will use jappleseed
- A working knowledge entering commands in the Terminal using zsh

In this section, we will walk through the process of creating properly structured ZIP archives for both your customized sudoers file and your sudo_local Pluggable Authentication Module (PAM) configuration file. Each ZIP archive must follow Apple's required directory structure and must be accompanied by a corresponding SHA-256 hash. These two components, the ZIP file and its hash, are mandatory for deploying service configuration files through Declarative Device Management (DDM) in Jamf Pro.

Once each ZIP archive and its hash have been created, you will upload them to a publicly accessible GitHub repository, ensuring that macOS devices can securely retrieve and apply these configurations during deployment. By the end of this section, you will have fully prepared, hosted, and deployment-ready packages for both privilege control and Touch ID based sudo authentication.

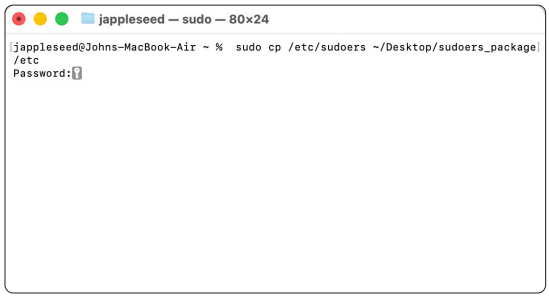
1. If necessary, Open Terminal and make a directory to contain the modified sudoers file. Enter the following command and press Return:
   `mkdir -p ~/Desktop/sudoers_package/etc`

2. Confirm the sudoers_package folder was created on your Desktop with the etc folder inside of it.

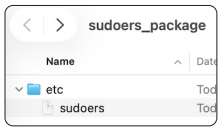
3. Copy the modified sudoers file to the etc folder inside the sudoers_package folder on your Desktop. Enter the following and press Return:

```
sudo cp /etc/sudoers ~/Desktop/sudoers_package/etc
```

4. Enter the administrator's password or Touch ID.

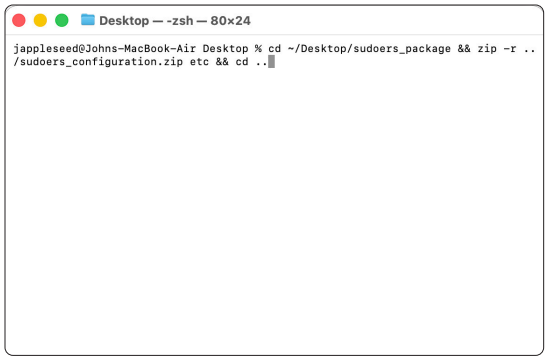


5. Open the sudoers_package folder on your Desktop. Expand the etc folder and confirm that you see the sudoers file.

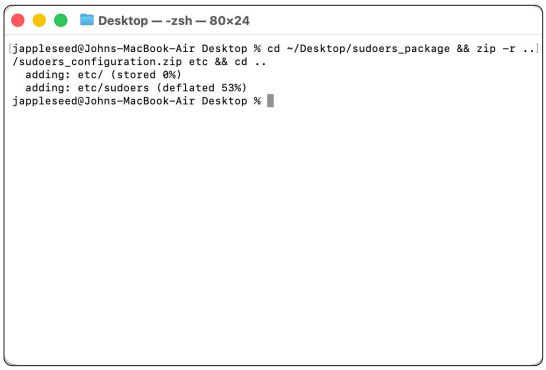


6. Let's create the zip file that contains the sudoers file inside the etc directory, enter the following command and press Return:

```
cd ~/Desktop/sudoers_package && zip -r ../sudoers_configuration.zip etc . && cd ..
```



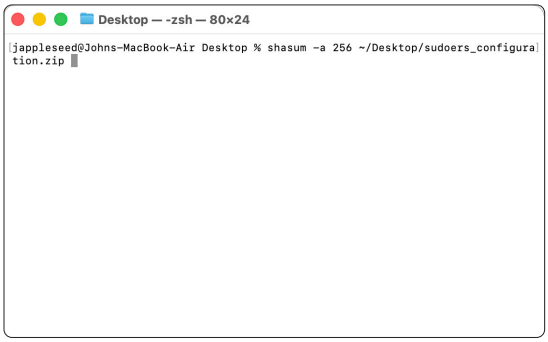
7. You will see the results of the command in the Terminal window.



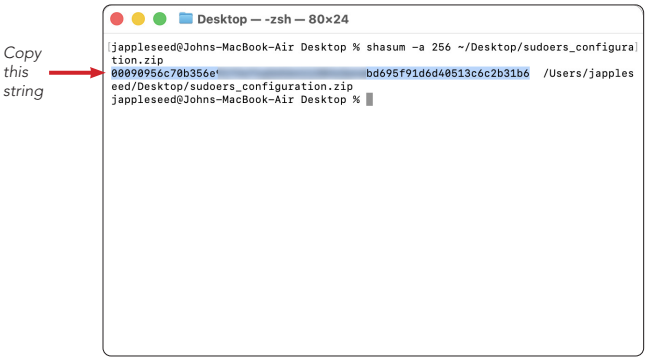
8. Confirm the sudoers_configuration.zip file is on your Desktop.

9. Create the SHA-256 hash of the sudoers_configuration.zip file which is required for the Blueprint in Jamf Pro. Enter the following and press Return:

```
shasum -a 256 ~/Desktop/sudoers_configuration.zip
```

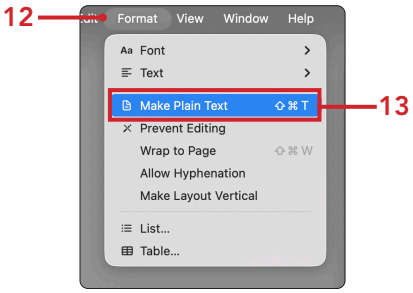


10. The SHA-256 hash of the zip file will be shown in the Terminal. It appears as a long string of numbers and letters. Copy the SHA-256 hash from the Terminal.

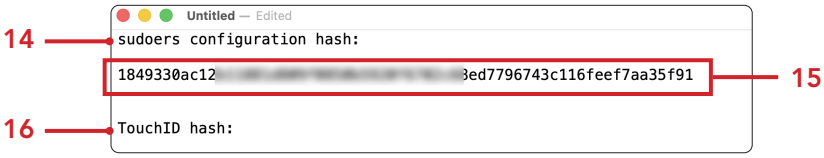


11. Open TextEdit located in the Applications folder.

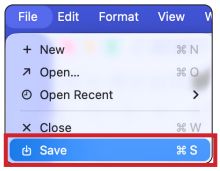
12. From the Format menu.
13. Select “Make Plain Text”.



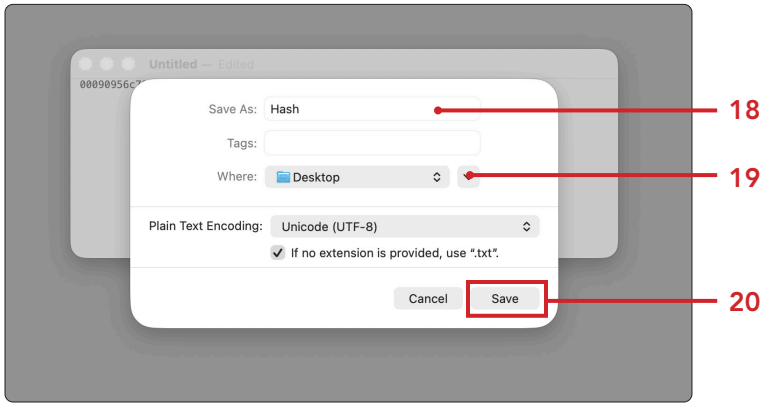
14. Enter the following: **sudoers configuration hash**:
15. Paste in the SHA-256 hash that you copied in an earlier step.
16. Enter **TouchID hash**: (This will be used in a later step)



17. In TextEdit, select the File menu, choose Save.



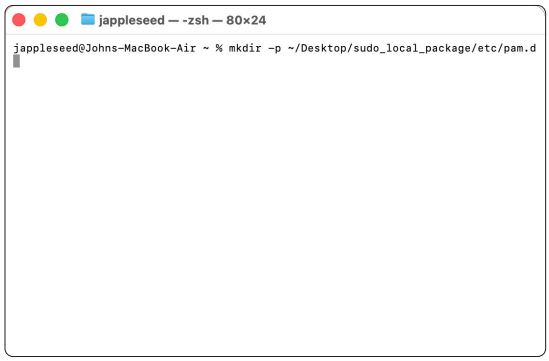
18. Enter **Hash** for the name.
19. Navigate to your Desktop.
20. Click Save.



21. Confirm the file named Hash was created on your Desktop and minimize TextEdit.app.

22. Switch back to the Terminal and make a directory to contain the modified sudo_local file. Enter the following command and press Return:

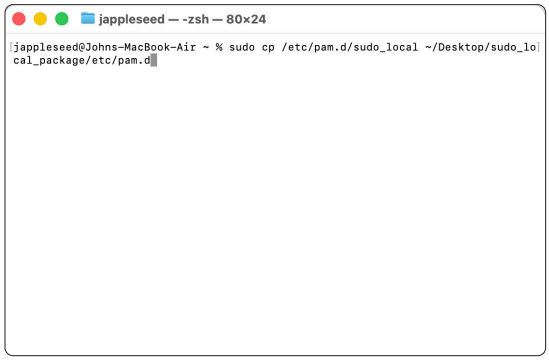
```
mkdir -p ~/Desktop/sudo_local_package/etc/pam.d
```



23. Confirm the sudo_local_package folder was created on the your Desktop with the etc and pam.d folders inside of it.

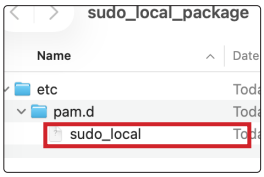
24. Copy the modified sudo_local file to the /etc/pam.d folder inside the sudo_local_package folder on your Desktop. Enter the following command and press Return:

```
sudo cp /etc/pam.d/sudo_local ~/Desktop/sudo_local_package/etc/pam.d
```



25. Enter the administrator's password or Touch ID.

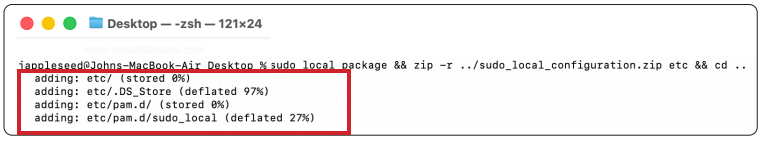
26. Open the sudo_local_package folder on your Desktop. Expand the etc folder then the pam.d folder and confirm that you see the sudo_local file.



27. Let's create the zip file that contains the sudo_local file inside the /etc/pam.d directory, enter the following command and press Return:

```
cd ~/Desktop/sudo_local_package && zip -r ../sudo_local_configuration.zip etc && cd ..
```

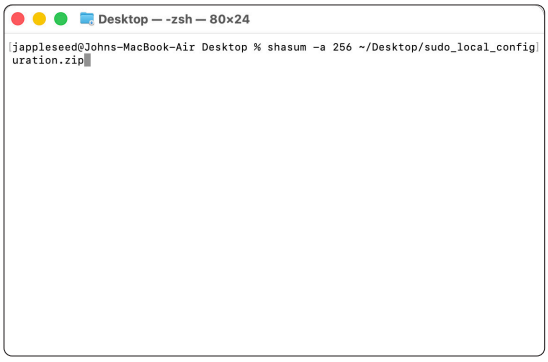
28. You will see the results of the command in the Terminal window.



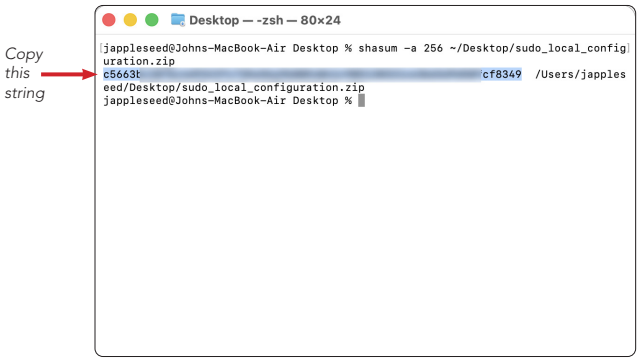
29. Confirm the sudoers_configuration.zip file is on your Desktop.

30. Create the SHA-256 hash of the sudoers_configuration.zip file which is required for the Blueprint in Jamf Pro. Enter the following and press Return:

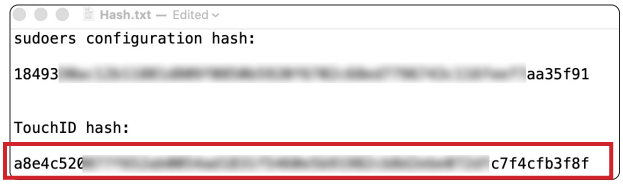
```
shasum -a 256 ~/Desktop/sudo_local_configuration.zip
```



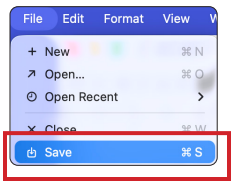
31. The SHA-256 hash of the zip file will be shown in the Terminal. It appears as a long string of numbers and letters. Copy the SHA-256 hash from the Terminal.



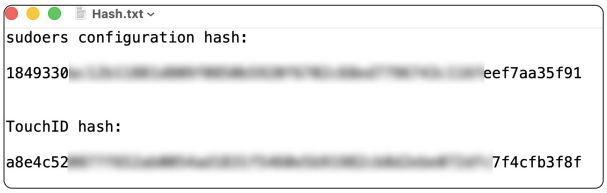
32. Go back to your TextEdit document, Hash.txt. and paste in the SHA-256 hash that you copied in an step 31.



33. Save the file.



34. Minimize or hide the Hash.txt document. We will need in the next section.



This completes this section. In the next section, we will upload the two zip files we created to our GitHub repository.

Section 3: Upload Configuration Files to GitHub

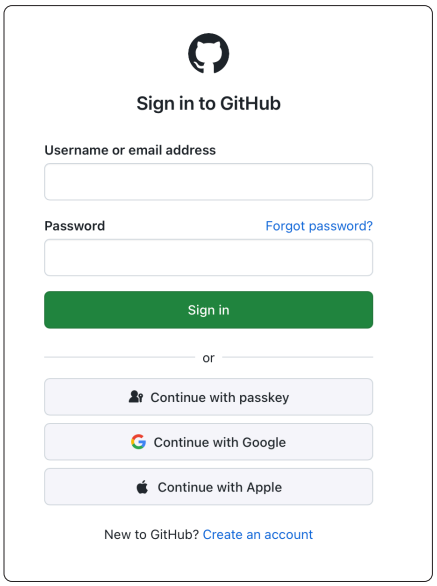
What You'll Need:
Learn what hardware, software, and information you'll need to complete the tutorials in this section.

Hardware and Software:
Requirements for following along with this section:

- A non production Mac computer with macOS 14 or later and administrative privileges
- Access to your Github repository with administrative privileges

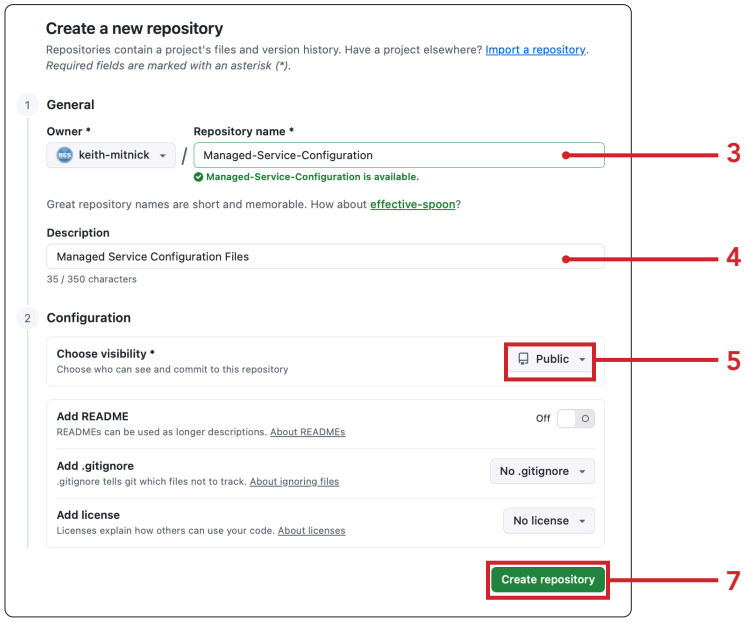
NOTE: We are using a Github repository in this section however, the files uploaded in this section can also be hosted on a web server. This guide will NOT cover hosting the files on a web server.

1. Using a web browser of your choosing, go to: https://github.com/login and sign in with your preferred method.

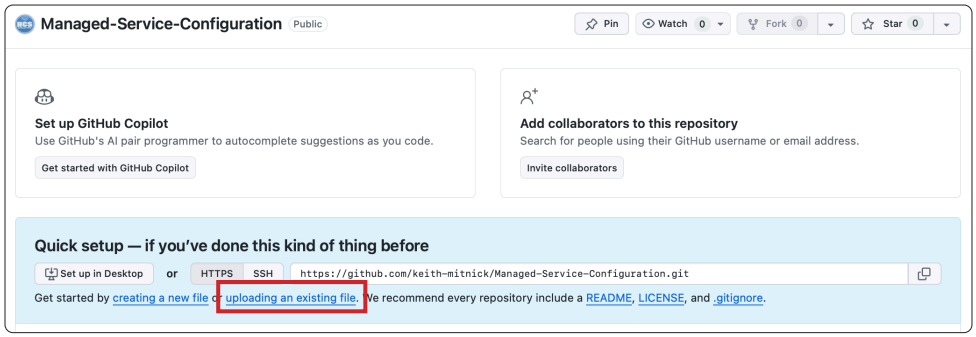


2. Click the Repositories Tab and from the sidebar, click New.

3. Enter **Managed-Service-Configuration** for Repository name.
4. Enter **Managed Service Configuration Files** for Description.
5. Select from Public from the Choose visibility menu.
6. Leave everything else at their default settings
7. Click Create repository

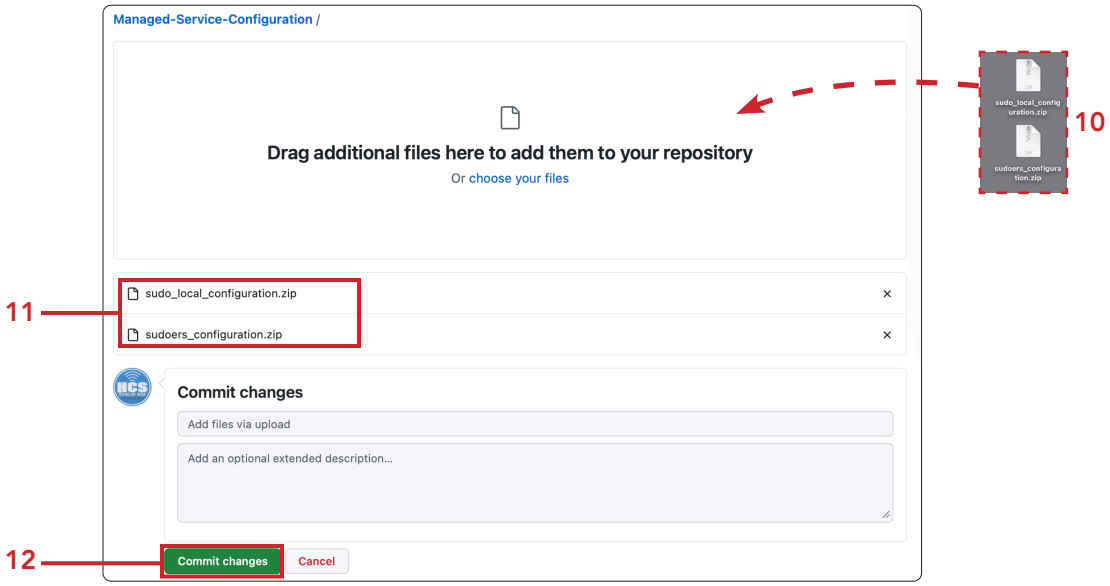


8. Click on the "uploading an existing file" link.

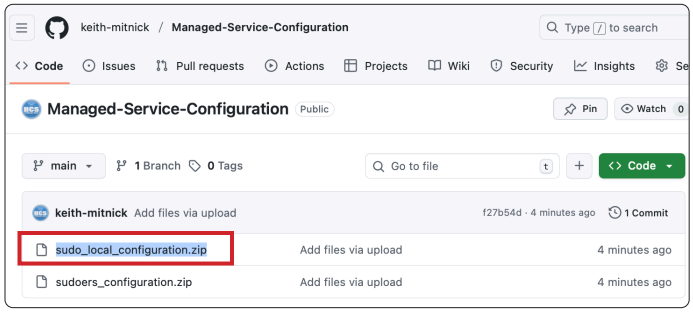


9. Make sure you have the sudo_local_configuration.zip and sudoers_configuration.zip files on your Desktop.
   NOTE: We created these files and saved then to the Desktop in an earlier section of this guide.

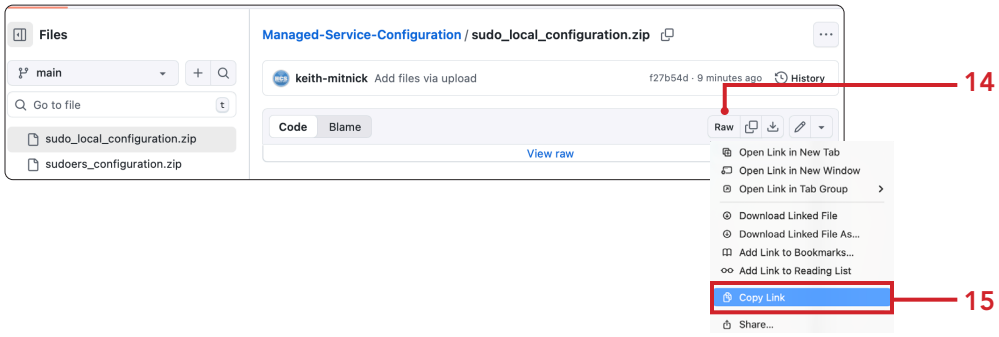
10. Drag and drop the sudo_local_configuration.zip and sudoers_configuration.zip files from your Desktop to the field.
11. Confirm the files show up in the window once the upload is done.
12. Click Commit changes



13. Click on the sudo_local_configuration.zip file.

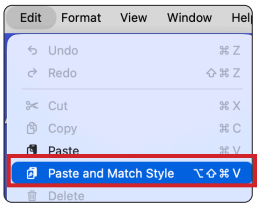


14. Hold down the Control key and click on Raw to show its menu.
15. Select Copy Link.



16. If necessary, open the Hash file located on your Desktop and create a new line under the Touch ID configuration hash string.

17. Select the Edit menu, then choose Paste and Match Style.
NOTE: We are not choosing Paste as it will not show the full link to the file.

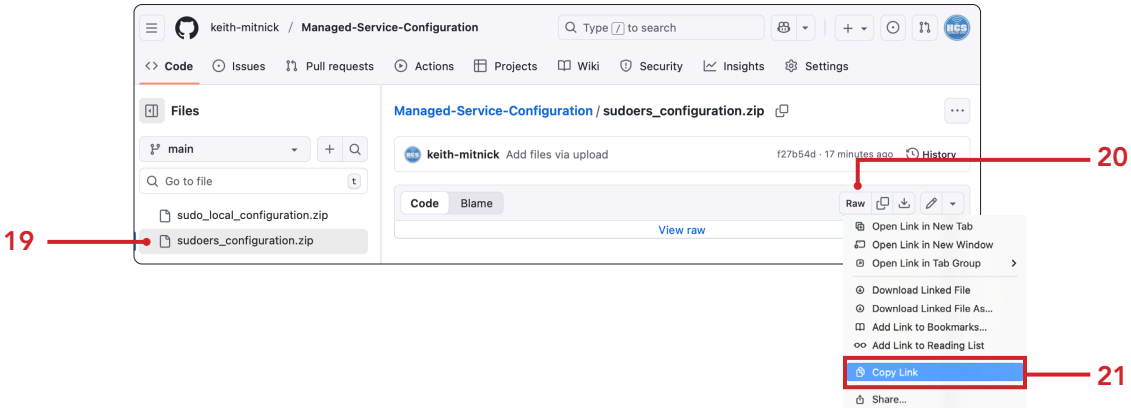


18. The direct link to the sudo_local file will appear under the TouchID hash string. Create a new line under the sudoers configuration hash string.

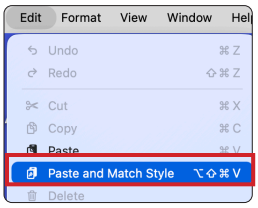
19. Switch back to GitHub and click on the sudoers_configuration.zip file.

20. Hold down the control key and click on the Raw button to show its menu.

21. Select Copy Link.

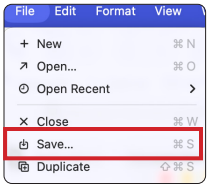


22. Select the Edit menu, then choose Paste and Match Style.
    NOTE: We are not choosing Paste as it will not show the full link to the file.



23. The direct link to the sudoers file will appear under the sudoers configuration hash string.

24. Choose the File menu and select Save.



25. Confirm the Hash file is on your Desktop.

This completes this section. In the next section, we will create and deploy a Blueprint in Jamf Pro.

Section 4: Create and deploy Blueprints with Jamf Pro

What You'll Need:
Learn what hardware, software, and information you'll need to complete the tutorials in this section.

Hardware and Software:
Requirements for following along with this section:

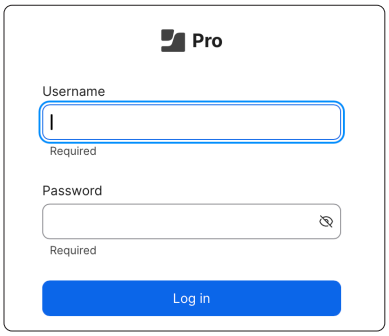
- A non production Mac computer with macOS 14 or later and administrative privileges
- Access to a Jamf Pro Server running 11.23 or later with administrative privileges
- Access to the file named Hash that we created earlier in this guide. It includes the SHA-256 Hash strings and GitHub URL's

In this section, we will walk through how to create Blueprints in Jamf Pro to deploy both the customized sudoers file and the Pluggable Authentication Module (PAM) configuration that enables Touch ID authentication for sudo in the Terminal. Blueprints provide a powerful way to deliver complex management workflows by leveraging the capabilities of Declarative Device Management (DDM). SSO needs to be enabled in your Jamf account in order to use Blueprints. You can find instructions to enable SSO in your Jamf account here:

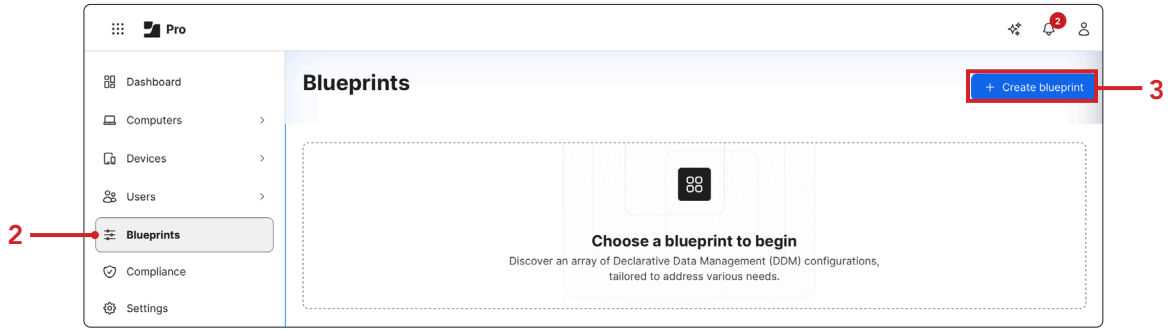
https://learn.jamf.com/en-US/bundle/jamf-pro-documentation-current/page/Jamf_SSO.html

Blueprints allow you to scope management settings to specific devices using modular, customizable components, such as payloads, configuration files, and service configurations, all organized in a single, unified location. By using Declarative Device Management (DDM), devices proactively and autonomously apply the settings defined in the Blueprint, verify their compliance state, and report any changes back to the Device Management Server asynchronously. This modern approach ensures that your macOS devices continuously maintain the intended security posture while reducing administrative overhead and improving reliability across your fleet.
NOTE: Blueprints can only be scoped to Smart Computer Groups and Static Computer Groups.

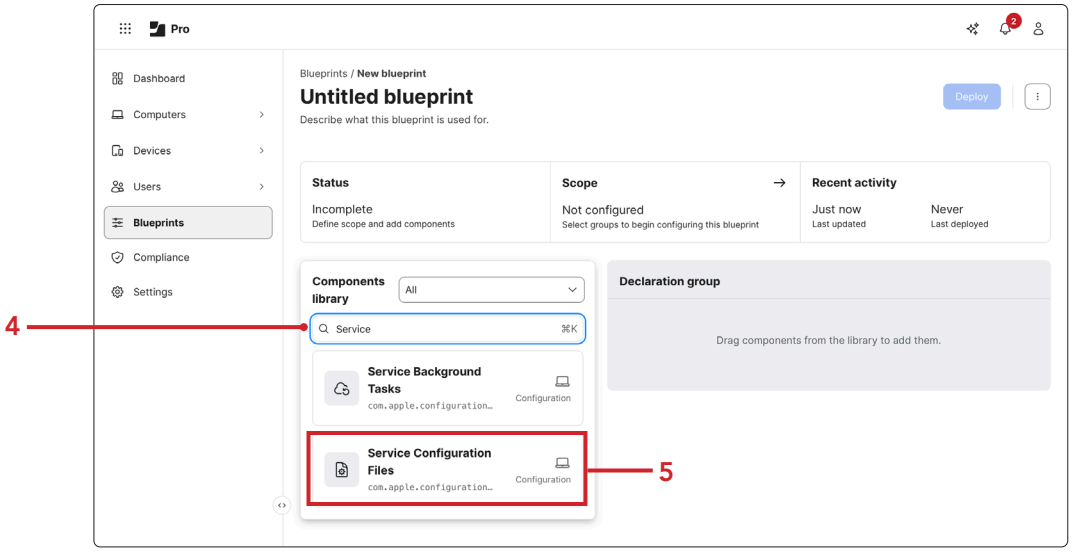
1. Log into Jamf Pro with administrative privileges.



2. Select Blueprints from the sidebar.
3. Click Create blueprint (+).

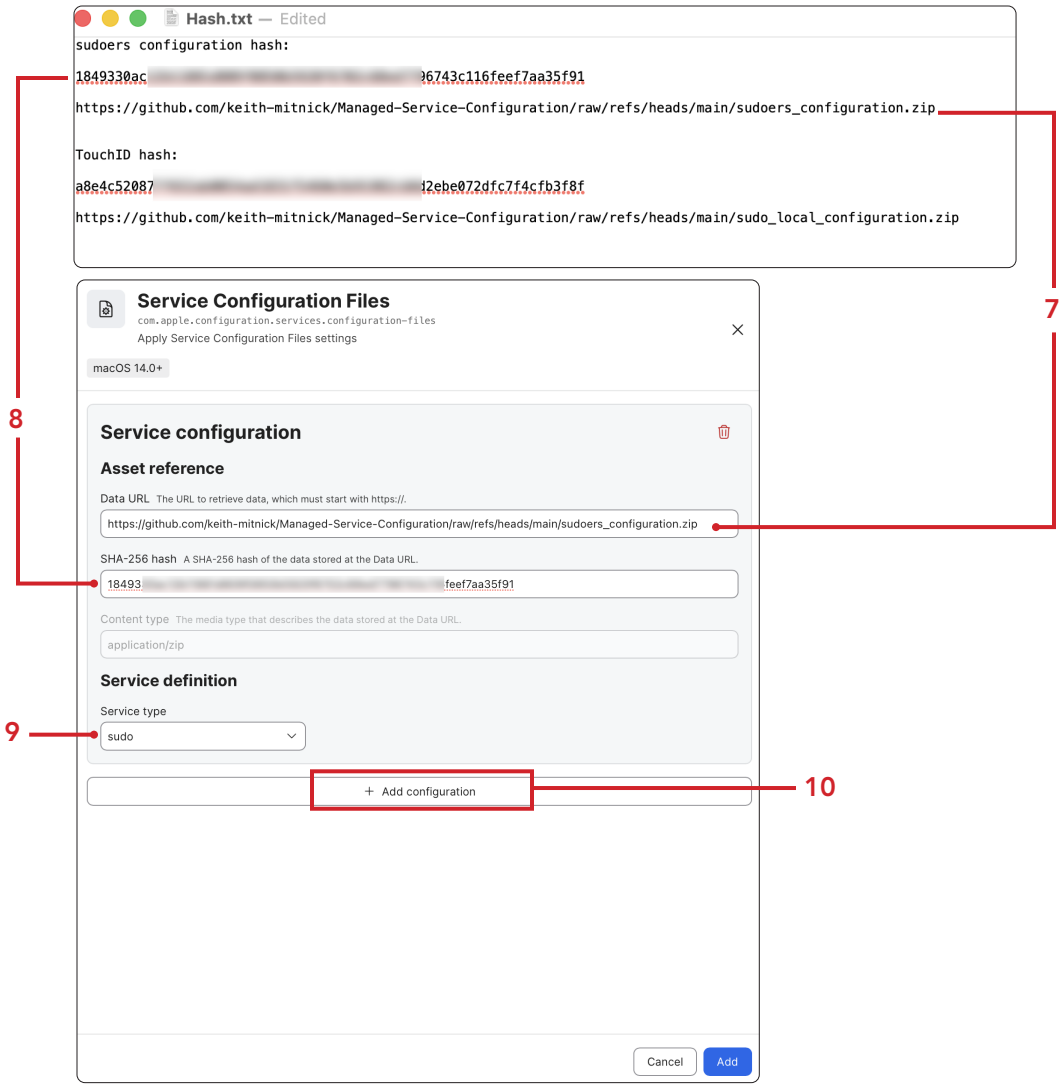


4. In the Components library, Enter **Service** in the search field.
5. Select Service Configuration Files.

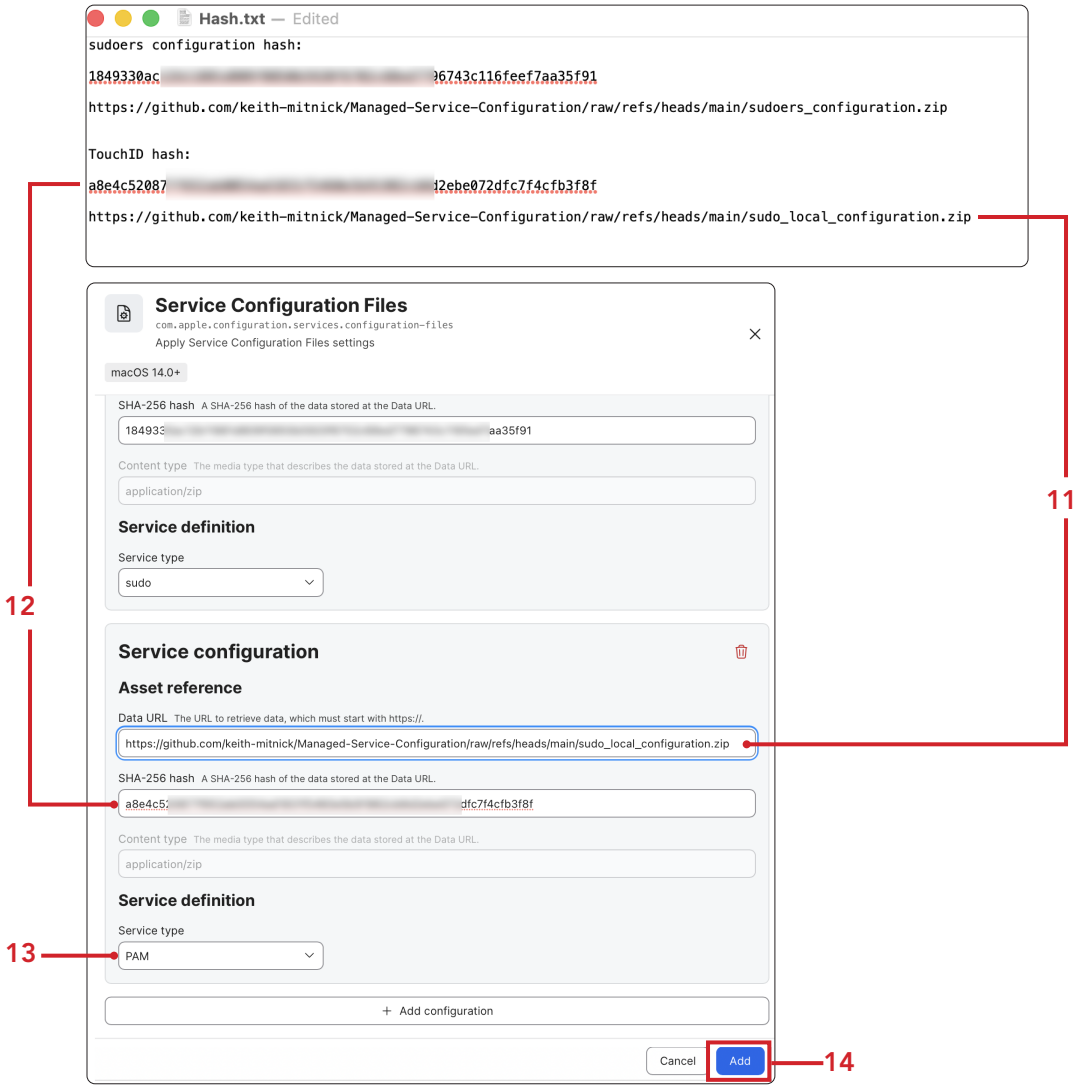


6. Open the Hash file that was saved to your Desktop earlier in this guide.

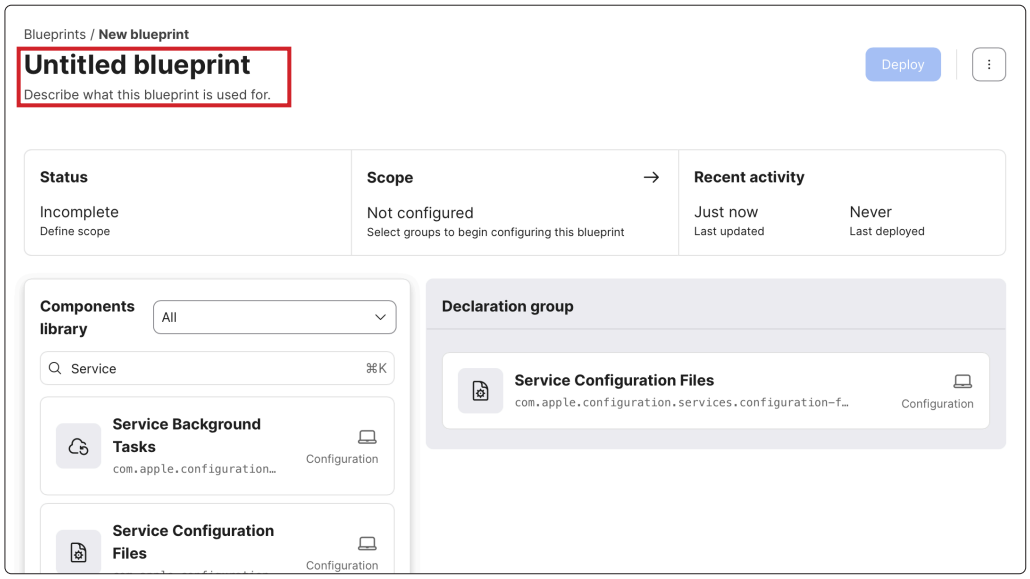
7. Copy the URL in the sudoers configuration hash section of the Hash file and paste it in the Data URL field (Must start with https://).
8. Copy the hash string in the sudoers configuration hash section of the Hash file and paste it in the SHA-256 field.
9. From the Service type pulldown menu, select “sudo”.
10. Click Add configuration (+).



11. Copy the URL in the TouchID hash section of the Hash file and paste it in the Data URL field (Must start with https://).
12. Copy the hash string in the TouchID hash section of the Hash file and paste it in the SHA-256 field.
13. From the Service type menu, select "PAM".
14. Click Add.



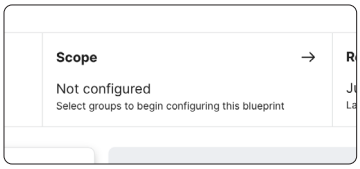
15. Single-click on Untitled blueprint to rename it.



16. Enter **sudo and Touch ID configuration** for the name.

17. Enter **This Blueprint deploys the sudoers file limiting the use of sudo to the jappleseed user on Staff laptops. It also enables Touch ID for terminal commands** for the Description.

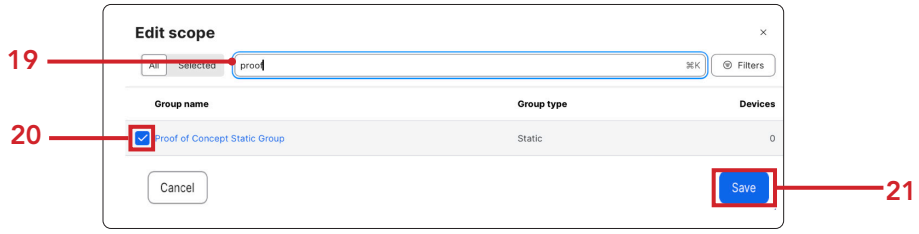
18. Click Scope.



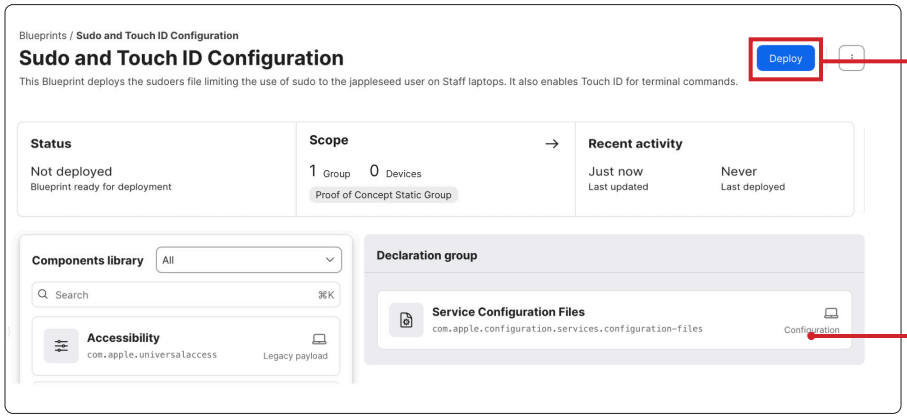
19. In the search field, enter **proof.**

20. Select the check box for Proof of Concept Static Group. (if you did not create this optional static group as outlined in the preface of this guide, select a non-production test group of your choosing).

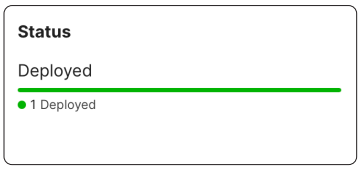
21. Click Save.



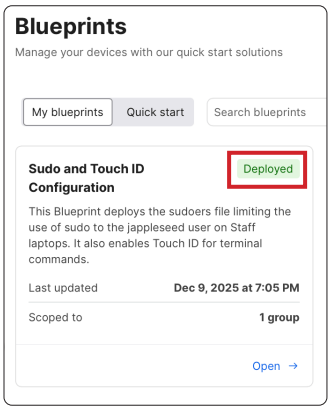
22. In the Declaration group section, confirm the Service Configuration Files is listed.
23. Click Deploy to deploy the Blueprint.



24. After a few moments, refresh the page. You will see the status as Deployed.



25. Confirm the status of the Blueprint shows as Deployed.



This completes this section. In the next section we will log into the Mac computer that we deployed the blueprint to to make sure all is working.

Section 5: Testing the Deployed Blueprint

What You'll Need:
Learn what hardware, software, and information you'll need to complete the tutorials in this section.

Hardware and Software:
Requirements for following along with this section:

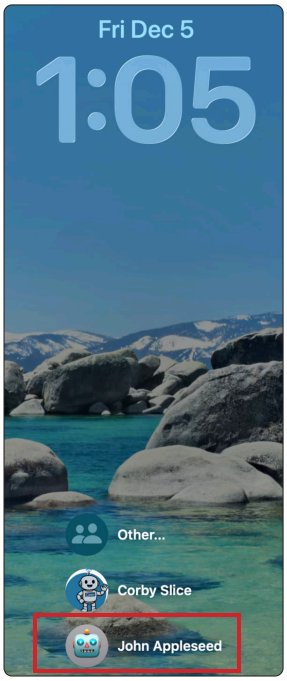
- A non-production Mac computer with macOS 14 or later and administrative privileges

Before beginning this section, make sure you have access to a secondary, non-production Mac computer that is part of the Proof of Concept Static Group in Jamf Pro. We scoped the Blueprint to that static computer group in section four of this guide. If you did not create the Proof of Concept Static Group, You can use an existing smart or static computer group but make sure only non-production Mac computers are in that group.

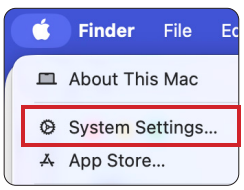
The non-production test Mac computer must already contain the jappleseed and cslice macOS administrative accounts, and Touch ID must be enabled for both accounts. The steps for creating these accounts and enabling Touch ID were outlined in the Preface. If those preparations are not yet complete, return to the Preface and finish them before continuing.

In this section, we will test our work to ensure the blueprint was deployed and the settings we configured are working properly.

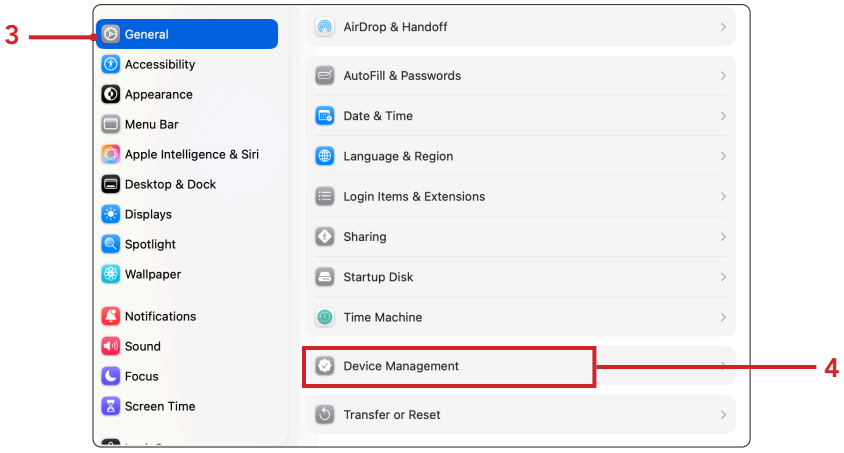
1. Log into your non-production test Mac computer as John Appleseed.



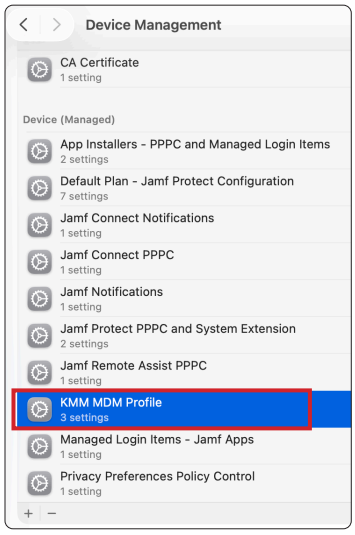
2. Click on the Apple logo and select System Settings.



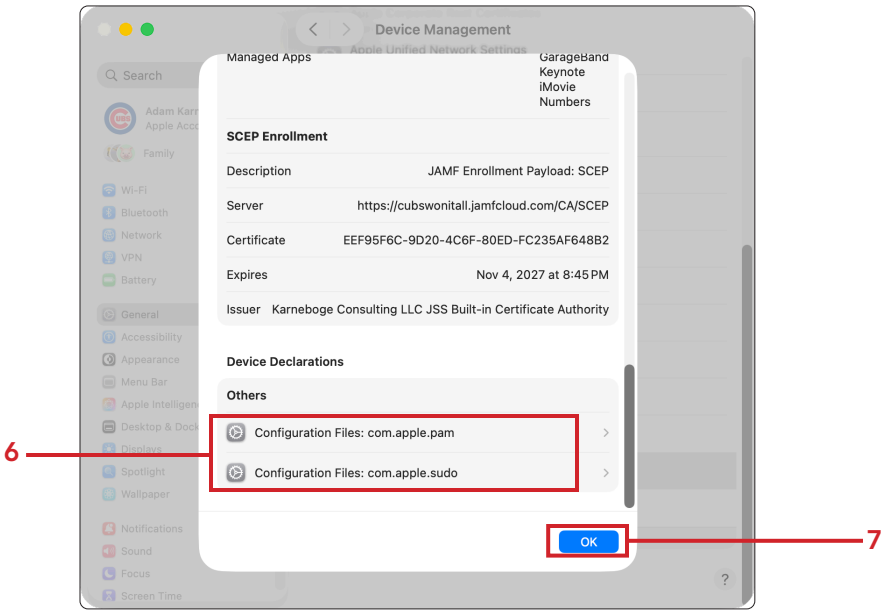
3. Click General.
4. Click Device Management.



5. Scroll down to the MDM Profile and double click the profile.



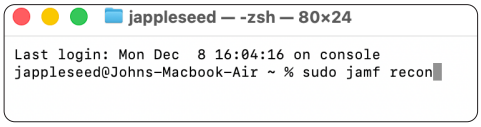
6. Scroll down to the Device Declarations section: Confirm the com.apple.pam and com.apple.sudo configuration files that were deployed with the blueprint are listed.
7. Click OK



8. Open Terminal located in /Applications/Utilities.

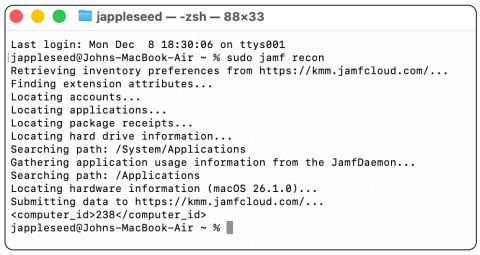
9. Run the following command:

```
sudo jamf recon
```

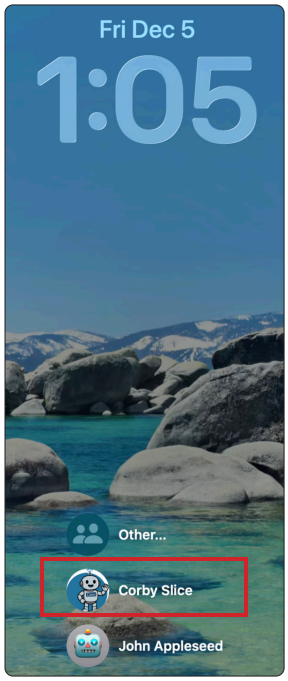


10. You will be prompted to use Touch ID or enter your password. Use Touch ID.

11. The command executes because jappleseed is listed in the sudoers file.



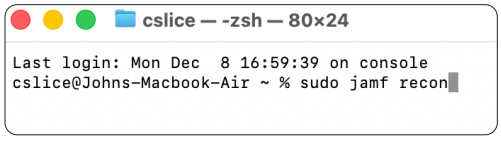
12. Logout out of the jappleseed account and login as Corby Slice.



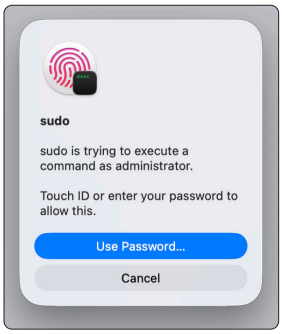
13. Open Terminal located in /Applications/Utilities.

14. Run the following command:

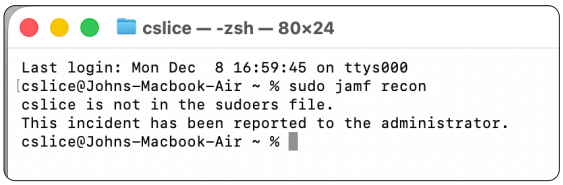
```
sudo jamf recon
```



15. You will be prompted to use Touch ID or enter your password. Use Touch ID.



16. The command will fail because the user cslice is not in the sudoers file. This is the expected result. The deployed blueprint is working correctly.



This completes this guide.