# Munki

Ref: <https://www.munki.org/munki/>

## MacOS Versions and Munki Versions

macOS 13 (Ventura) – Munki 5.6 or later

macOS 12 (Monterey) – Munki 5.6 or later

macOS 11 (Big Sur) – Munki 5.2

macOS 10.15 (Catalina) – Munki 5.2

macOS 10.14 (Mojave) – Munki 3.6

Munki 5.2.2 or later for use on Apple silicon

## Applications to Install

Google Drive for Desktop

Google Chrome

Slack

OSQuery

BitDefender

Adobe PDF Reader

## Introduction

Munki is used together with a webserver-based repository of packages, used to manage software installs (and some removals) on OS X client machines. Installs sw packaged in the Apple package format. Can also be configured to install Apple Software Updates, either from Apple’s server, or yours.

Releases page: <https://github.com/munki/munki/releases>

Munki auto-builds: <https://munkibuilds.org/>

Munki consists of client-side tools that are open source. Supports macOS 10.11 and later).

Munki Server – Can use any web server, where you need to be able to create directories and files.

Munki can install software delivered as **standard Apple packages**, as well as from **disk images** (disk image that gets dragged to the Applications folder.

Munki can also support “**Optional Software**”, made available on the client, so users can decide for themselves what items to install. This “self service” is similar to the self-service installs offered by Jamf Pro.

Munki can **update software** it did not install itself.

Munki can be configured to install **Apple Software Updates**, for users without Admin rights.

Related Tools: <https://github.com/munki/munki/wiki/More-Links-And-Tools#reportingweb-consoles>

## Munki Pieces

Munki uses three types of data:

1. **Installer Items** – Packages or Disk Images containing the software to be installed.
2. **Catalogs** – Lists of available software. Built using tools provided with Munki. Usually build from “pkginfo” files. The makecatalogs tool is used to build catalogs from pkginfo files. Stored on the web server as standard Apple plist files in text format. Pkginfo are also plist formatted files.

Pkginfo Files: <https://github.com/munki/munki/wiki/Pkginfo-Files>

Makecatalogs: <https://github.com/munki/munki/wiki/Makecatalogs>

1. **Manifests** – A list of what software should be installed on or removed from a given machine. Stored on the web server as standard Apple plist files in text format.

## Munki Behaviors

The managedsoftwareupdate process runs in the background of the client. It determines what is supposed to be installed or removed. For installs, the software is downloaded in the background. If there are changes, then:

1. If there is no user logged in, Munki proceeds to install or remove sw without asking. Does restarts if required.
2. If there is a user logged in, Munki will launch Managed Software Center to notify the user of available updates.

Admins can mark some updates as safe to install without user confirmation. These usually include a list of “blocking\_applications” so that Munki won’t try to automatically update sw that is in use.

Unattended Installs and Uninstalls: <https://github.com/munki/munki/wiki/Pkginfo-Files#unattended-installs-and-uninstalls>

Unattended\_install takes effect for managed\_installs and managed\_updates Manifest types, and unattended\_uninstalls for managed\_uninstalls.

Munki can also force installs: <https://github.com/munki/munki/wiki/Pkginfo-Files#force-install-after-date>

## Requirements

Munki 4 and 5 include Python.

Web server is simply a web server – no Munki tools installed.

A Mac is required to administer your Munki server – admin tools to create pkginfo files, catalogs and manifests. Needs to have the munkitools.mpkg installed.

User / Client Machines: munkitools.mpkg must be installed. Munkitools.mpkg contains 4 sub packages: app, core, launchd, and admin. Current packaged releases are available here: <https://github.com/munki/munki/releases>

The client installer package installs the munki command-line tools in /usr/local/munki, and the GUI Managed Software Center.app in /Applications/.

## Key Tools

### Admin Tools

munkiimport – helps an admin import software installation packages into the Munki server: <https://github.com/munki/munki/wiki/munkiimport>

makepkginfo – Creates basic pkginfo (metadata) for an installer package: <https://github.com/munki/munki/wiki/makepkginfo>

makecatalogs – Builds Munki catalogs from pkginfo files generated by makepkginfo or munkiimport: <https://github.com/munki/munki/wiki/Makecatalogs>

manifestutil – work with manifests: <https://github.com/munki/munki/wiki/manifestutil>

### Client Tools

managedsoftwareupdate – checks the server, builds a list of items that need to be installed or removed: <https://github.com/munki/munki/wiki/managedsoftwareupdate>

GUI application: Managed Software Center.app located at /Applications/Managed Software Center.app

## Demonstration Setup

Setting up a Munki client and server on a single machine – no separate web server needed. Quick and easy.

Ref: <https://github.com/munki/munki/wiki/Demonstration-Setup>

MacOS ships with Apache 2. Easy to set up a demonstration Munki server on any available Mac. Will set up Munki on a macOS without Server.app installed.

### Install the Client Software:

<https://github.com/munki/munki/releases>

Install munkitools-6.0.0.4520.pkg

### Building the “server” repository

Create a directory structure in /Users/Shared, and then configure Apache 2 to serve it via HTTP.

#### Building a “server” repository

From the Terminal:

cd /Users/Shared/

mkdir munki\_repo

mkdir munki\_repo/catalogs

mkdir munki\_repo/icons

mkdir munki\_repo/manifests

mkdir munki\_repo/pkgs

mkdir munki\_repo/pkgsinfo

chmod -R a+rX munki\_repo

Tell Apache 2 to serve the munki\_repo directory via HTTP:

sudo ln -s /Users/Shared/munki\_repo /Library/WebServer/Documents/

This creates a symlink inside /Library/Webserver/Documents that points to our new munki\_repo.

By default /Library/WebServer/Documents is Apache 2's DocumentRoot

#### Activate Apache 2:

sudo apachectl start

#### Populate the repo

To import packages, download the current munki installation package at: <https://github.com/munki/munki/releases/latest>

Install the Munki tools by double-clicking the installer package. Restart required.

Tools for admins are available from the command line and are installed in /usr/local/munki.

To import packages into the rep, use munkiimport – which needs to be configured before use:

bash-3.2$ /usr/local/munki/munkiimport --configure

Repo URL (example: afp://munki.example.com/repo): file:///Users/Shared/munki\_repo

pkginfo extension (Example: .plist): <just hit return>

pkginfo editor (examples: /usr/bin/vi or TextMate.app): TextMate.app <substitute your favorite text editor>

Default catalog to use (example: testing): testing

Repo access plugin (defaults to FileRepo): <just hit return>

Path to the Munki repo is: ?users/Shared/munki\_repo. This is what we enter wit the file:// prefix.

Specify an extension for pkginfo files: “.plist”

Editor to use for the pkginfo files: TextMate.app

Default catalog new packages/pkginfo should be added to. Use a “testing” catalog for this.

What Repo access plugin to use: leave blank, since we’ll be using FileRepo

### Example Package

Firefox from <http://www.mozilla.com> – a disk image with .dmg extension is downloaded to the Downloads folder.

To import:

##### Step 1:

bash-3.2$

/usr/local/munki/munkiimport ~/Downloads/Firefox\ 61.0.2.dmg

Item name: Firefox

Display name: Mozilla Firefox

Description: Web browser from Mozilla

Version: 61.0.2

Category: Internet

Developer: Mozilla

Unattended install: False

Unattended uninstall: False

Catalogs: testing

Import this item? [y/n] y

Upload item to subdirectory path []: apps/mozilla

Path /Users/Shared/munki\_repo/pkgs/apps/mozilla doesn't exist. Create it? [y/n] y

No existing product icon found.

Attempt to create a product icon? [y/n] y

Attempting to extract and upload icon...

Created icon: /Users/Shared/munki\_repo/icons/Firefox.png

Copying Firefox 61.0.2.dmg to /Users/Shared/munki\_repo/pkgs/apps/mozilla/Firefox 61.0.2.dmg...

Edit pkginfo before upload? [y/n]: y

Saving pkginfo to /Users/Shared/munki\_repo/pkgsinfo/apps/mozilla/Firefox-61.0.2...

##### Step 2: Run the munkiimport tool and provide it a path to our downloaded disk image

munkiimport then asks us to confirm or override some info about the package.

Accept the item name, but provide a new Display name and Description

Accept the version and the catalogs

Then munkiimport prints back our choices and asks if we want to import the item. Yes.

munkiimport asks us to upload the package to a subdirectory path: upload to a directory named “Mozilla” inside the directory named “apps”.

munkiimport looks for an existing icon for Firefox, offers to create one for us: Yes.

munkiimport copies the Firefox package to: /Users/Shared/munki\_repo/pkgs/apps/mozilla, and save the pkginfo to /User/Shared/munki\_rep/pkgsinfo/apps/mozilla/Firefox-61.0.2

Then asked:

Rebuild catalogs? [y/n] y

Single item is added to the testing catalog. To test, go to:

<http://localhost/munki_repo/catalogs/testing> - will see a property list which contains a pkginfo for Firefox.

##### Step 3: Creating a client manifest

Client manifest tells Munki what to install.

Use manifestutil:

% /usr/local/munki/manifestutil

Entering interactive mode... (type "help" for commands)

> new-manifest site\_default

> add-catalog testing --manifest site\_default

Added testing to catalogs of manifest site\_default.

> add-pkg Firefox --manifest site\_default

Added Firefox to section managed\_installs of manifest site\_default.

> exit

We have created a new manifest named “site\_default”. We have added “testing” to the list of catalogs to consult, and “Firefox” to the list of packages to install.

Check your work: http://localhost/munki\_repo/manifests/site\_default.

##### Step 4: Munki Client Configuration

Client stores its configuration in /Library/Preferences/ManagedInstalls.plist. Need to create it:

sudo defaults write /Library/Preferences/ManagedInstalls SoftwareRepoURL <http://localhost/munki_repo>

We’ve told the client tools the top-level URL for the munki repo: <http://localhost/munki_repo>.

Check your work:

defaults read /Library/Preferences/ManagedInstalls

##### Test the Munki Client Software

Run the Munki client from the command line to download Firefox:

sudo /usr/local/munki/managedsoftwareupdate

Managed Software Update Tool

Copyright 2010-2014 The Munki Project

https://github.com/munki/munki

Downloading Firefox 61.0.2.dmg...

0..20..40..60..80..100

Verifying package integrity...

The following items will be installed or upgraded:

+ Firefox-61.0.2

Web browser from Mozilla

Run

managedsoftwareupdate --installonly install the downloaded updates, or use the Managed Software Center.app to install.

##### Demonstrating Managed Software Center.app

/Applications/Managed Software Center.app

Can use this to install Firefox by clicking on **Update**.

**To remove the local Munki server**

sudo apachectl stop

sudo rm /Library/WebServer/Documents/munki\_repo

sudo rm -r /Users/Shared/munki\_repo

**To remove the local install of the Munki client and its data:**

<https://github.com/munki/munki/wiki/Removing-Munki>

sudo launchctl unload /Library/LaunchDaemons/com.googlecode.munki.\*

sudo rm -rf "/Applications/Utilities/Managed Software Update.app"

sudo rm -rf "/Applications/Managed Software Center.app"

sudo rm -f /Library/LaunchDaemons/com.googlecode.munki.\*

sudo rm -f /Library/LaunchAgents/com.googlecode.munki.\*

sudo rm -rf "/Library/Managed Installs"

sudo rm -f /Library/Preferences/ManagedInstalls.plist

sudo rm -rf /usr/local/munki

sudo rm /etc/paths.d/munki

sudo pkgutil --forget com.googlecode.munki.admin

sudo pkgutil --forget com.googlecode.munki.app

sudo pkgutil --forget com.googlecode.munki.core

sudo pkgutil --forget com.googlecode.munki.launchd

sudo pkgutil --forget com.googlecode.munki.app\_usage

sudo pkgutil --forget com.googlecode.munki.python