



Mac mini Dev Ops (for MacDevOps YVR 2018)

Hi, I'm an "itty-bitty" consultant from Winnipeg serving about 30 micro-businesses with offices situated in 6 Provinces. 18 of these locations have a macOS Server that I Admin. The smallest is for just 3 people in a Financial Services company and the largest is 50 people who work for a distributor with a head office in BC and a branch offices and servers in Alberta and Manitoba.

CLICK

MAC
DEV
OPS
YVP

Mac mini Dev Ops

or
JAMF Envy!

precursor.ca/slides

Copyright © 2018, Alex Narvey

So what I really do could be called “mac *MINI* Dev Ops”. And the way I do it is motivated by the fact that I have a case of Jamf envy. You see, I like what Jamf Pro does but the product is not really in the budget for either myself or my client base. But that’s where Munki came to my rescue. Mat X has asked me to give a short talk about my implementation of Munki One-to-Many and my Munki Self-Service items.

CLICK

MAC
JLV
OPV
YVP

Munki One to Many

precursor.ca/slides

Copyright © 2018, Alex Narvey

Lets start with Munki One to Many...

CLICK



As you know Munki runs on a Web Server and every one of my clients' offices has a Mac mini or a Mac Pro with macOS Server and a an instance of Carbon Copy Cloner for which they are using for backup. So no budget required to get Munki going!

CLICK



macOS Server can host the Munki primary repo and Carbon Copy Cloner can sync the package repo to each client location.

CLICK

MAC
3LV
0PT
YUP

Centralized Manifests



munki.consultant.com

precursor.ca/slides

Copyright © 2018, Alex Narvey

So I have Centralized Manifests maintained at my head office using the GUI interfaces of Munki Admin and AutoPKGR...

CLICK

Centralized Manifests



munki.consultant.com

Local Packages



munki.example.com

precursor.ca/slides

Copyright © 2018, Alex Narvey

And local package repositories set up at each client so they can enjoy LAN speed updating instead of Manitoba's ridiculously slow internet speeds.

CLICK

Centralized Manifests

SoftwareRepoURL *https://munki.consultant.com/munki_repo*

precursor.ca/slides

Copyright © 2018, Alex Narvey

I just create my Munki preferences file with a SoftwareRepo URL pointing to my office "munki.consultant.com"

CLICK

MAC
3LV
0PT
YUP

Local Packages

PackageURL *https://munki.example.com/munki_repo/pkg*s

precursor.ca/slides

Copyright © 2018, Alex Narvey

And each client's Munki preferences file specifies their own local server as the Package URL ie. "munki.example.com"

Here's a little animation of what it looks like...

CLICK

MAC
DEV
OPS
YUP



munki.consultant.com

precursor.ca/slides

Copyright © 2018, Alex Narvey



And here's what it looks like one munki primary server spread to many local client package repos.

CLICK

MAC
JLV
0BT
YVP



munki.consultant.com

precursor.ca/slides

Copyright © 2018, Alex Narvey



Pushing the repo to each client is great for their local computers to get their updates at LAN-based speeds.

But these local Munki repos are only available inside the office and are not served to the internet. That is partially because of security and partly because my clients' upload speeds are often very slow. So what about computers that leave the office?

CLICK

Manifest



munki.consultant.com

Packages



munki.example.com

precursor.ca/slides

MAC
DEV
OPS
YVP

Manifest



munki.consultant.com

Packages



munkix.consultant.com

precursor.ca/slides

Copyright © 2018, Alex Narvey

When this MacBook Pro leaves the office DNS will make sure that it uses the package repo from my speedy cloud source.
CLICK.

MAC
3LV
0PT
YUP

Split Horizon DNS



A record

munki.example.com = 10.0.100.10



URL Forward

munki.example.com = munkix.consultant.com

precursor.ca/slides

Copyright © 2018, Alex Narvey

When inside the office the Mac gets its DNS from macOS Server which provides an A record to serve the local subnet address for the package repo.

CLICK

And when outside the office the external DNS source does a URL forward to our inexpensive and speedy cloud-based repo.

CLICK.

MAC
DEV
OPS
YUP

munkix.consultant.com



Package Repo in Linode (5\$ / mo.)

<https://www.linode.com/docs/getting-started>

<https://clburlison.com/munkirepo-guide-part-1/>

<https://www.digitalocean.com/community/tutorials/how-to-set-up-http-authentication-with-nginx-on-ubuntu-12-10>

precursor.ca/slides

Copyright © 2018, Alex Narvey

Based on a suggestion by J.D. Strong at MacDevOps 2017 I chose Linode as my cloud repo. Its very inexpensive and easy to use.

I've included some links to tutorials which show how to do it and they are available in the slide download.

CLICK

MAC
JLV
OPV
YVP

Munki Self-Service

precursor.ca/slides

Copyright © 2018, Alex Narvey

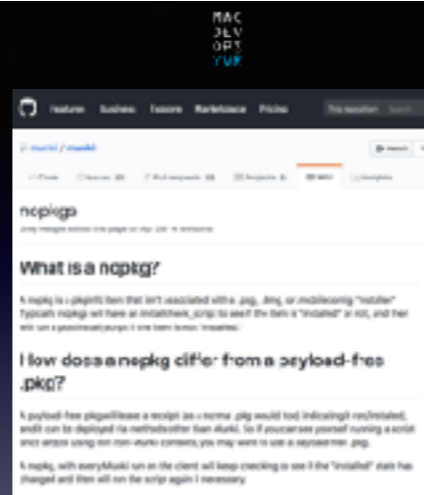
Now on to Munki Self Service items.

CLICK



I was impressed by the Jamf Self Service panel

CLICK



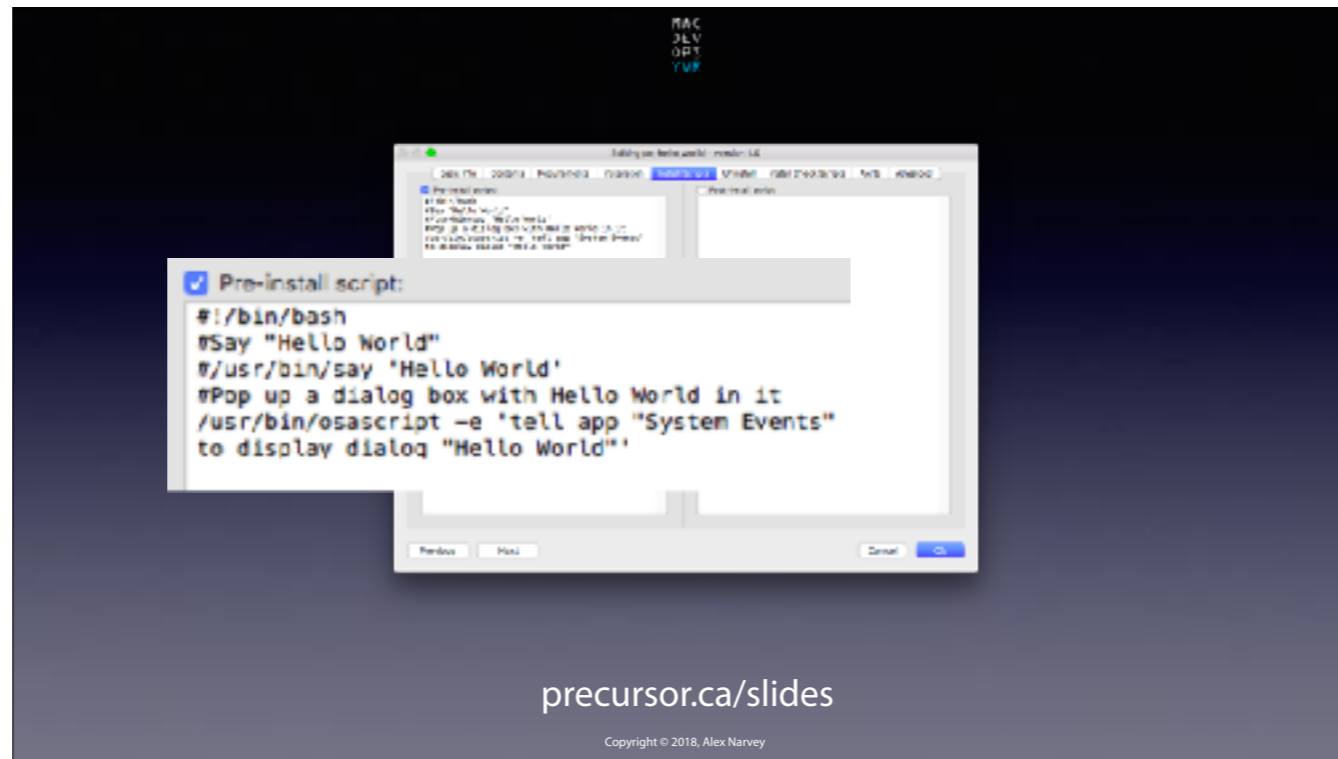
<https://github.com/munki/munki/wiki/nopkg>
<https://groups.google.com/forum/#!topic/munki-dev/KW9OeH4jtYI>

precursor.ca/slides

Copyright © 2018, Alex Narvey

It turns out that Munki can provide similar functionality utilizing On Demand items which have no package and perform scripts as root.

CLICK



Then put your script into the Install Script pane. Here is one for a “Hello World” script.

CLICK

PS DNS flush DNS cache



```
#!/bin/bash
# Define variable(s)
os=$(/usr/bin/sw_vers -productVersion)
#Test OS version and use the required flushstring
if [[ $os = 10.10.0 ]]; then
    flushstring="discoveryutil udnsflushcaches"
else
    if [[ $os = 10.10.1 ]]; then
        flushstring="discoveryutil udnsflushcaches"
    else
        if [[ $os = 10.10.2 ]]; then
            flushstring="discoveryutil udnsflushcaches"
        else
            if [[ $os = 10.10.3 ]]; then
                flushstring="discoveryutil udnsflushcaches"
            else . . .
```

precursor.ca/slides

Copyright © 2018, Alex Narvey

Here is one that helps the user flush the DNS cache without restarting.

CLICK

PS Empty Trash force empties the trash



```
#!/bin/bash
# Set the Variables
USERLOC="/Users"
THEUSER=$(who | awk 'NR==1 {print $1}' )
USERPATH="${USERLOC}/${THEUSER}"
TRASHPATH="${USERPATH}/.Trash"
#Force the Trash of all mounted volumes to empty
/bin/rm -rf $TRASHPATH /Volumes/*/.Trashes
```

precursor.ca/slides

Copyright © 2018, Alex Narvey

Here is one that helps the user force empty the Trash.

CLICK

MAC
JLV
0PT
YUP

PS Rebuild Mail

Rebuilds Mail.app envelope files



```
#!/bin/bash
# Apple Mail envelope rebuilding script
# Parts based on the Bash Script of Paul Galow https://github.com/pbihq/too
# Which was based on the AppleScript by Brett Terpstra http://brettterpstra
mail-dot-app-on-el-capitan/
# Start script
clear
echo "Apple Mail Database rebuild started..."
# Define variable(s)
os=$(sw_vers -productVersion)

. . .
```

precursor.ca/slides

Copyright © 2018, Alex Narvey

This rebuilds Mail.app's Envelope files in case Mail is acting up (which never happens ;-).

CLICK

MAC
JLV
0PT
YUP

PS CC Updater Kicks off Adobe CC RUM update



```
#!/bin/bash
# Begin updating Adobe Creative Cloud

# killing all Adobe Apps (contributed by Oliver Hetzner)
adobepid=$(/usr/bin/pgrep Adobe)
bin/kill -9 $adobepid

#running the manager
/usr/local/bin/RemoteUpdateManager

# giving the User a dialog that we are ready now.
# (contributed by Oliver Hetzner)
/usr/bin/osascript<&t;&t;END

. . .
```

precursor.ca/slides

Copyright © 2018, Alex Narvey

This one lets a standard user kick off an Adobe Creative Cloud update session.

CLICK

Scripting Tips

1) Don't forget to start with the Shebang

`#!/bin/bash` or `#!/bin/sh`

2) Use full path names

`/usr/bin/osascript` instead of `osascript`
`/Users/username/` instead of `~/`

3) Avoid XML characters ">","<" and "&"

use `>` for `>`
 use `<` for `<`
 use `&` for `&`

precursor.ca/slides

Copyright © 2018, Alex Narvey

Some scripting tips: Don't forget to kick off your script with the Shebang.

CLICK

Use the full path name for a tool

CLICK

Specify the full path to the Home directory

CLICK

These Munki plist are XML files so avoid using special XML characters like "greater than""less than" and "ampersand"

CLICK

MAC
DEV
OPS
YUP



#!
Scripting OS X
#tohellandback

**Control ssh access with munki
nopkg scripts**

<https://scriptingosx.com/2014/12/control-ssh-access-with-munki-nopkg-scripts/>



#!
Scripting OS X
#tohellandback

**Control Apple Remote Desktop
Access with Munki**

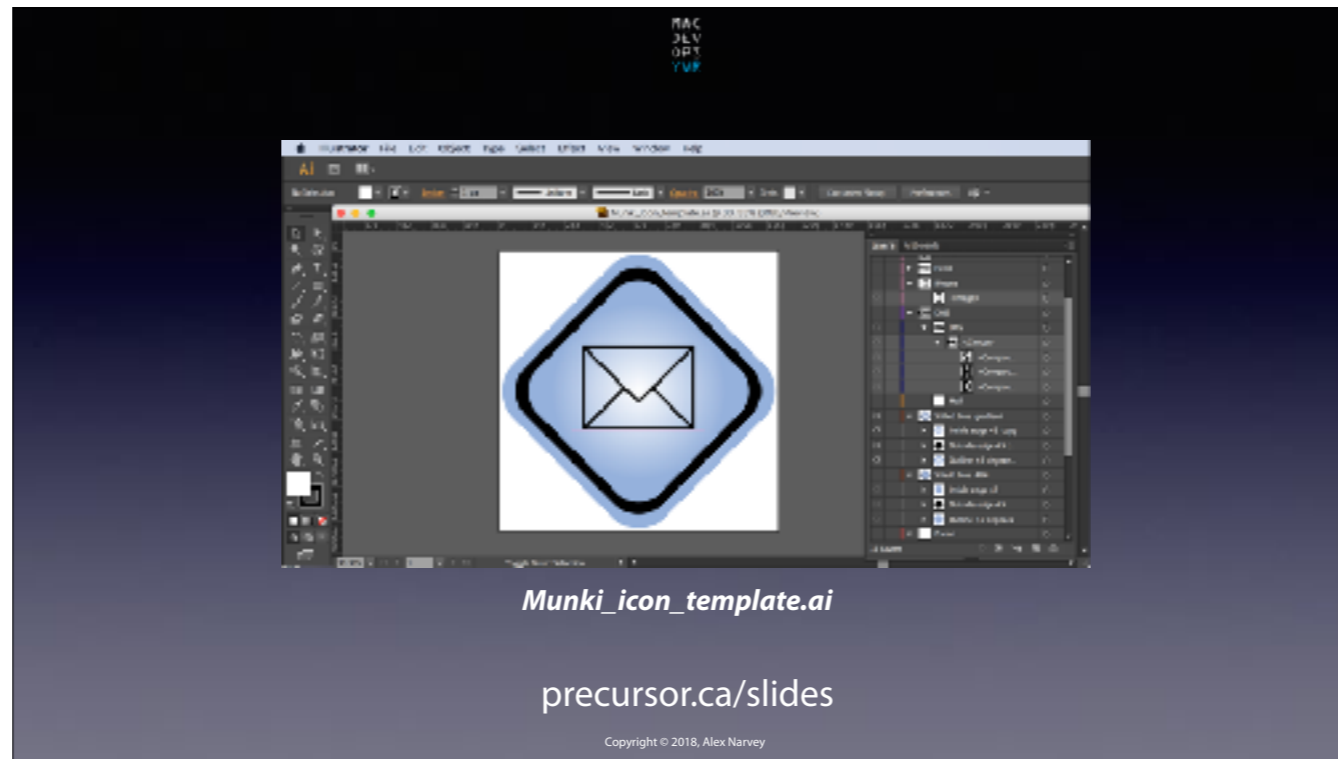
<http://scriptingosx.com/2016/01/control-apple-remote-desktop-access-with-munki/>

precursor.ca/slides

Copyright © 2018, Alex Narvey

Some people are a lot better at scripting than I am. E.G. on OSXscripting.com you can find some Munki “nopkg” scripts that have built-in check routines to find out if they already have been accomplished or still need to be done.

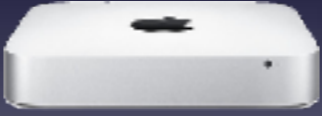

CLICK



Mat X was particularly interested in how I made the icons. Actually that part was the most fun.
I made a multilayered Adobe Illustrator template. Its available from the Slide download link if you want it.
CLICK

MAC
3LV
0PT
YUP

The Future?



DNS
AFP
OD Master
SMB
WebServer

VPN
Caching Server
DHCP
WebDAV for iOS

TimeMachine

precursor.ca/slides

Copyright © 2018, Alex Narvey

So you have probably figured out that I am one of those poor slob who enabled practically every service that macOS Server has to offer at every site I Admin. And you know that Apple deprecated them all.

So what does the future hold?

CLICK

MAC
DEV
OBT
YUP

The Future?



Synology + DEP Munki bootstrapping from the Cloud?

precursor.ca/slides


Copyright © 2018, Alex Narvey

Right now it is looking like Synology to take over for most functionality of macOS Server and moving Munki fully into the Cloud to be used in DEP Munki bootstrapping. I already have DEP Munki bootstrapping working at one client but I was waiting for WWDC before I finalize my plans and begin deploying at all my locations.

CLICK

MAC
DEV
OPS
YUP

Alex Narvey

 <https://Precursor.ca>

 @precursorca

 github.com/precursorca

precursor.ca/slides

Copyright © 2018, Alex Narvey

Thanks for listening. I hope some of you smaller operations find it useful.

THE END.