Open (and/or Free) vs Closed Source

AKA Steel Cage Death Match

Ben Toms and Allister Banks

Feedback: http://j.mp/psumac2015-91

Worksheet: http://url.aru-b.com/openVs

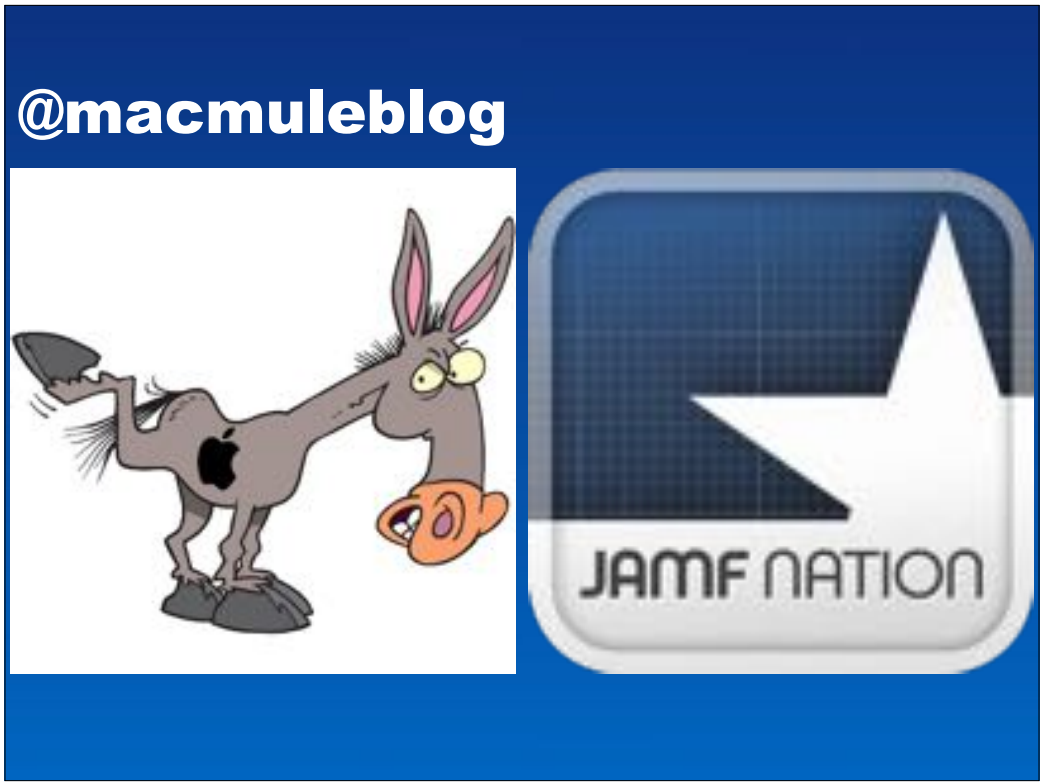
Open (and/or Free) vs Closed Source

AKA Steel Cage Death Match

Ben Toms and Allister Banks

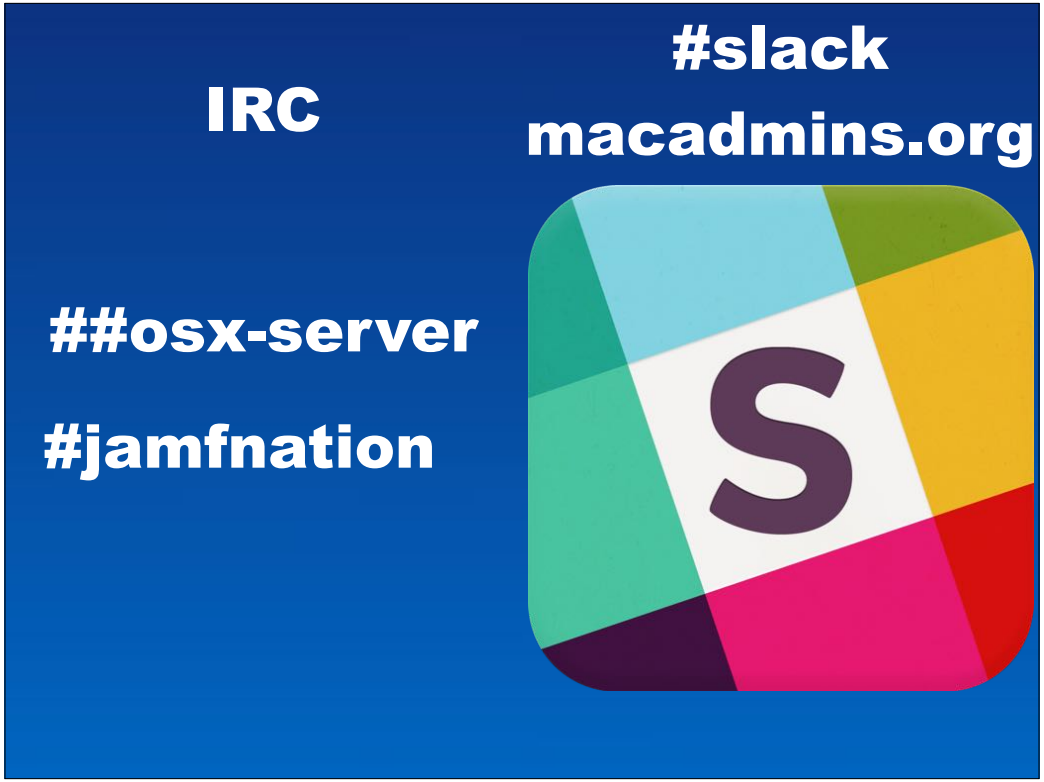
Feedback: http://j.mp/psumac2015-91
Worksheet: http://url.aru-b.com/openVs

Hello! My name is Allister Banks, the gentleman next to me is Ben Toms, and welcome to our steel cage death match. About me, I've had a pretty good year: I've been peter-principle promoted to technical lead of the Apple support department at a place called Montefiore Information Technology, which watches over 25000 total employees, over 700 of which are in IT, 8000 devices use activesync to check mail, but enough numbers. I had the honor of writing a book with Charles Edge, an article on autopkg for MacTech Magazine, and had my first pull request merged in Munki yesterday, which some of you may have seen in Greg's presentation earlier, and it was my 15th wedding anniversary…



My name is Ben Toms. You might know me as macmule from my blog with the tagline “Doing the donkey work to make you look like a smart ass!”

I’m also one of the deans of JAMFNation



A mod on the OSX-Server & JAMFNation IRC channels And an admin on the MacAdmin slack team.

And a co-founder of London Apple Admins with Graham Gilbert, the man who at this very conference last year taught america to faff.

faff - a definition

(UK, slang) To waste time on an unproductive activity.

“I spent all day faffing around creating NetBoot Images.”

“Creating NetBoot Images is a faff.”

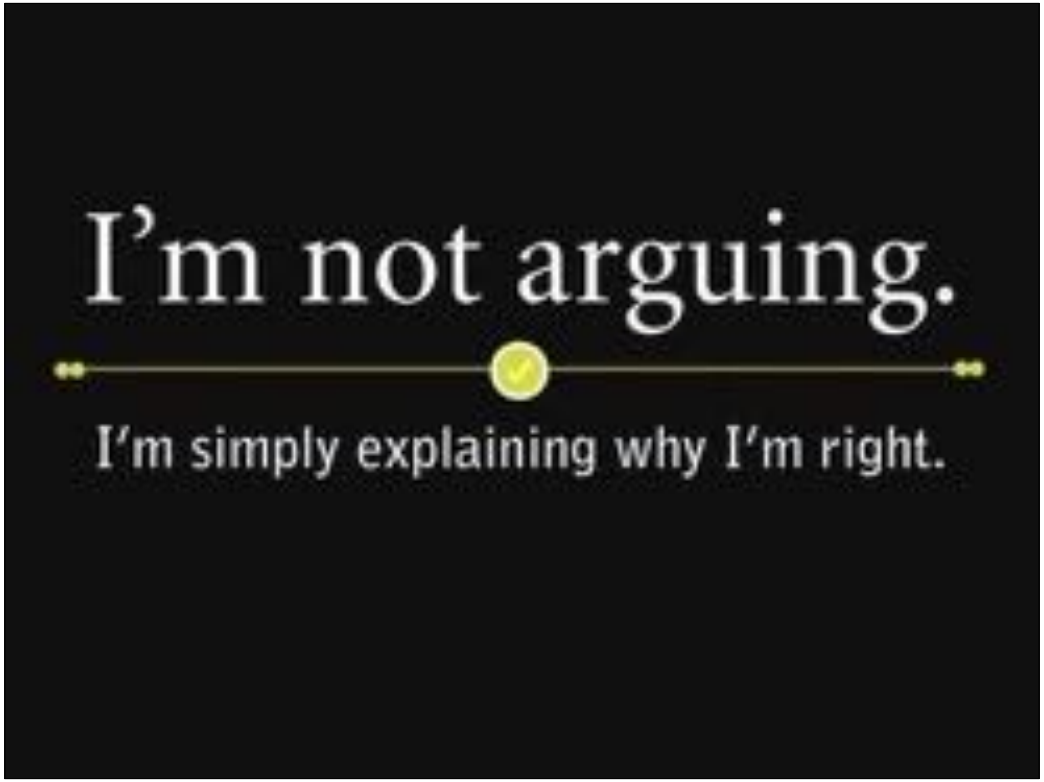
And over the past year I have been doing my hardest to win “Most breakable Admin of the year award”

From being mugged in New York & breaking my right ankle in January.. To 3 months later breaking my left foot falling down the pictured stairs on a holiday in Spain

But the community have been awesome, from cards wishing me well.. To.. well.. cheeky gits.

You might be able to hear another issue in my voice.. so bare with..

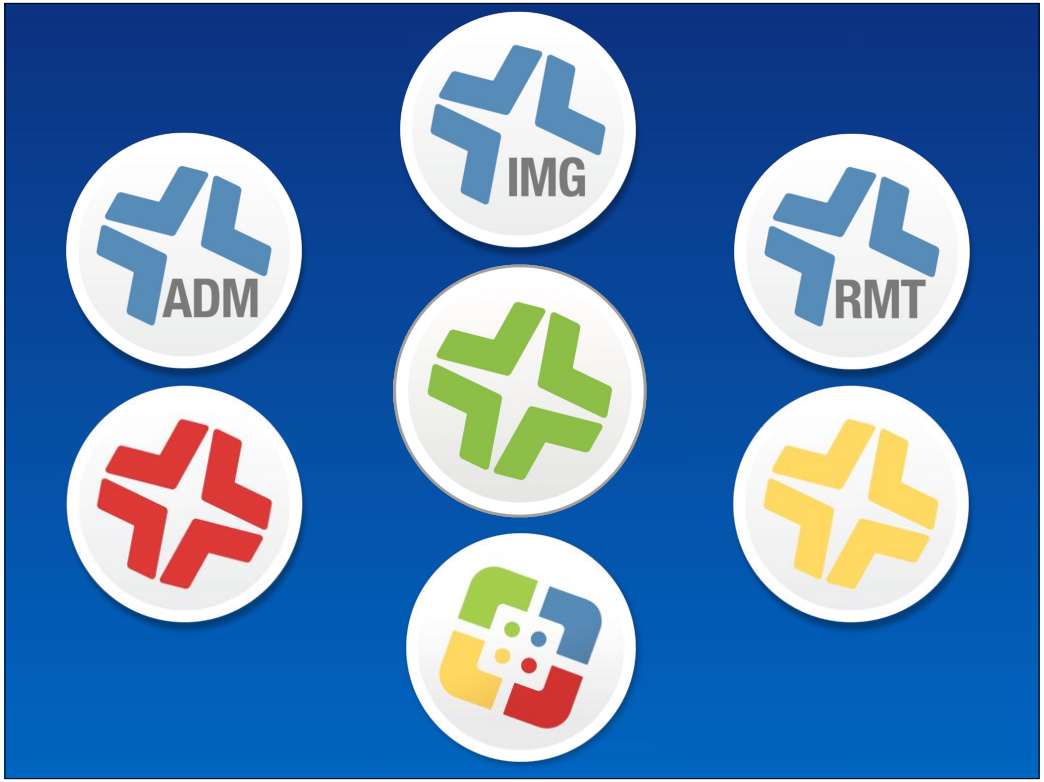
(Allister:) You'll get used to this cadence of I shoot of my mouth first and Ben gets time to attempt to refute my unassailable logic, it's kind of like the closing arguments in a trial that the lawyer for the defense has the benefit of being the last thing the audience hears. My intention is to put his way of doing things and the vendors he represents for the purposes of this session on the defensive, since they've made that wonderful faustian bargain called capitalism where they've exchanged money for good and services, because it is my thesis that the open source and free tools are actually a better deal overall…



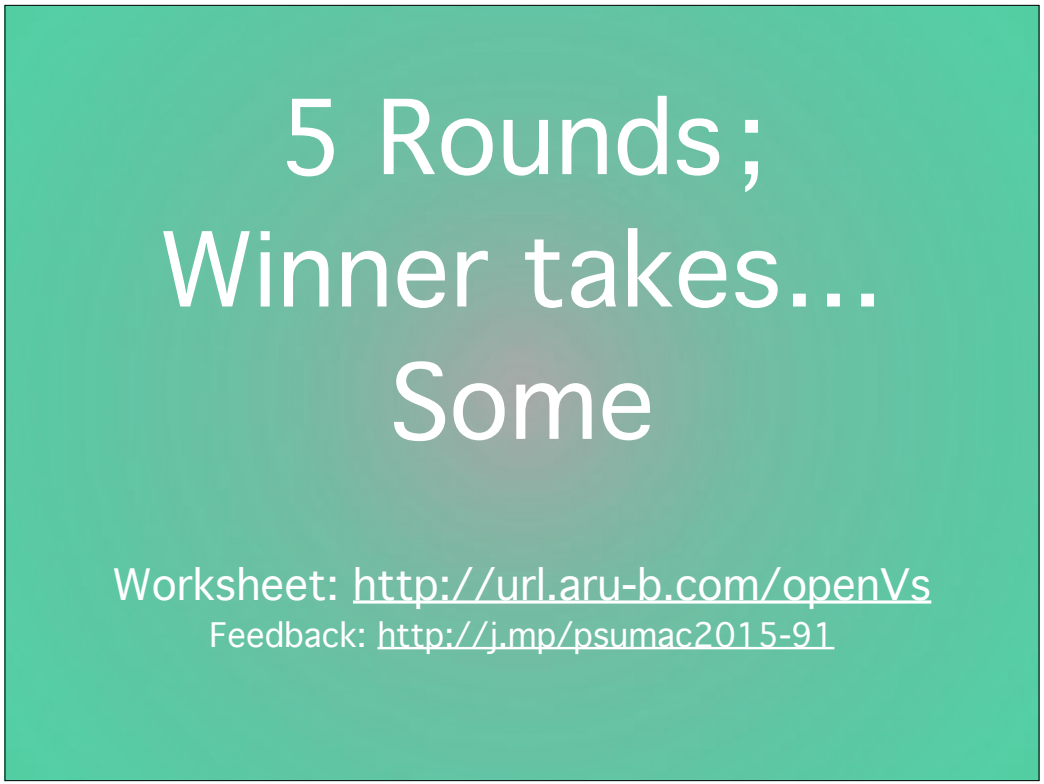
(Ben:) I'm going to attempt to show that using a paid for solution is neither

burying your head in the sand..

or making a deal with a devil…



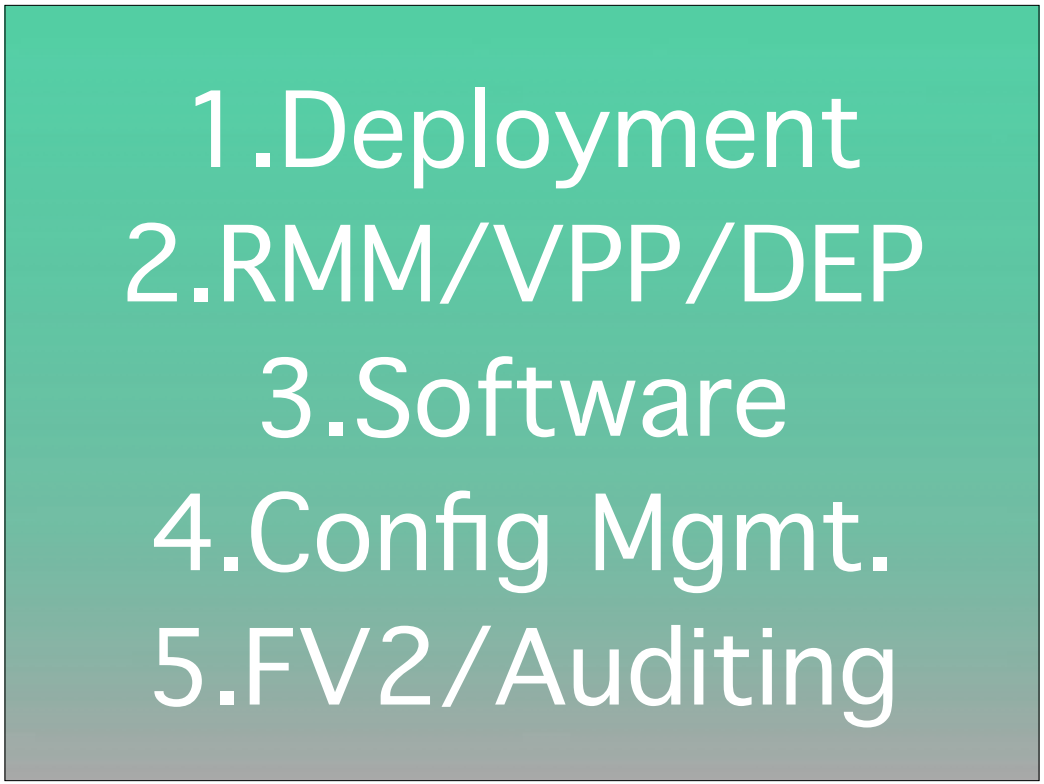
I'll be using examples with the Casper Suite, as this has been my bread & butter for the past 7 of the 13 years of me being an admin & showing a number of facets of lifecycle management that can be achieved by using the Casper Suites apps alone.



(Allister:)Breaking down the rules of this match, there are 5 rounds covering the various aspects of Mac lifecycle mgmt, and on the worksheet we've put a table for you to be able to keep track of who you think 'wins' each round, and at the very end AFTER Q&A we'd like just a show of hands as to who you thought won the whole thing.
Everybody who wants it has plugged in that link and downloaded it? Ok, moving on.
As you can see on there, we've come up with these 5 sections:

Image creation, restoring that image or otherwise deploying a thin or no-image workflow, RMM stands for remote monitoring and management, which I'm combining with orchestration for when you need to do maintenance en masse, and VPP along with DEP is important to more and more of us for 'modern' mgmt,

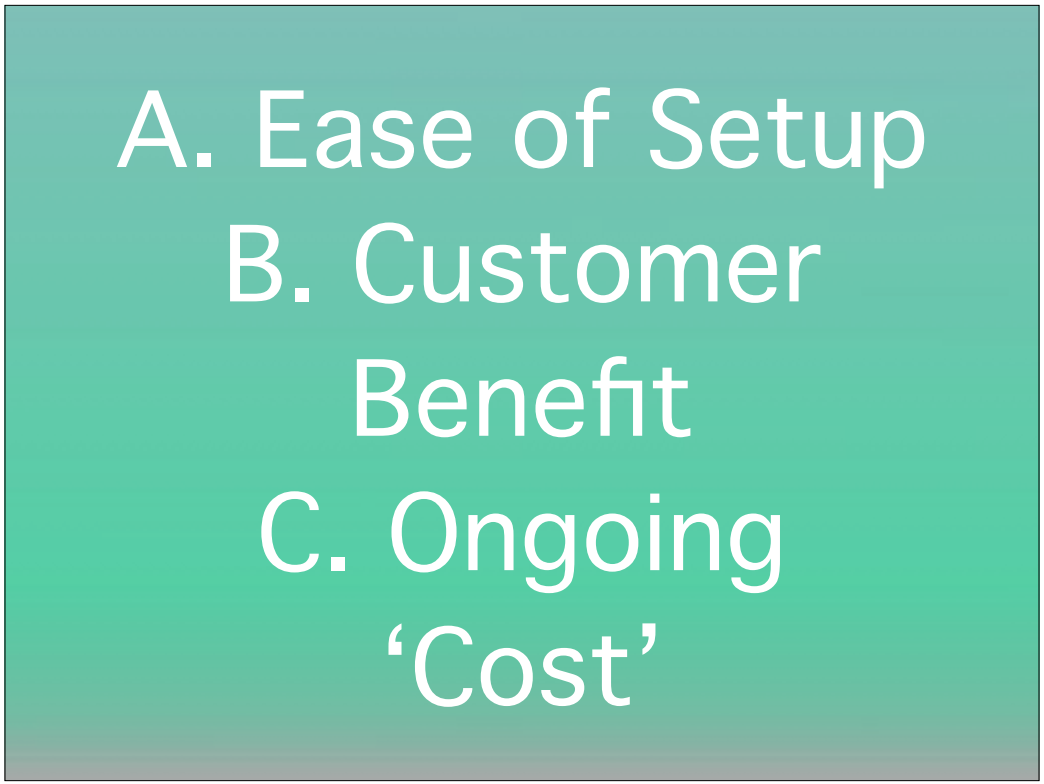
Making packages as necessary, along with Software installation and updates is big, as is an interface to either request other software or perform ad-hoc tasks,
Deploying what was primarily managed preferences as has now completely moved away from file-based mgmt with configuration profiles as one aspect, but per-user and less easy-to-manage things are important as well,



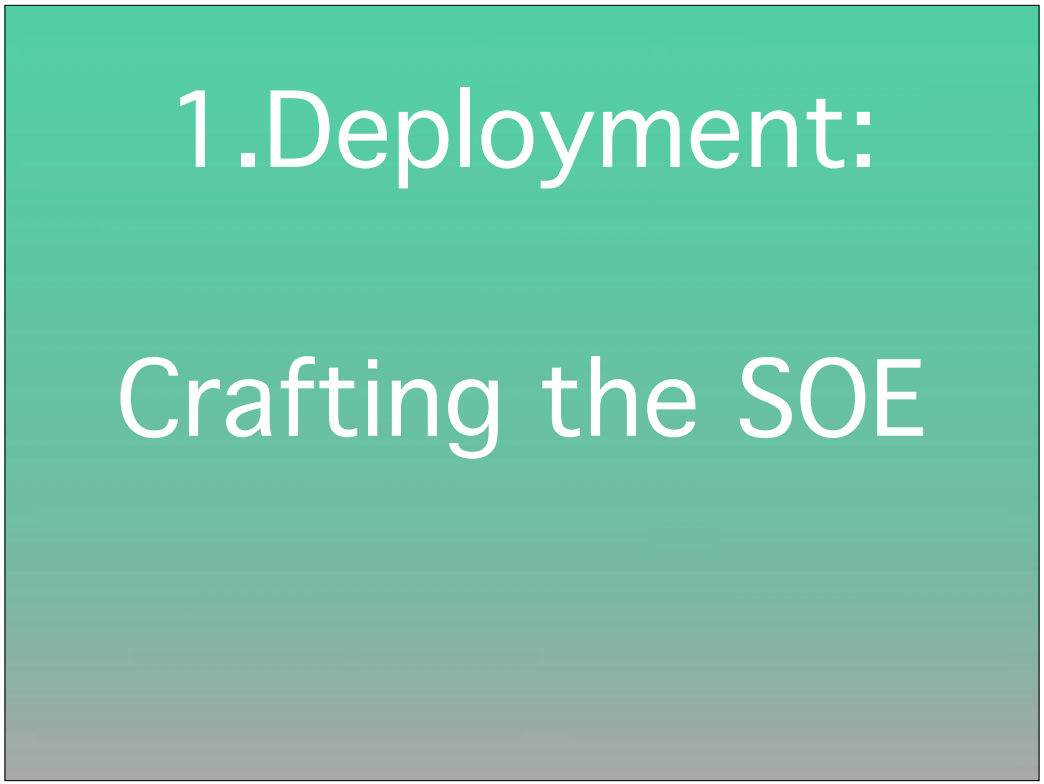
And finally when we're encrypting machines we need a workflow around escrowing the recovery key or otherwise managing it, and for ongoing capacity and licensing mgmt we think inventory/auditing and reporting tools are critical to have.

Judgement Criteria:

As we'll be talking about fundamentally different types of products, we came up with three pieces of criteria for you to use when judging the case we're making in each of those five categories:



Which presents the overall better admin setup experience (or install), which has more customer-facing benefits, and who has the lower ongoing maintenance load(for upkeep). Everybody understand the rules of the game? Check for weapons? Ready? Let's Go! First set of tasks!



Let's get right into it, round one, arriving at the SOE! I consider that we've ingested a computer into mgmt if it has our standard operating environment, meaning mgmt tools, a set of apps, and is using domain accounts, all wrapped in the security mandated by HIPAA along with things we opt into. When I came on the scene at my current job there were plenty of machines already out there, and very little expertise or familiarity with tools among the members of my team, they weren't imaging, they made bespoke, hand-crafted special snowflakes.

So we skipped de-programming people from golden master imaging and could go straight into better practices. The jobs involved with the SOE are making a modular image in a repeatable way, and we still even do image new machines because it's faster than certain pkg installs, even if we were thinking we'd point customers at office2016 from the Mac App Store once it's released, that's 700 megs per product for Excel Powerpoint and Word. Also we need to often migrate previously used machines. And we want to restore that image in a bunch of flexible ways.

First, we use CreateUserPackage by Per Oloffson, it's a GUI app that's pretty straightforward to setup our local admin as a fallback for the admin groups you'd have in AD

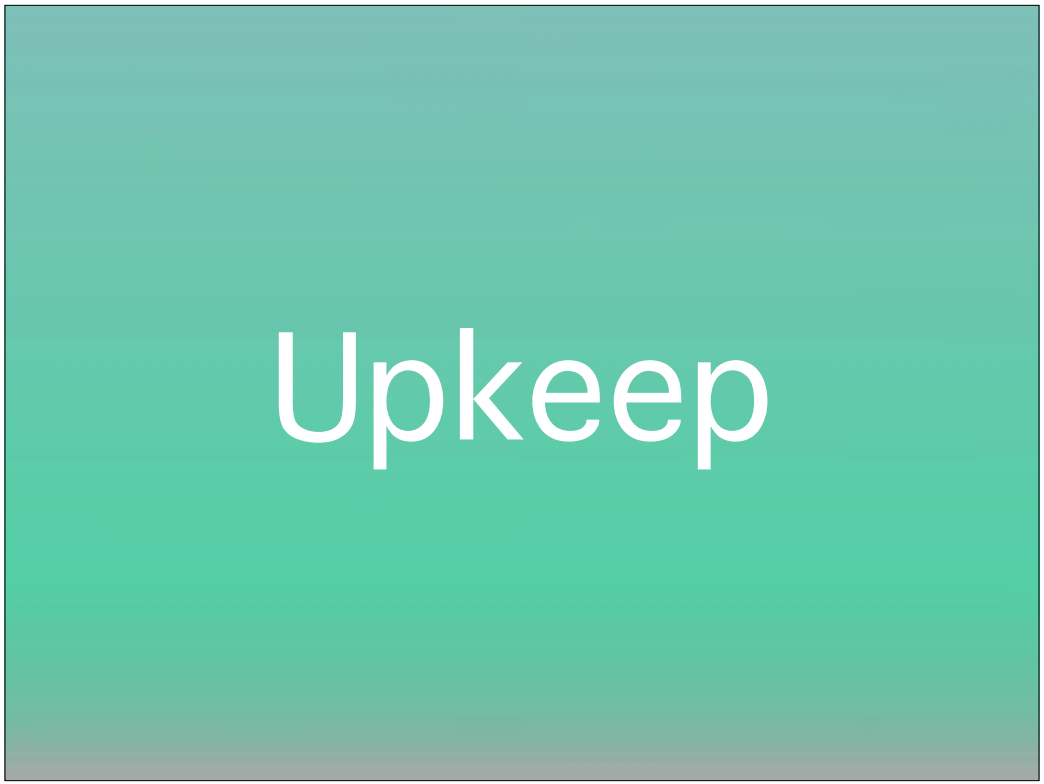
Then we make a base bare image, not even updates added, of the most current Install OS X app, yay for a unified 10.10.4, with the AutoDMG GUI. With our fastest I/O machine which is the blackmac/darthVader's bong that takes about 15 minutes.

Then we use the template feature of AutoDMG to add in our createUserPackage and the 2011 office suite SP4 minus outlook - of you're not on Office365 accounts come talk to me if you have questions about that - and the last ingredient there is any Apple updates like Safari iTunes or security patches, which AutoDMG fetches for you and the community collaborates on. That takes less than 10 minutes to cook,

and we then update a DeployStudio workflow that is named with the month we last cooked an image. For testing on newly-released builds and to keep around older images we have multiples of these nested DS workflows, and I'm not going to really dive into DS or munki or configurations like bootstrapping filevault2 yet, but to keep it simple for now the overarching steps DS delivers for us are: prep munki and the configuration mgmt tool puppet, prep Joseph Chilcote's tool outset which performs firstboot tasks for us, let deploystudio set the computername and prep for binding, and we happen to use firmware passwords as well.

We update images a little more than once a quarter and rotate our binding password regularly, but install-wise there are setup assistants and GUIs that lead you through most tasks that work pretty reliably and update often enough to take into account curveballs that are thrown.
We also use this netboot-based process for migrating into mgmt, including createOSXinstallerPackage for updating older versions and a custom tool to convert local home folders to the customers AD account.
And for putting the current Quality-assured software we maintain on machines, Munki layers all those frequently-updated apps and even delivers certain mgmt tools and settings to the point that very little needs to be in the image.

It's less than a half hour process with only communicating with me and finding a wired connnection as the hardest parts for techs setting up a new machines or migrating an existing one, and my moving parts are all automatible if not already automatic. As I said I made an applescript-and cocoaDialog GUI app for techs to run after they've finished the upgrade to Yosemite to convert local home folders to an AD account and rotate the filevault key to make sure it's escrowed. So the doctors, teachers, and other IT pros see a fast, efficient process with almost all of settings taken care of.



And maintenance-wise, AutoDMG is very versatile and lightyears easier than all the modular image creation tools that came before it, and it gets updated regularly as Apple releases patches and new whole installers. DeployStudio has kept pace with fusion drives and the NVMe interface that allows the metalBook to get a super fast bus without PCIe... which certain inventory systems are still having issues recognizing…*ahem* DeployStudio is well understood and has been around since before Mike Bombich's NetRestore packed it in, which some of us are old enough to fondly remember. Since munki is laying down the apps, only the most recent stable versions are present at imaging or ingest time, which autopkg helps make a low-maintenance prospect.

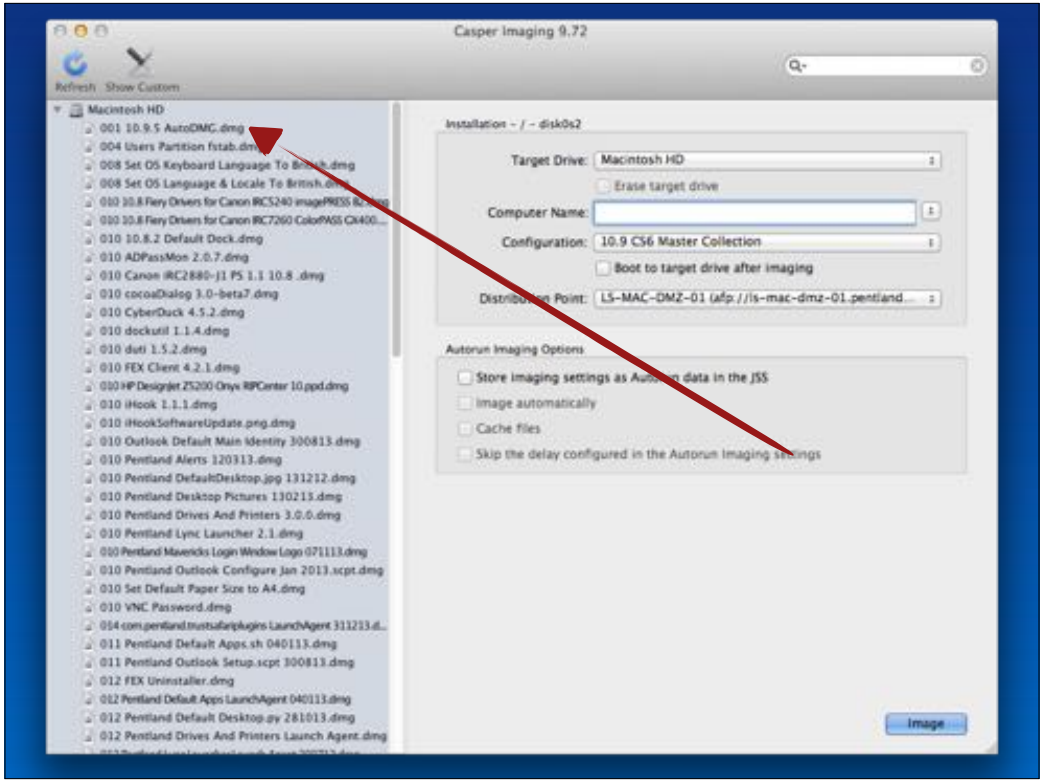
(Ben:)So let's start at the beginning with the OS. The Casper Suite has one app that alone that can be compared with Marmite in that you either "Love it or Hate it"

"Composer" can be used to create a deployable OS.dmg for monolithic imaging (which you shouldn't do.. except for when you should) or capturing an OS of a never booted Mac via target disk mode, this is handy when it comes to forked builds.

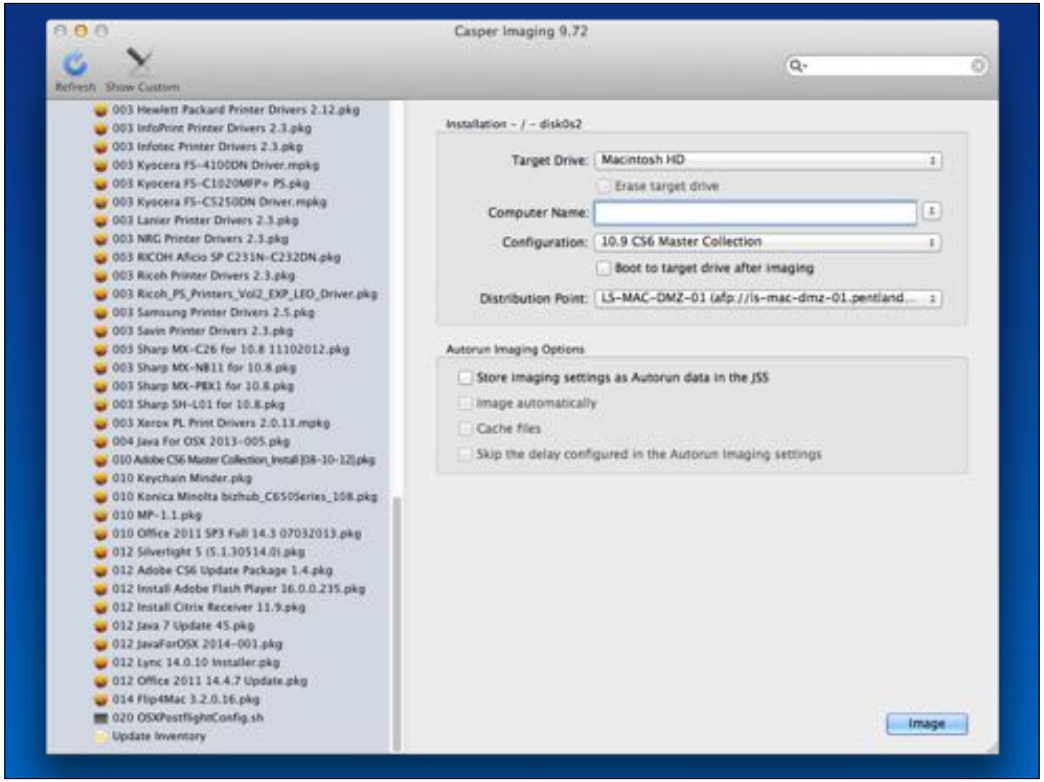
Since the days of 10.5 "Casper Admin" has taken OS installers directly, so no faffing around creating OS.dmg or you own installer packages.

These can take some time to deploy, so you may want to create an OS.dmg via Casper Admin by compiling a configuration containing an OS installer. As per AutoDMG.

Now we have the OS, you can chuck in the rest of the build, create a configuration.. & deliver this via Casper Imaging which can be run from a NetInstall or NetBoot image.. or even via Target disk mode imaging.



If a new OS comes out? Swap out the OS.dmg with one for the new OS (minor or major) & test your new image.



Once all DMG/PKGs are cached the Mac installs those DMG/PKG's in order & then runs a script that triggers a policy to do things like AD binding etc.

No-imaging? Login as admin on the mac… don't erase the HD & don't deploy an OS..

Whichever method you load up casper imaging, once you're authenticated to you JSS it will use the network segments set to point you to the relevant/local distribution point to pull the items to be used during imaging.. So it's easily reproducible & scalable…

In fact i've taught people to image for a number of years now by telling them the following

In fact i've taught people to image for a number of years now by telling them the following.

All in all, a spinning disk 2011 13" MBP on a 100 MB connection will be fully imaged within 90 minutes with OS, Office 2011, CS6 & other stuff.. but the steps take the tech 5 minutes in total.

The end result is a Mac that any AD user can login to & have their apps on there apps ready & where set auto-configure for them.


Simples!

Alternatively, if you have SSH access to your Macs & then are not enrolled onto the JSS, Recon can be used to enrol devices on your network remotely (skipping existing).

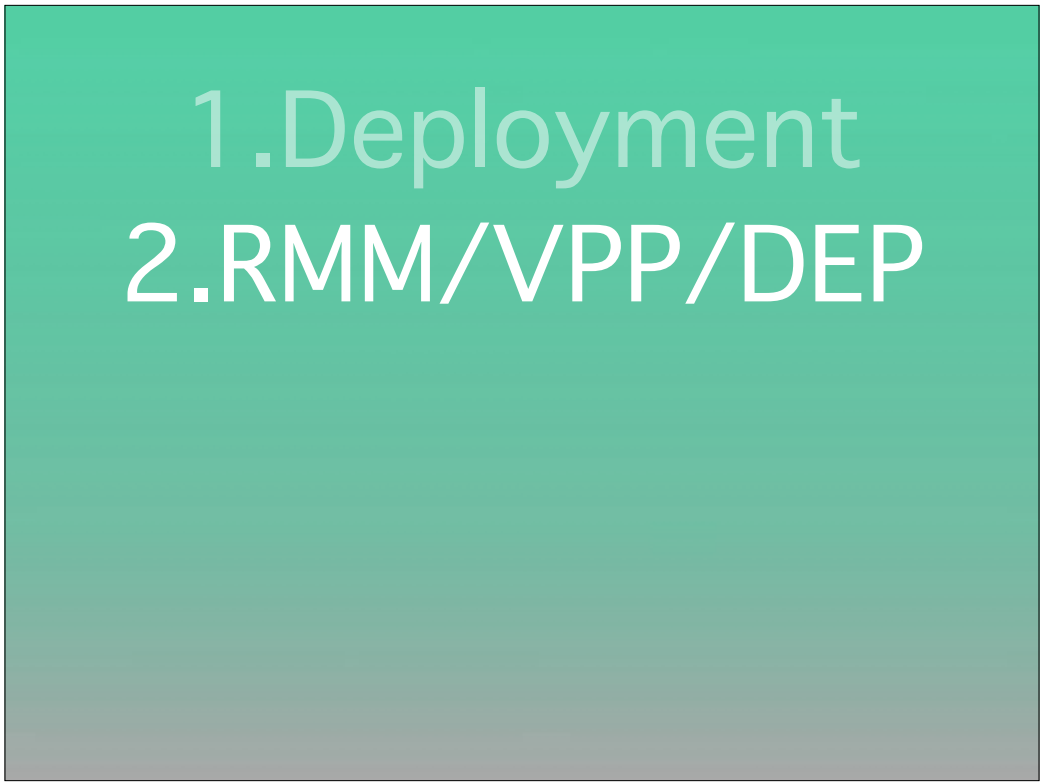
This can be handy when coming into a new environment.

Lastly you can enrol your org & JSS into DEP then leverage DEP (if available) & again post enrolment deliver software & settings etc..

This is possible as the JSS is an MDM.. or if DEP is not an option & you're in the "imaging is dead" camp.. Boot up your shiny new Mac & login as admin, open Safari & enrol the Mac into the JSS.

Recon, DEP & manual enrolment can all leverage the post enrolment trigger, with which we can deliver software, settings etc.. Notify the users that they have been enroled &/or open Self Service.. (if delivering).

So there are a few options depending on the deployment method wanted.



Here's where we get into some real challenges with playing in Apple's toolbox, since before Pepijn Bruienne cracked the code on getting his own MDM running in a recent post on his enterprisemac.bruienne.com blog, it seemed that Open Source was out of the picture for things like VPP and DEP.

As there are still enough issues for developers that stop them from solely distributing through the app store this hasn't been as much of an issue for folks without an MDM, and enough have found the DEP workflow incompatible enough with how thy work that it may just be a non-starter and not a concern, but we're going to concede a lack of open source tools for those purposes.

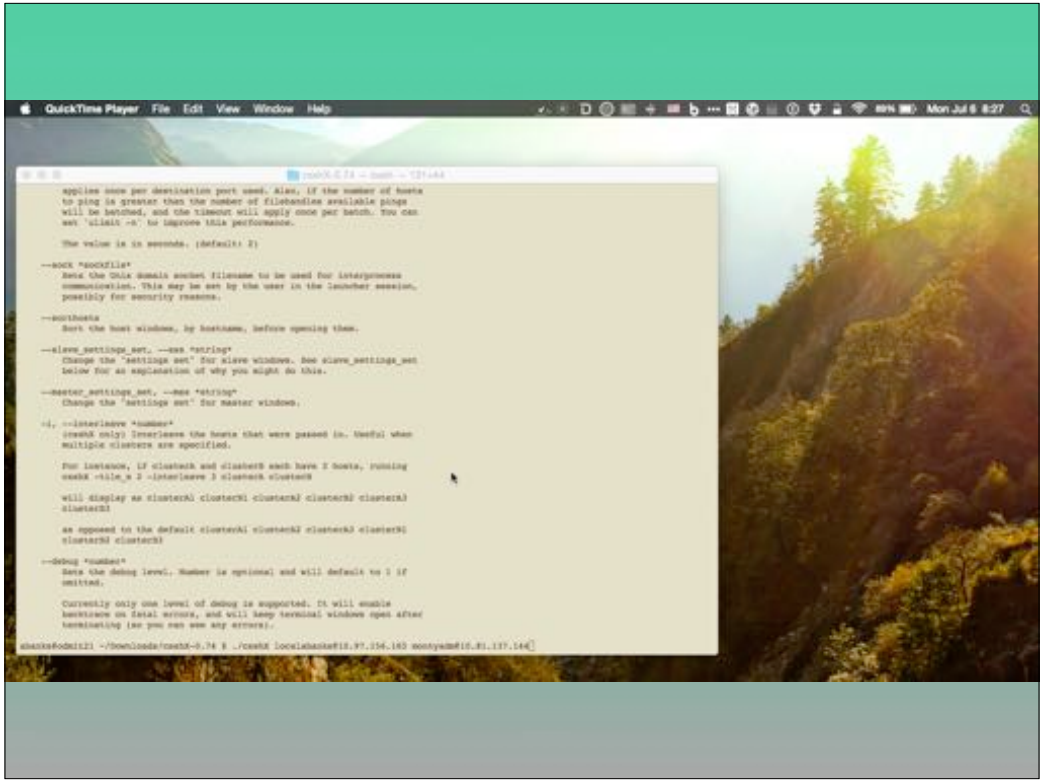
Another challenge is being able to provide assistance to machines over the WAN, there just aren't too many great solutions - if they're free, like Google Hangout's screensharing, you're usually still paying or making money for Google in some way.

Unless you're Richard Stallman, and a bit nutty, you're probably not into being a hard-line FOSS person hosting email on top of your other tasks, at least I hope so for your families and weekends sake.

I'm just going to focus on getting tooling for Remote Monitoring and Mgmt, which is to say reaching out and touching your fleet of machines and pulling back inventory data in some cases. Also theres something I refer to as Orchestration, which you need when guiding a bunch of machines through a change

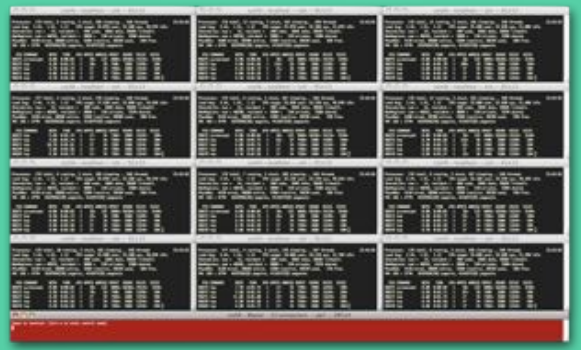
and Apple Remote Desktop is just not going to cut the mustard.
Let's start with an easy scenario and get more complicated. It's a pretty basic config to turn on SSH and VNC access to machines, DeployStudio can do it for you among others,

and installing a suite of clients meant to work together like puppet and it's spec-gathering tool facter is similarly straightforward. So for remote assistance and orchestration there's not much to it client-side, and puppetlabs makes many of their server tools for storing its specific information in a database, but I'm just going to highlight two tools, clusterSSH for OS X and Puppetlabs Marrionette Collective, AKA Mcollective or mco.



(csshX demo)

csshX:



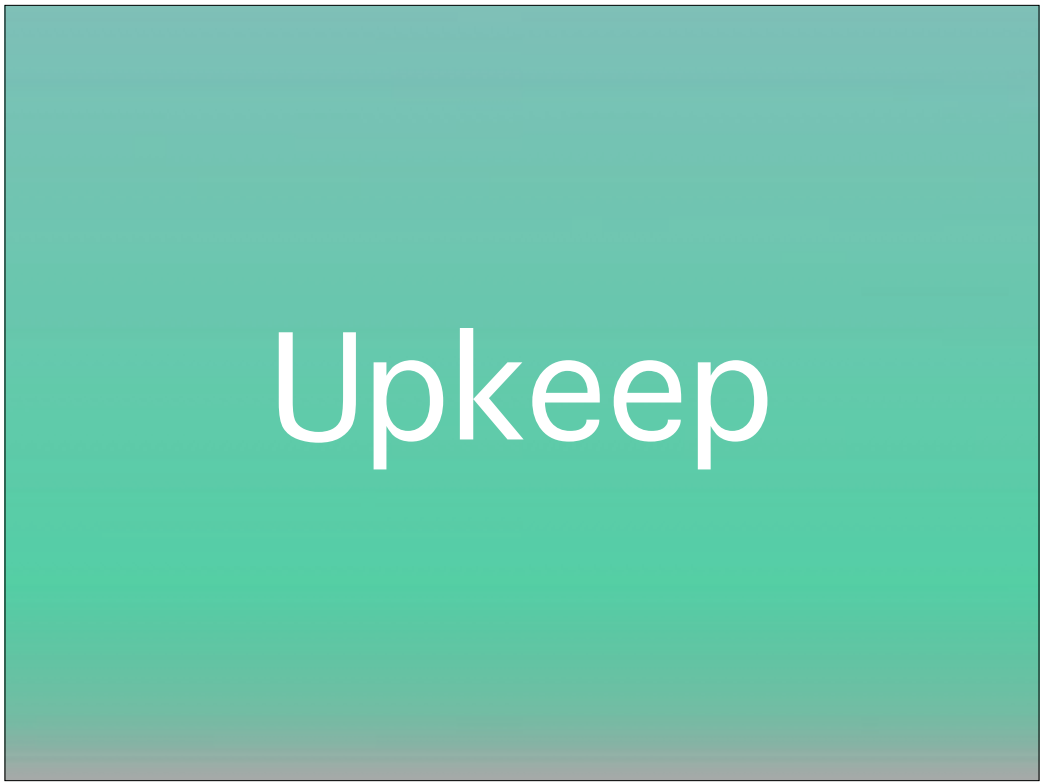
‘And one TTY to bind them...’

csshX is super cool, that it's this perl script that opens many sessions that you can start to do all sorts of relatively low-tech fun stuff with. In one instance I was performing command line copies from the remote sessions _to_ my admin workstation, and then had them apply what I had sent - great fun and quick and easy to configure for those times that Apple Remote Desktop is refusing to believe the screen of a machine you just controlled is online for things like Send Unix Task.

mco is a bit of a bear to set up as it has many non-trivial infrastructure dependencies, ActiveMQ being the server-side message bus, but it's super efficient over the WAN to say, make all of the online machines you have configured, check in and get an updated config. That's not to say many people are actually even actively using it in the real world, though, as on top of the complexity to set up and maintain, many folks say they'd just rather trust the schedule their suite of client config tools check in on for a urgent change. And between munki and puppet's launchdaemons maybe you're covered.

A simple service like csshX allows you real parallel operation that's as reliable and well-understood as any single ssh session. You get the full power of the command line and the consistency of on-demand results. MCollective has a rich way of stopping the 'stampeding herd' affect by setting a random delay consistently to even out the load on CPU or network resources when you're aiming it at heavier-duty tasks. It was also built for lossy, slow internet connections which makes it optimal to use over the WAN. And both of these orchestration tools allow you think about your fleet as a whole and chop them up in chunks to operate on a bit at a time.

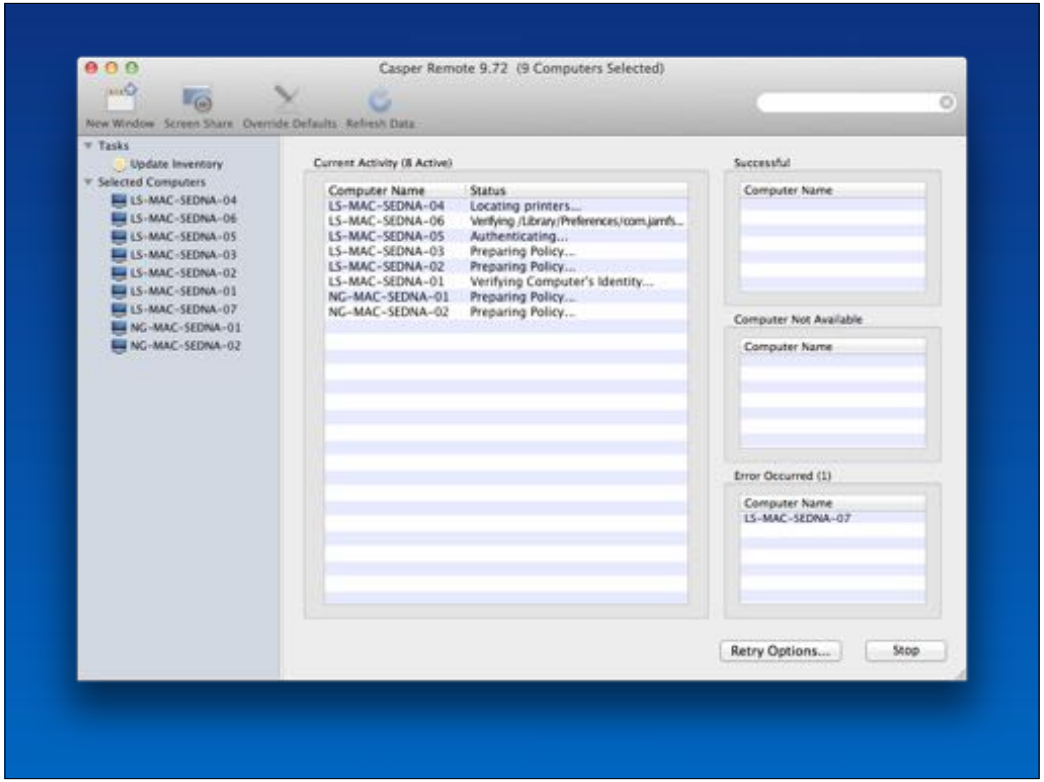
And going back to DEP and VPP, there is the principle of ‘You Ain’t Gonna Need It’ in software development to help you prioritize what your fleet needs. Apple has been kind enough to not validate receipts and use the honor system for MAS-only apps like iPhoto, and autopkg can push MAS apps into munki to distribute, so maybe the VPP thing isn’t as important at a smaller scale. Or as Joe Chilcote says “it gets easier the less you care about EULA’s”.
DEP hasn’t really sold as many admins because people don’t leave their macs in a cab after a rowdy night out, but I’m going to admit that for free and open source, after Meraki no longer is a purely free MDM, there aren’t good substitutes for a paid plan. I’m just not universally sold on the value of DEP for Macs. MDM, sure, but plenty of larger-scale places get along without it, as will we with per-employee licensing for MaaS 360.
I’ll also concede that you probably want a remote assistance tool for frontline techs like Bomgar or GoToMyPC in order to be able to look over a customers shoulder from afar, but I do hope having your mgmt tools like Munki and Puppet facing the WAN means you can possibly address a customers issue even if they’re not accessible over VNC.



As maintenance and upkeep goes, there is no more breakage to a tool like csshX than customers being offline when you want to run it, but hey, some would say Apple Remote Desktop has a non-viable solution for that and even performs less consistently when clients ARE online. MCollective has had a reputation of sometimes having false starts and being so powerful you can easily DDOS your own resources if you're not careful. And Puppetlabs has deprecated mcollective and will be replacing it with a new tool that I certainly have an eye out for. But you have a real roadmap with MCollective so I encourage folks who are a bit more fearless about setup to investigate.

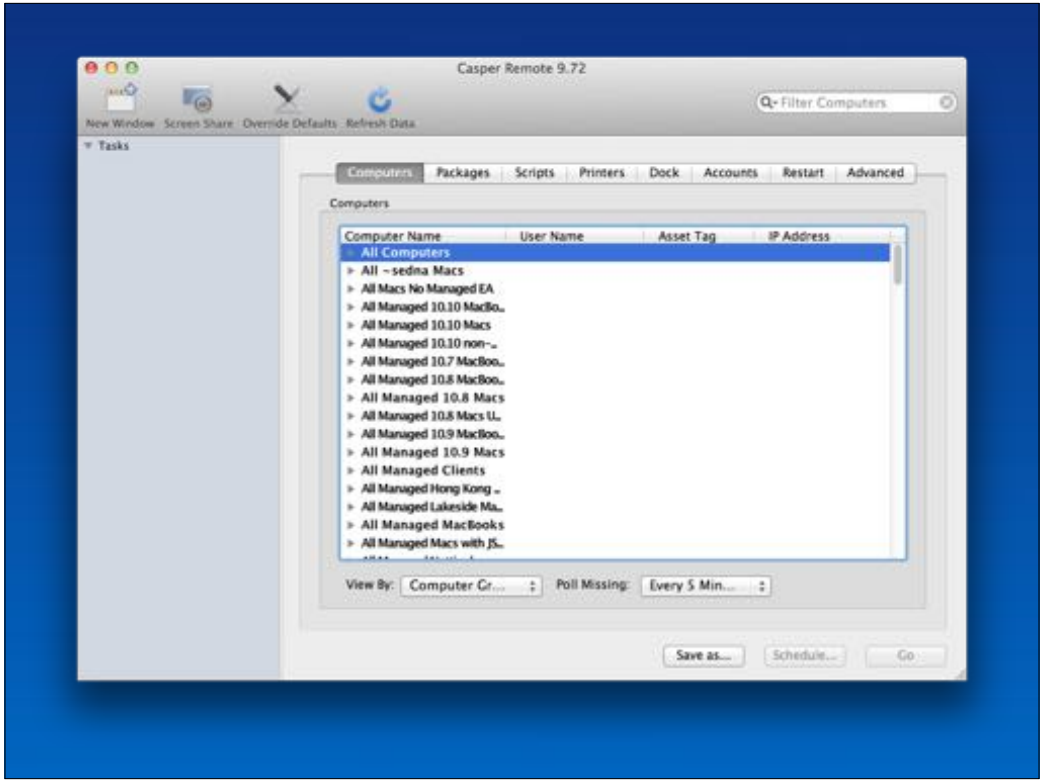
(Ben:)Let's talk about the aptly named "Casper Remote"

"Casper Remote" grabs the mac clients information from the JSS, so no faffing around scanning network ranges for devices.



When an activity is chosen, the mac running Casper Remote initiates an SSH connection with those targeted Macs.

This is using the management account details held in the JSS. So this works when rotating the management accounts password via the JSS.



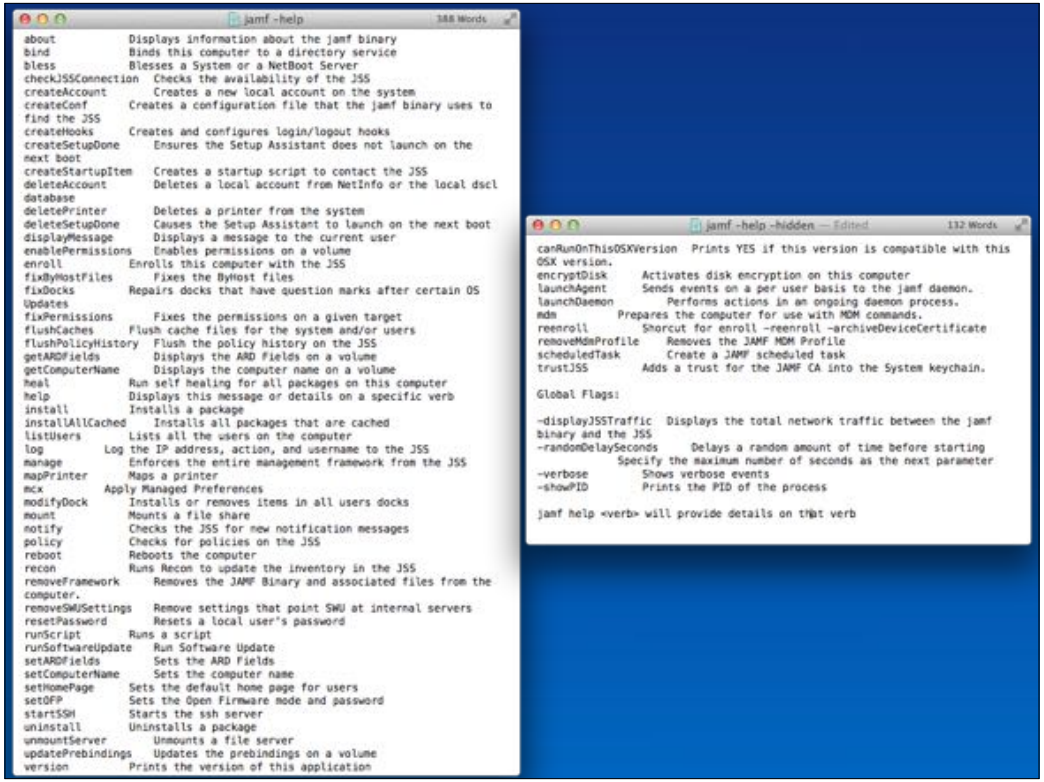
Various activities can be run from “Casper Remote”, including Screen Sharing, installing of Packages, printers or running scripts held within the JSS, creating accounts, binding to AD, setting EFI passwords, flushing caches amongst others…

“Unix” commands can also be run (for those ARD users amongst you)..

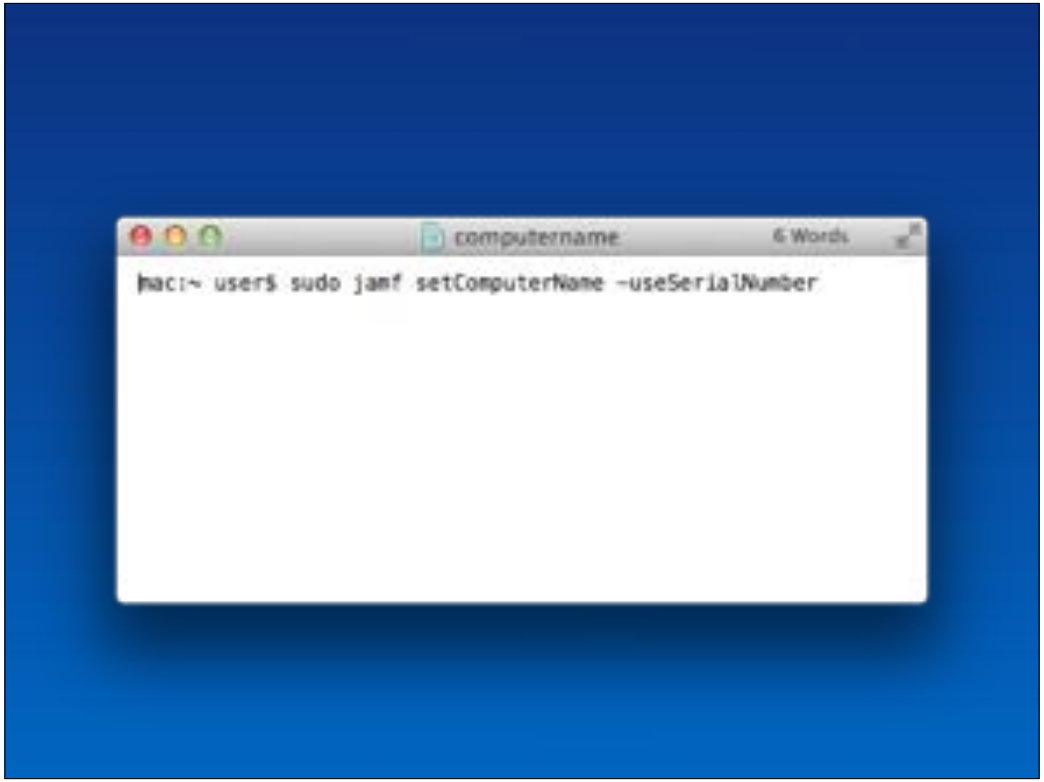
& as you can see, smart groups created from within the JSS are shown within “Casper Remote”

This is all helped by the JAMF binary. When a Mac is enrolled onto a JSS the JAMF binary is installed.
This binary will contact the JSS every 15 minutes as per the default check in trigger (but randomised up to 5 minutes either side to lessen load)
When they do they will update a few items.. such as their IP on the JSS, which is then where apps such as Casper Remote get the Macs details from.

When running activities on a target mac, essentially commands are run over SSH that invoke the JAMF Binary.



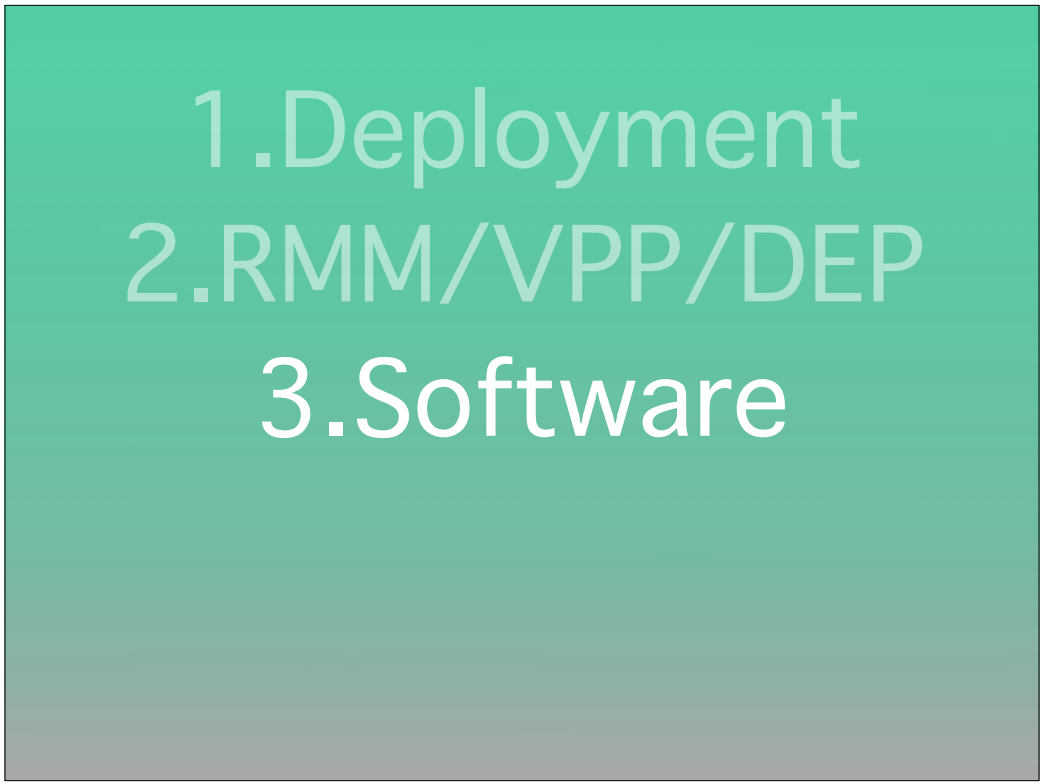
There are a number of commands the binary accepts with the standard & hidden help documentation showing.. but each command has its own help documentation too.



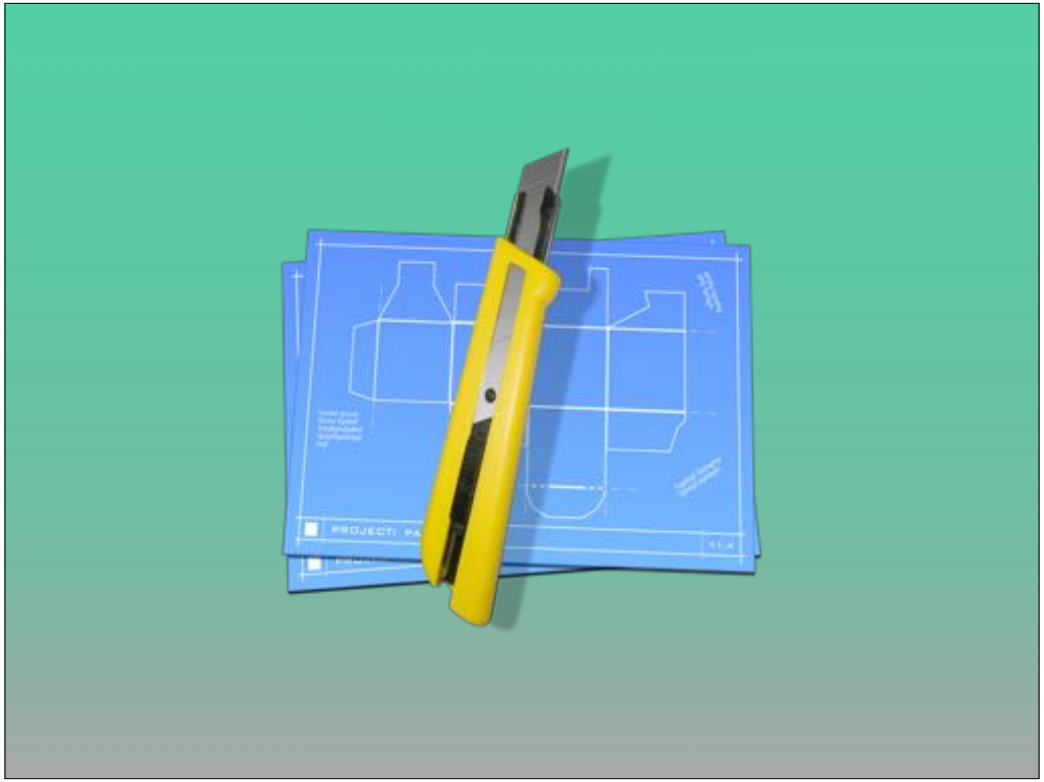
This can lead to commands like the above being run, which some would argue you should shy away from as they are not “proper”

I paid for it so I'll use it.. seems silly not too

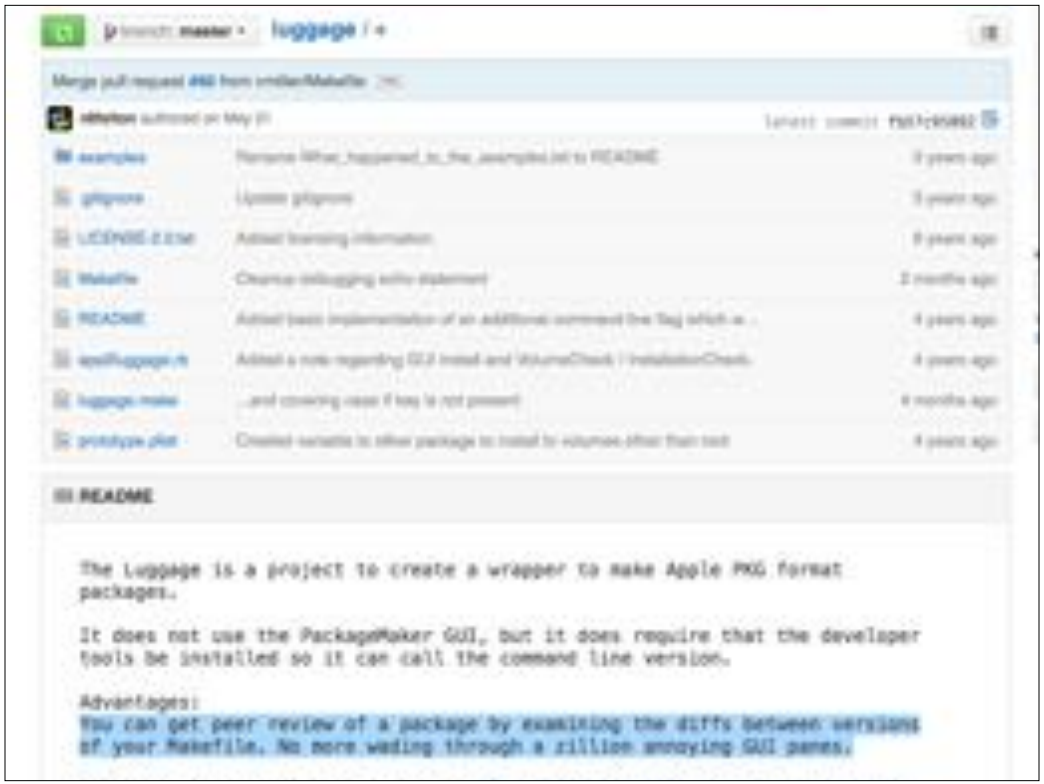
The JSS is an MDM, as for VPP (1&2) & DEP.. so it's all there.. (if you can use it)



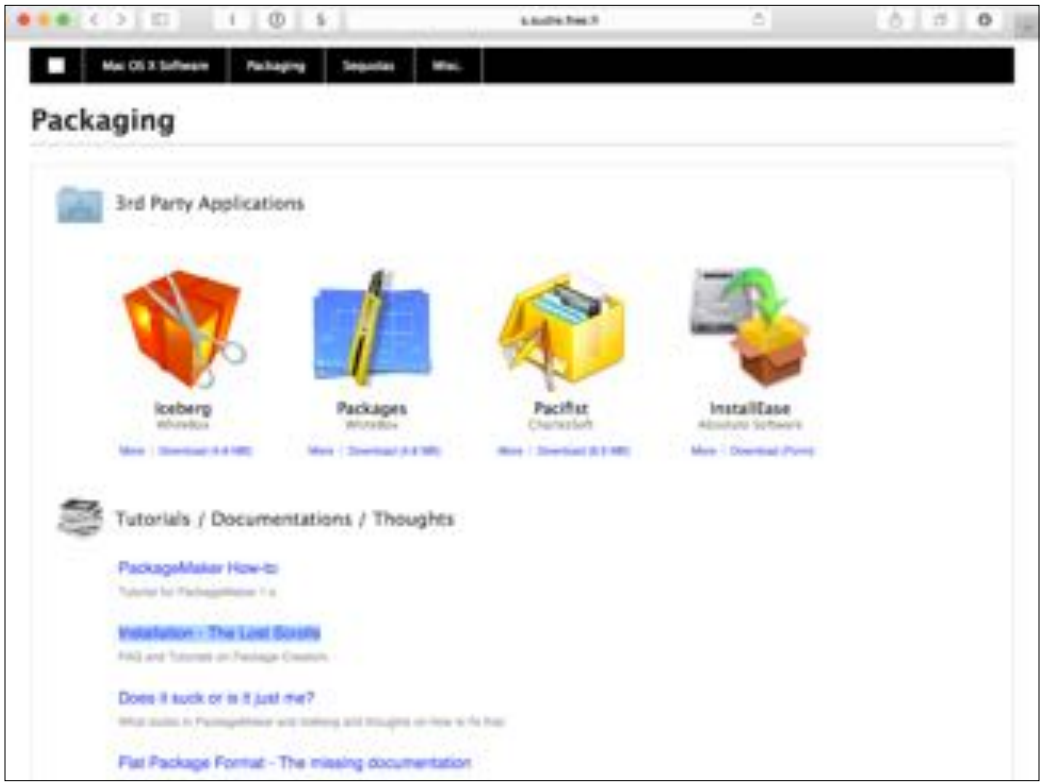
Now it's my turn to come out, guns blazing, and I intend to absolutely obliterate the competition here.



For packaging it's a bit of a mixed bag, as the free solution Stephan Sudre's Packages is a great closed source entry for when you need a pretty GUI, both on the creation end and the display end for customers



I'm personally really in to theLuggage, for the fact it operates on text files that can be backed up and is therefore shareable and revision controllable. Which helps if you've ever come back to a project months later and can't remember what it was you did that made it work. I'll admit there's a learning curve for both this and Packages



However, between the luggage-dev mailing list, the linkfather's posts on packaging, MacTech magazine articles and managingosx wordpress posts by Greg Neagle the bagel on pkgbuild, and Stephane's older but really comprehensive missing manual for flat packages (and what a name, right? The Lost Scrolls), the community really has you covered wherever you want to begin.

And Greg implored you to check out autopkg last year, if you haven't taken a look yet I don't know what you're waiting for. I'm lucky enough to have some of my recipes used by more than a few people, which puts me among 50+ other authors with the distinction of being participants in the autopkg organization as Github refers to it, so I get to use the word 'we' about it's development, and when I say we play nice - even though it was originally conceived to automatically prep for the best patch mgmt solution Munki, it has been extended to support paid mgmt systems.

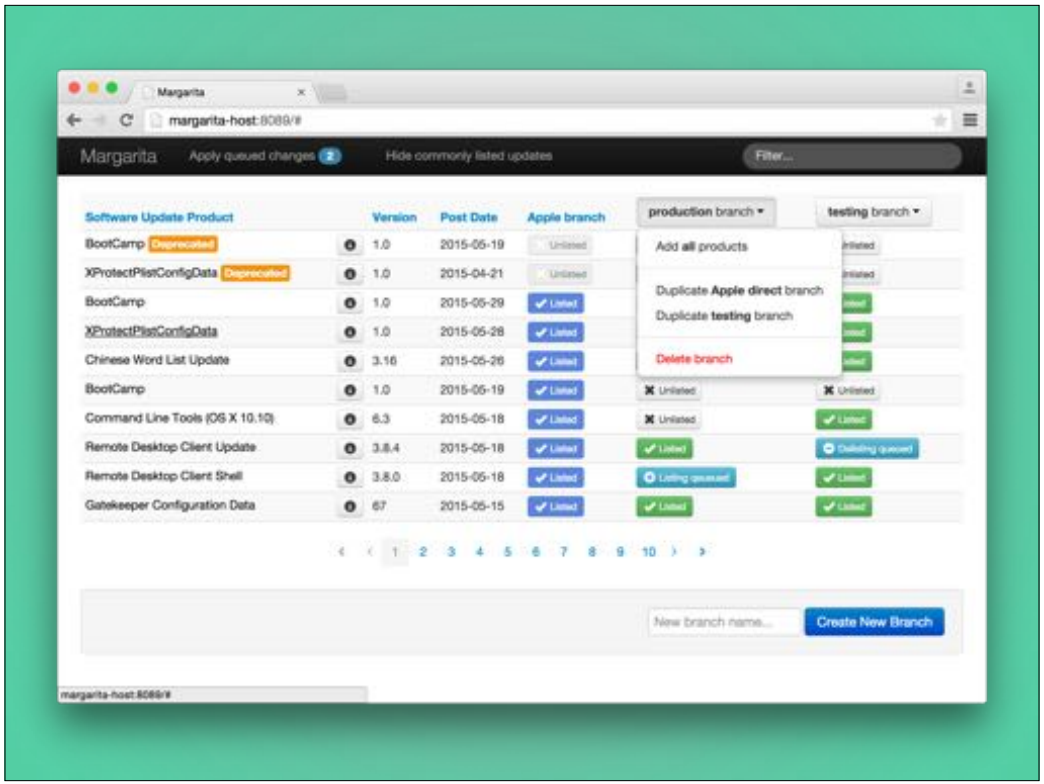
With the efforts of Shea Craig there's yet another piece of free, continuously developed and robust software which makes the on-ramp to getting software into the JSS almost as easy. Don't think too deeply about the image in this slide, I can't really explain. Anyway. I talked about the luggage and WhiteBox Packages, but AutoPKG helps assure I need to reach for those tools very infrequently, as it can fetch a software product or update and prepare it for your deployment system, with folks like AnthonyReimer even hooking it into DeployStudio. So much has been said about autopkg I'm going to leave it at that.

And really, so much has been said about munki that I'm not going to be covering it much either… to paraphrase the famous words of Mike Tyson, its defense is impregnable

I've even got a years head start talking about Apple Software updates, here's a link to my talk on Reposado as a software update service from last year, but as a further pre-emptive strike against Ben…

yes, even JAMF does something in the way of contributing to open source, as evidenced years back with their first foray on Github, which does contain reposado and may contain nuts. Spoiler alert: they aren't the best stewards of their own project

They're missing an update from earlier this year which would really help folks who've disabled automatic checking of Apple updates, as it says towards the beginning of this post on Tim Sutton's MacOps.ca blog, reposado was updated earlier this year to add tools to manage when xprotect updates (among others) get offered to clients.



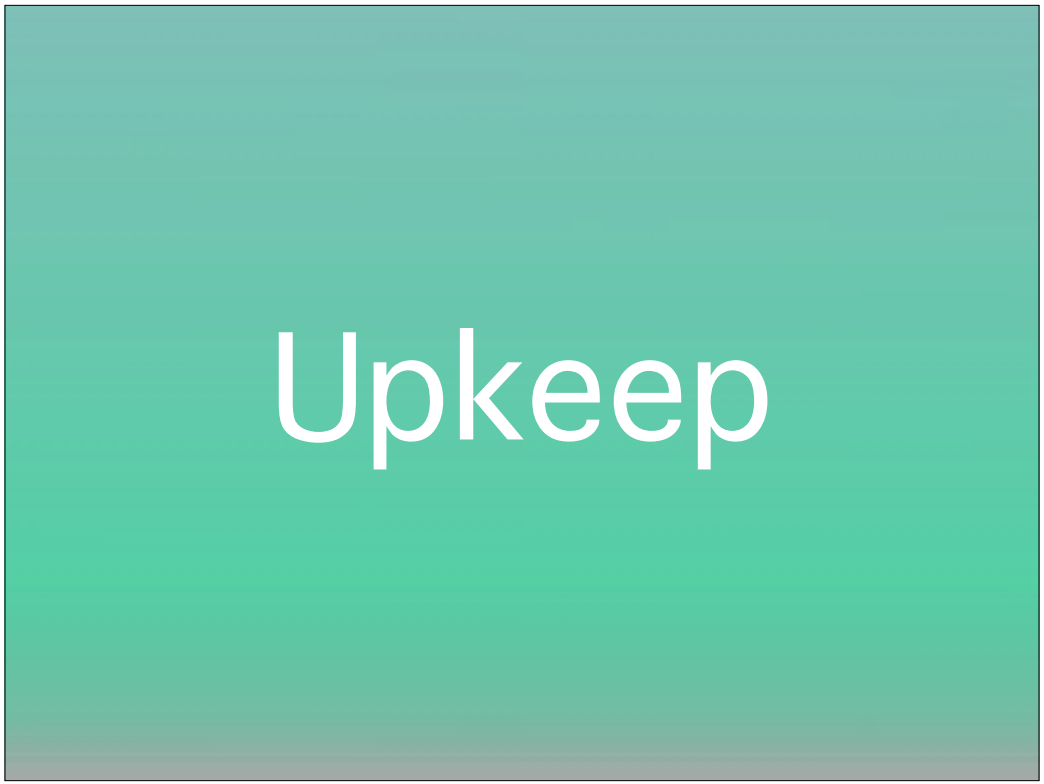
Something that I'm happier to help add to Margarita, since it's in python, rather than the PHP of NetSUS. We have the option of web gui's too..

Or objective-C GUI's that really help folks understand and interact with munki…

And just to highlight one lesser-known tool in this space, we load Apple Update information into munki with SUS Inspector, for things like enforcing OS and security updates with the munki client gently prodding folks before unceremoniously kicking them off.

I see so much ungainly jiggery-pokery with how admins of other systems need to notify customers, do arms deals just to get them to log out, and a fair amount of custom dev anyway because the brittle mechanisms some of these mgmt suites use is not optimal for the most common tasks we'll perform around software. Not that vendors make it easier. There are so many design decisions in munki that make the agony of Adobe products a thing of the past, and we can efficiently get these products and updates to them with phased rollouts so they're always getting well-vetted software.

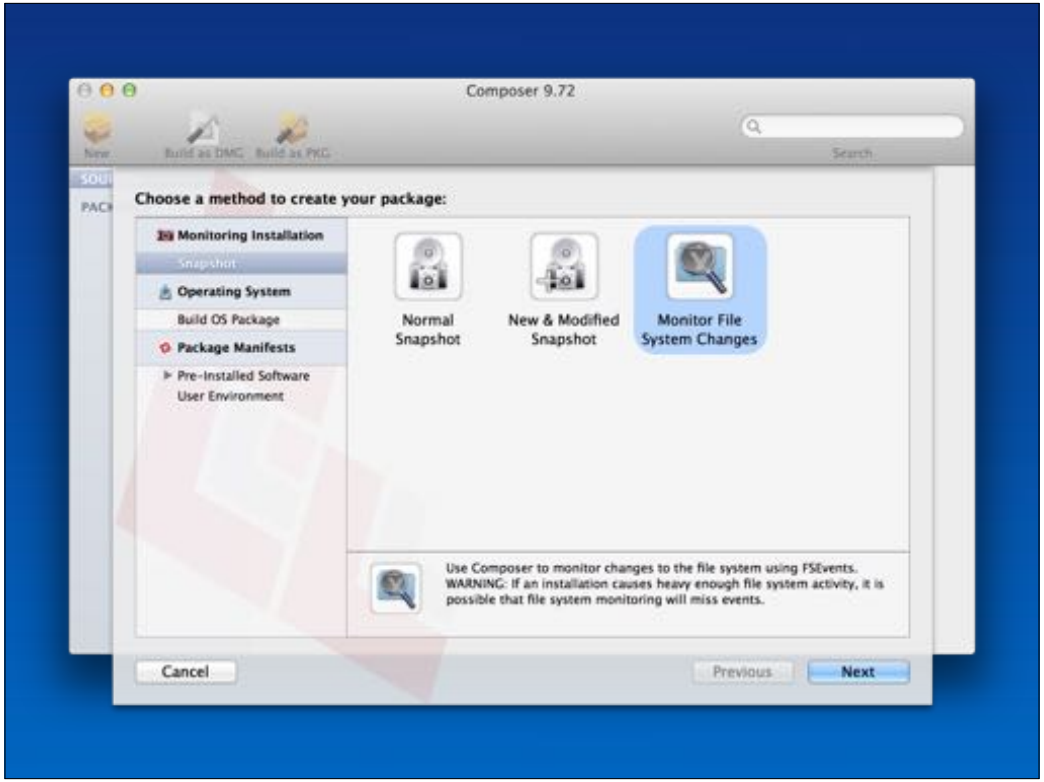
Brand X just doesn't build that into the design. You load that software in with a quick trip to autopkg, or roll up a new pkg with theLuggage in a way you can immediately see the bill of materials, and turnaround time to your customers is way more expedient than tappa-tappa-tappa in certain *ahem* less optimal packaging apps or PHP-tacular interfaces.



Autopkg taking care of itself is great, reposado being able to hook up to a notification service like I recently blogged about on AFP548 means you feel in control and on top of things. The only thing you kindof want to take responsibility for is pushing things from a testing group to production, and all these things being open source means I could, for example, put out a proof of concept script to automatically promote the products I'm less concerned about after a time period. In yet another effort to replace myself with a _tiny script_

(Ben:)

The common misconception is that Composer is just for snapshotting.

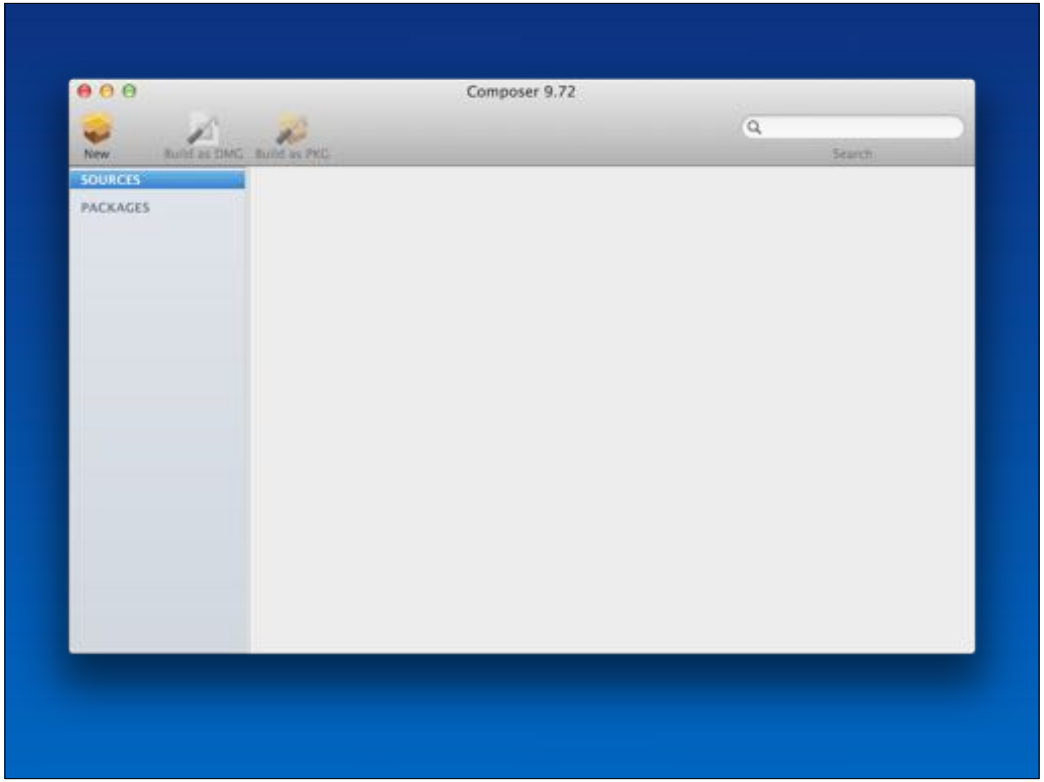


This has probably arisen from the fact that on 1st launch you'll see the above.

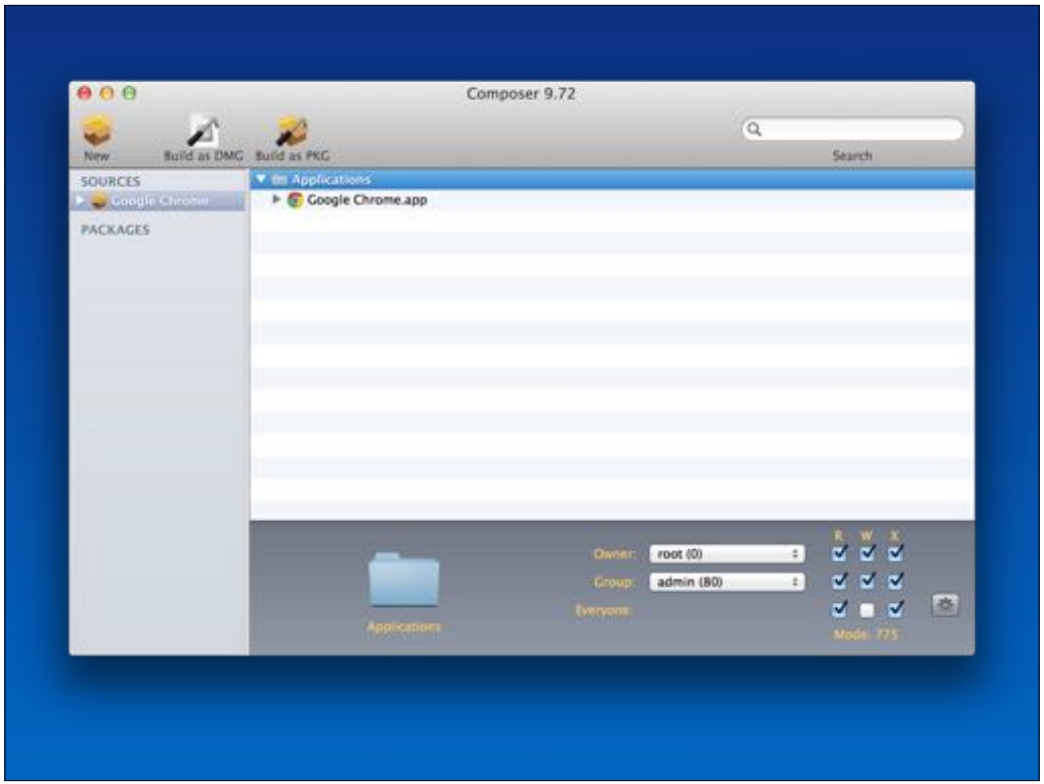
Yes, it can be used for snapshotting.. which has it's uses when troubleshooting, such as creating a "snapshot" that leverages FSEVENTER by monitoring files system changes (shown).. The files captured by Composer can then be used to A/B against originals etc..

Please don't use this to build any packages.. just for testing or spelunking.

But click "Cancel"



And you can make a PKG or DMG from anything dragged into the “Source” part of the sidebar.

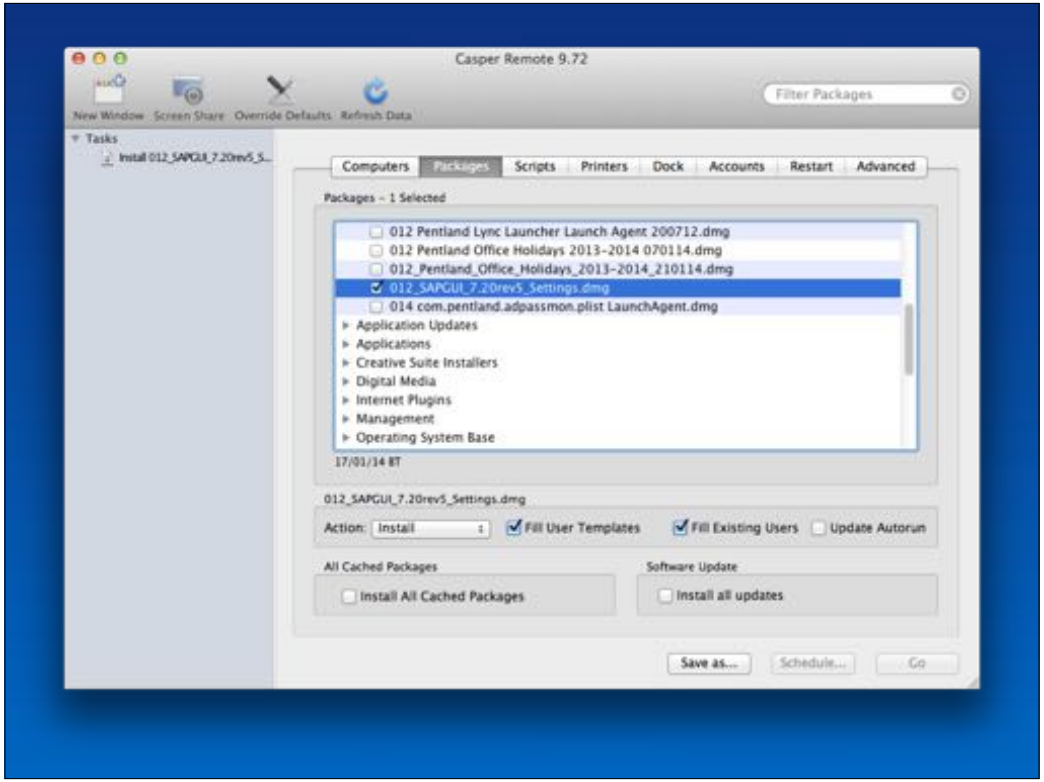


Once an item has been dragged, “Composer” copies it to it’s temp location & will create a DMG or PKG with the item being at it the path from where is was dragged.. As shown.

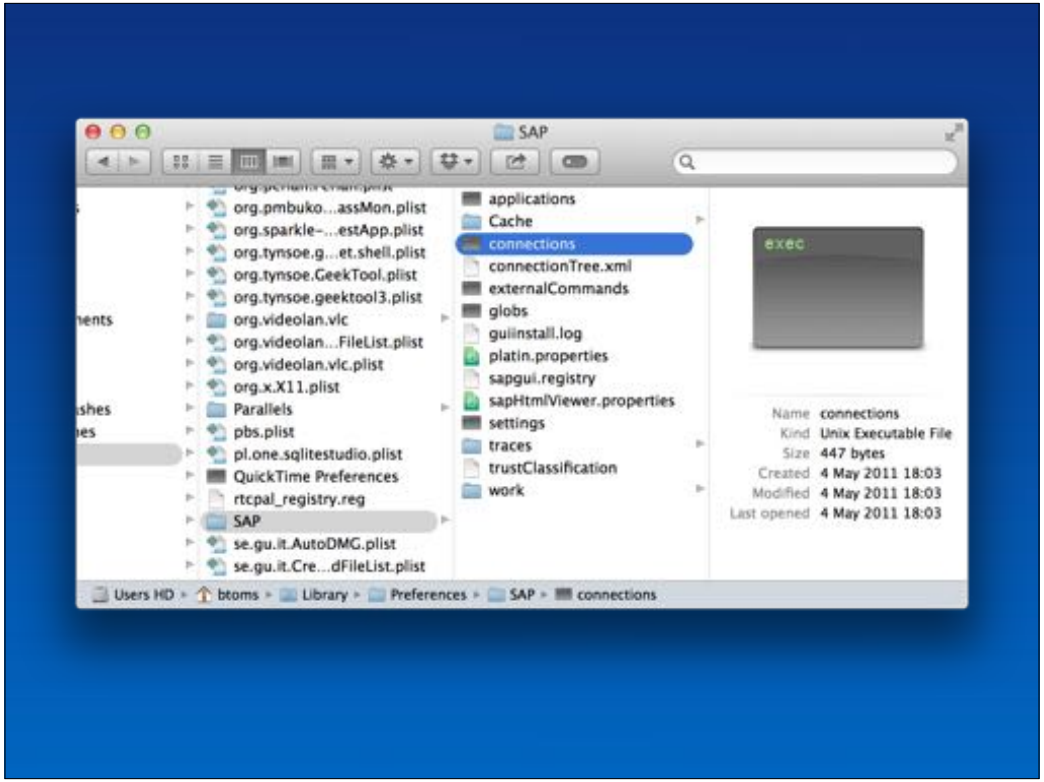
If a DMG is deployed, it’s contents are ASR’d or similar to the target Mac.. so uploading a “Chrome.dmg” from Google without repackaging the app to Casper Admin will have the “Google Chrome.app” path as being the root of the DMG & so will deploy the app to the root of the target Macs HD.

Which is rarely desirable.

Now this doesn't mean that everything needs to be packaged, & many PKG's can just be dropped directly into Casper Admin.



DMG's created in Composer can also be deployed via the suite & leverage the "Fill User Templates" & "Fill Existing Users" feature.



This will copy an item in a DMG to the same location across either all existing users &/or to all user templates.

Such as the above.

This is handy for apps like SAPGUI where the settings file is a UNIX executable as show & needs to live in the users library.

I know this is frowned upon by some.. but i've used this for years with..


NO RAGRETS

So far.. so OK… right?

I have & still get by with the JSS's method of uploading a DMG/PKG to install.. creating a smart group of Macs to install the software too, then creating a policy to install the DMG/PKG to those Macs in the policy scope.

We cache the updates locally, including one's apple updates & notify the user & then install at logout so there are no conflicting processes & we're in the user space.

BUT.. this is a workflow I've concocted.. whereas

There is a better tool out there for “patch management”.. as much as Greg hates that term.

JAMF are aware & have promised to address things..

In the mean time (& breaking my promise at the beginning of this talk).. I'd like to point you to AutoPKGr with it's JSS Importer which started as a gun to head project for Allister.

Auto Update Magic

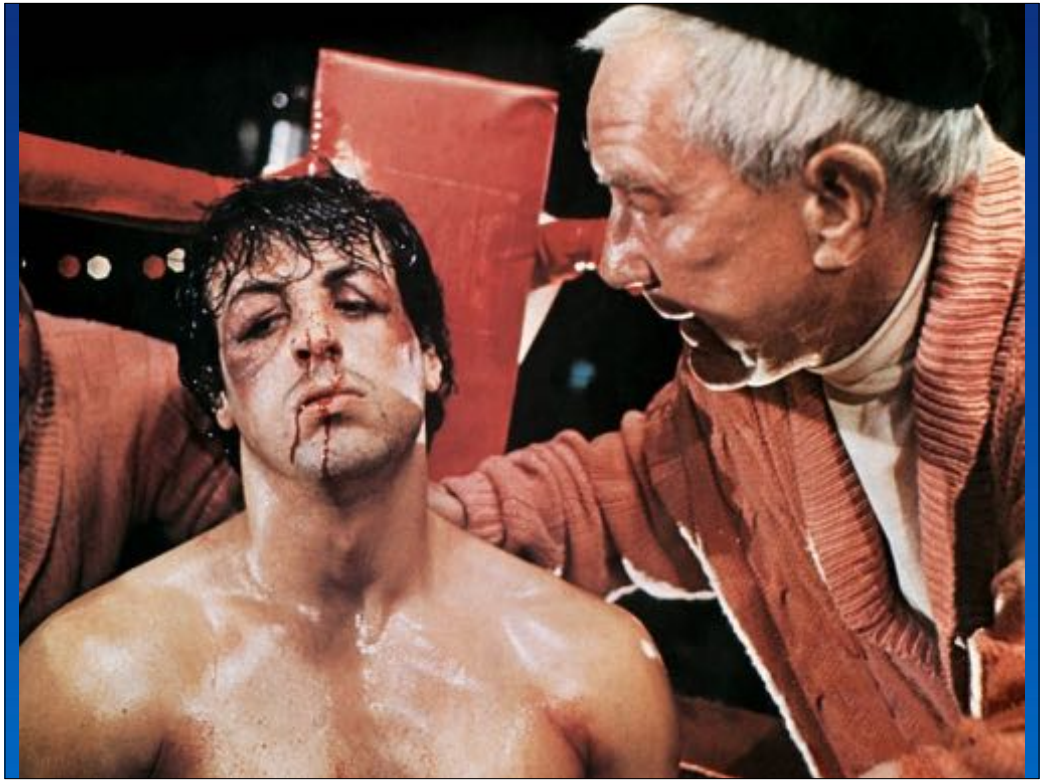
http://bit.ly/1HAR0rJ

Patchoo!

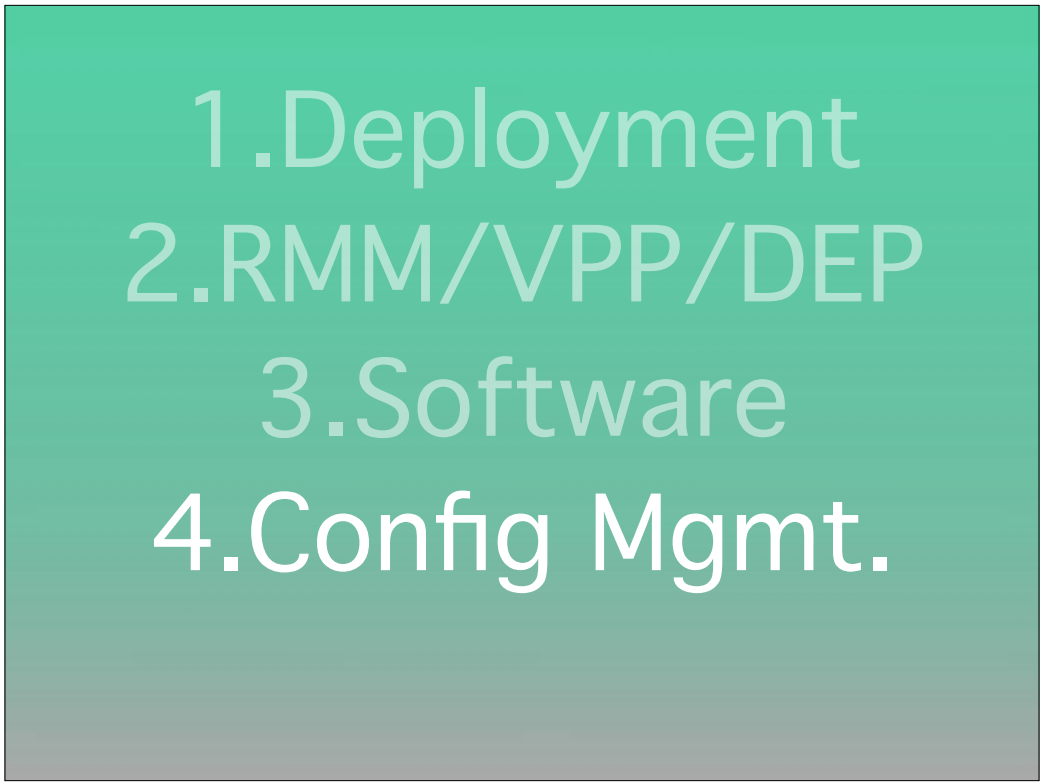
http://bit.ly/1lyq64K

Once you have the AutoPKGr running.. then you'll need to deploy it..

Both of the shown are great workflows that attempt to bridge the gap between the Casper Suite & functionality offered via Munki.



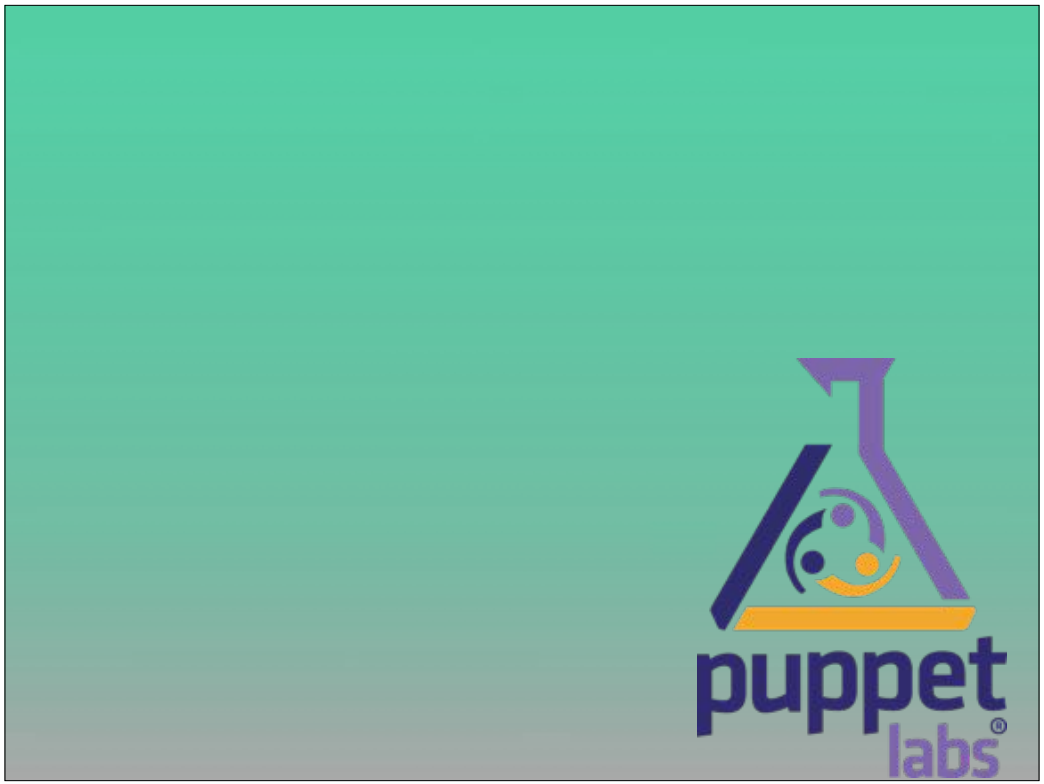
Ok… after this round it feels like i'm back in NYC..



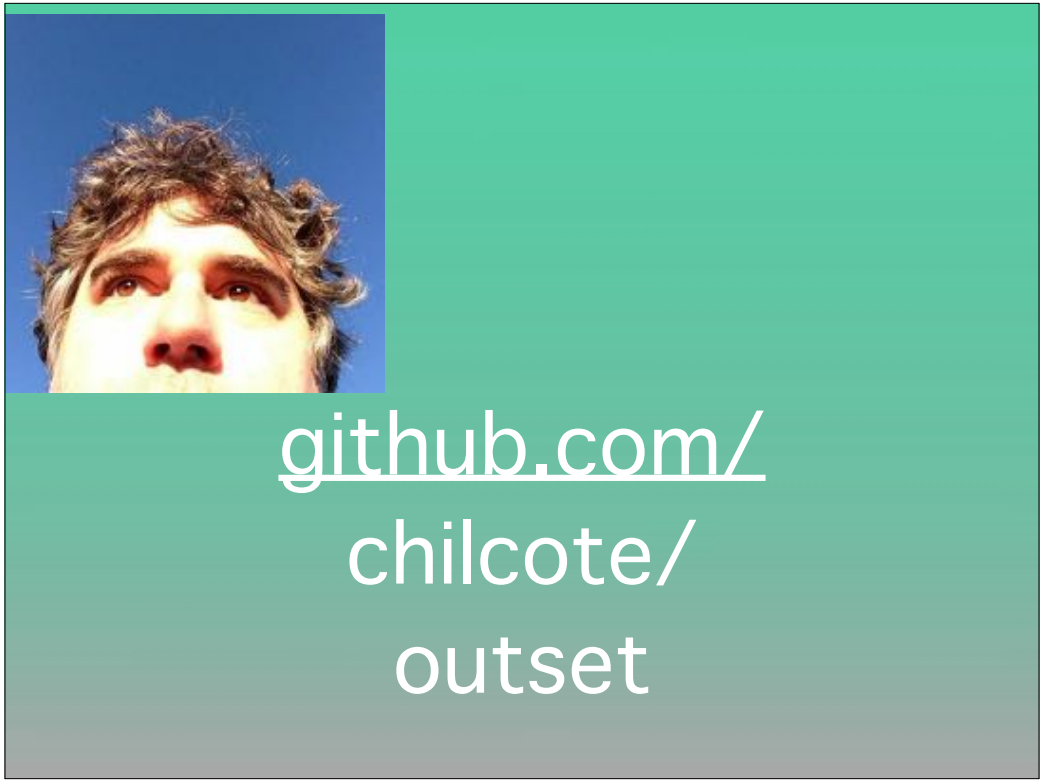
The full-blown MDM Ben touted was supposed to have taken this next round regarding configuration mgmt as well, but in comparison to that singular type of tool that for the most part plays by Apple's rules with a few exceptions of older, poorly-implemented, and undermaintained half-solutions,

lots of folks can fill out the product offering and decide they're going to contribute a solution to the entire problem set of cfg mgmt. I mentioned puppet when talking about mcollective, other Open source tools like Chef were the original innovators in this space, they can manage config profiles and puppet even had extensive support for etc/auth modifications,

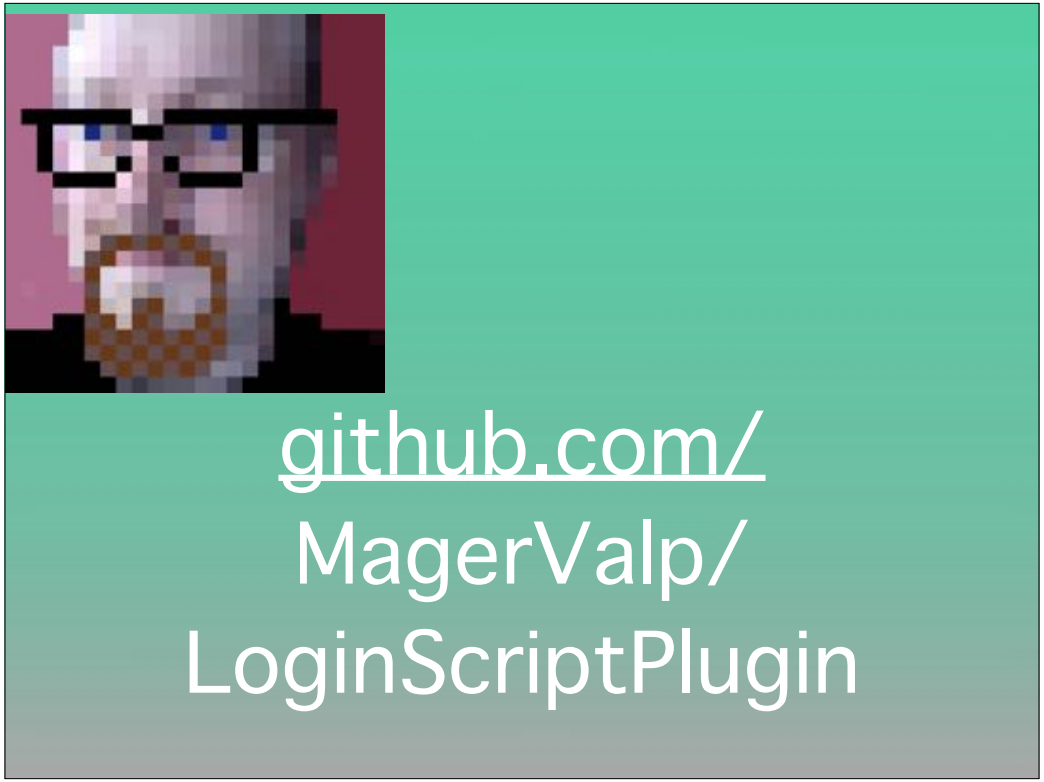
and for per-user or other periodic use cases two stand-out tools from awesome folks in the Mac community are Joseph Chilcote with his awesome outset project, and PerOloffson(aka MagerValp) with his more industrial strength and slightly more complex loginScriptPlugin.



We went with Puppet because we have a lot of potential growth, and it helps us build our other infrastructure with devops practices, with the side-benefit of therefore being buzzword compliant.



Outset can install pkgs or run scripts for you at an agreed upon time, with the further flexibility that it can happen repeatedly like every boot or just once the next time any user logs in. Yes, you need to know what you want to do and how to achieve it, but from that point on outset can take over. This is perfect for bootstrapping an imaged computer, or per-user. Whereas naïve solutions like something I heard was in use somewhere called "FillUserTemplate"(I feel dirty just saying it), would actually fall short when you're for example using network accounts.

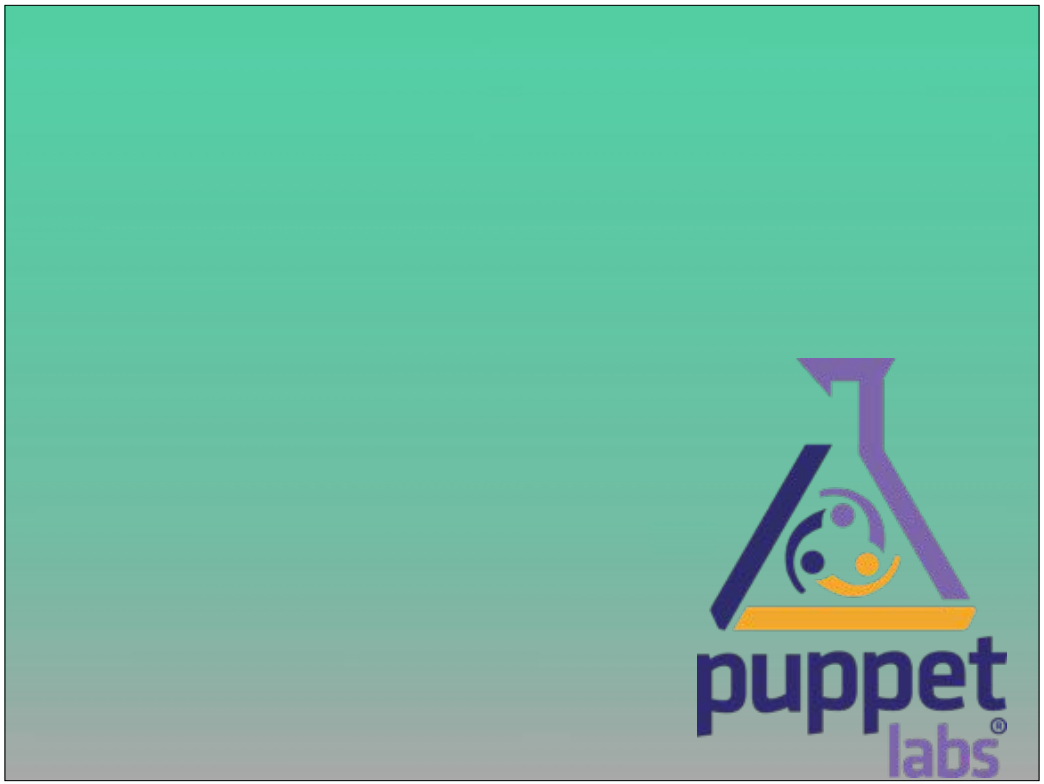


Similarly interacted with most often for user logins, but for when you need more power, MagerValp's loginScriptPlugin can also beat just about all the other processes that would load during login to make sure your task runs _first_, like an overexcited youTube commenter FIRST. It has gotten easier to set up and configure since he first released it, and you do want to be a little cautious about how much you try to make it do, with great power comes great responsibility, but as long as you aren't making poor assumptions in your script, which he specifically highlights in the README, you should be good to go.

Now onto the main event: managing preferences via profiles or otherwise. Munki can now push config profiles, which was a great addition if that's all you needed. If you weren't up-to-date, that's the new icon of the managedSoftwareCenter that updates and optional installs are presented to clients through.

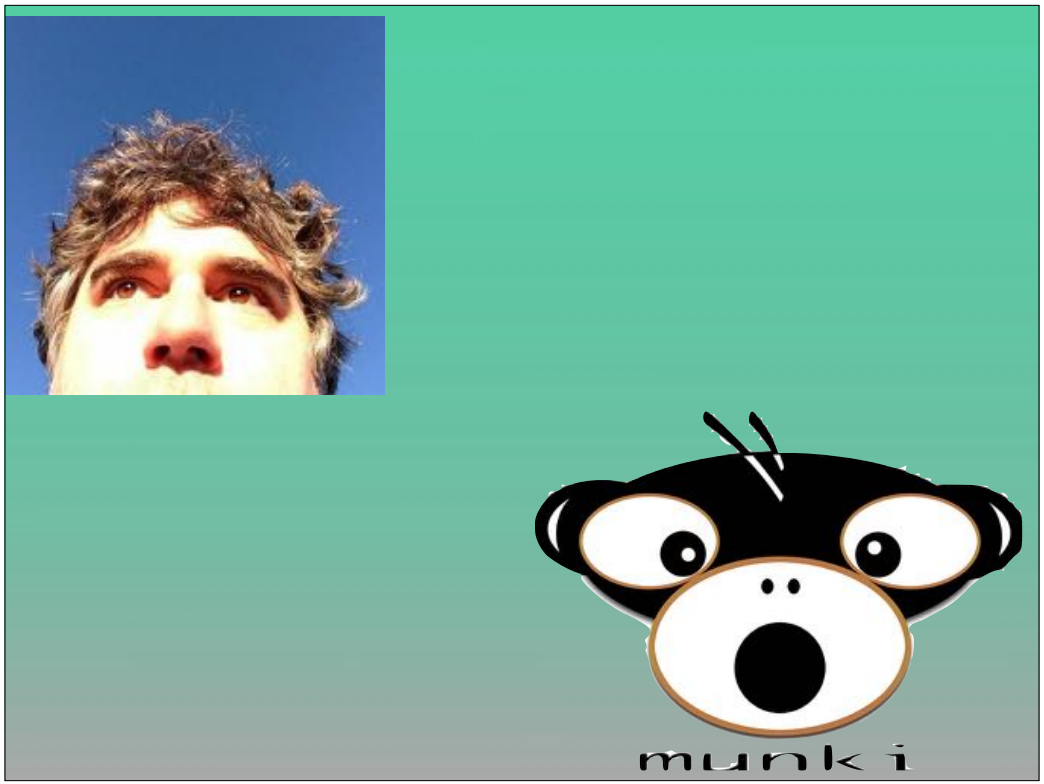
Tim Sutton had originally paved the way with mcxToProfile, which was extended by Greg Neagle for migrating from MCX, and also uncovered some features not exposed in other mgmt systems by default to regain some mgmt control when using profiles we used to have with MCX.

And something that Graham gilbert also collaborated on, a way to deploy profiles even if you're not on Munki 2 or you don't use munki, here's a script to roll it into a pkg so you can make sure it's applied.



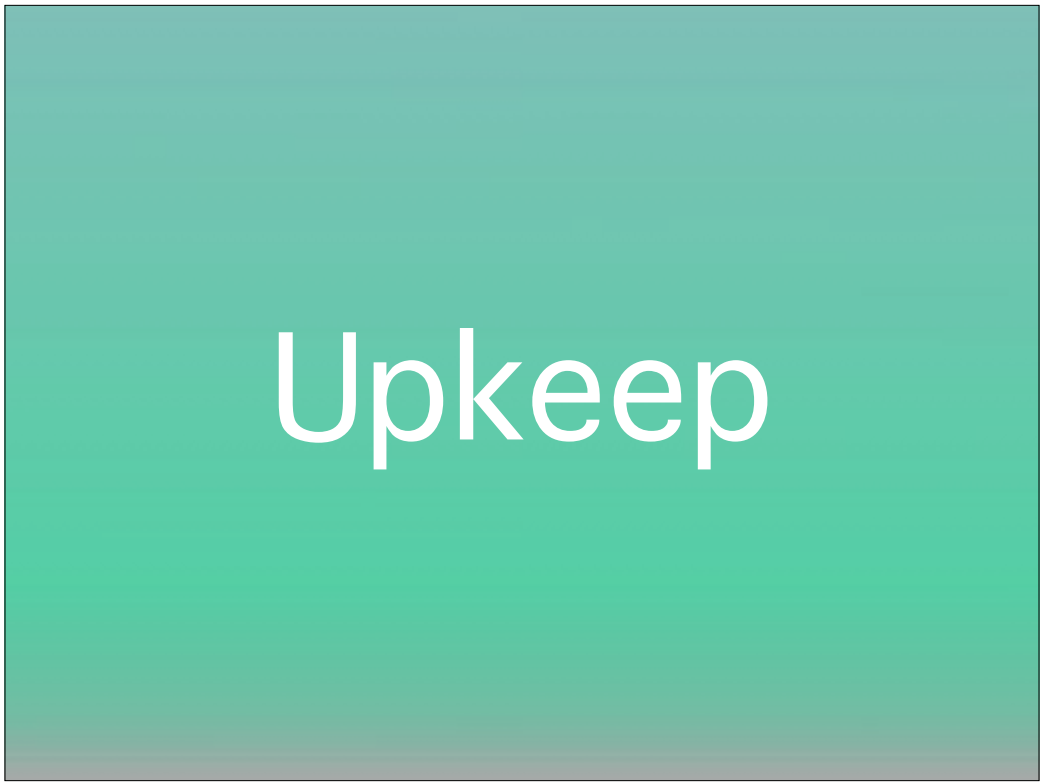
Puppet, which essentially roots your box and can do practically anything you can achieve on a linux/unix system, also provides security benefits, which you usually want from your mgmt system. It can secure how munki communicates over the network, it can setup your munki web server securely, and it has functionality that can sign your configuration profiles, if you're like me and consider the green Verified checkmarks in SystemPreferences a 'good thing™'. Puppet is non-trivial to setup itself and get talking over the network, but to take advantage of what it can provide to munki can be very minimal -

there's a great series by NickMcSpadden where he goes through some of the options on his OSX Dominion blog, and come talk to me if you're interested in signing profiles with Puppet.



munki and outset are certainly not hard at all to use for configuration as long as you take the relatively little time to get them going and deploy them.

Taking the time to set friendly defaults is what sets great admins apart from BOFHs. I'm pretty convinced most of your customers would actually prefer to create value for your organizations rather than go through wizards or run afoul of the rules that enforced settings would have helped them stay on the right side of. They don't wonder where that new ap is if you add a postflight to munki that leverages dockutil to shove it into their dock. And another thing munki is currently tinkering with the idea of supporting is adhoc-tasks: let your end users clear font caches, or add a network interface like tethering to an iPhone or use a thunderbolt adapter if they're standard users.



Well configured and secured systems mean less helpdesk calls and more productive customers. Munki and Puppet can use actual versioning for what you deployed, you don't know how frustrating it is to have an MDM wipe away a config with no history of what it was like, and when you're enforcing mgmt it's not the time to feel like you can't do something repeatably, or with the auditing that you can easily add to these tools. now I can't **make** anyone care to put in the effort to follow better practices, but you can implement as much as you like, and as you get more comfortable, most of these tools REDUCE the amount of effort that's required to have a safe, happy user base.

(Ben:)Config profiles? Perrrrllleeeaasseee..

With the JSS being an MDM, it can push profiles to clients as per any other MDM & as well as perform such MDM actions as remote lock & remote wipe.

There are some payloads included within the JSS, & you can upload pre-created profiles &/or generate your own on the JSS by uploading a plist.

By using the inbuilt profile payloads & leveraging the variables available to MDM delivered profiles a level of flexibility can be achieved in profile delivery that has yet to be met in the FOSS space.

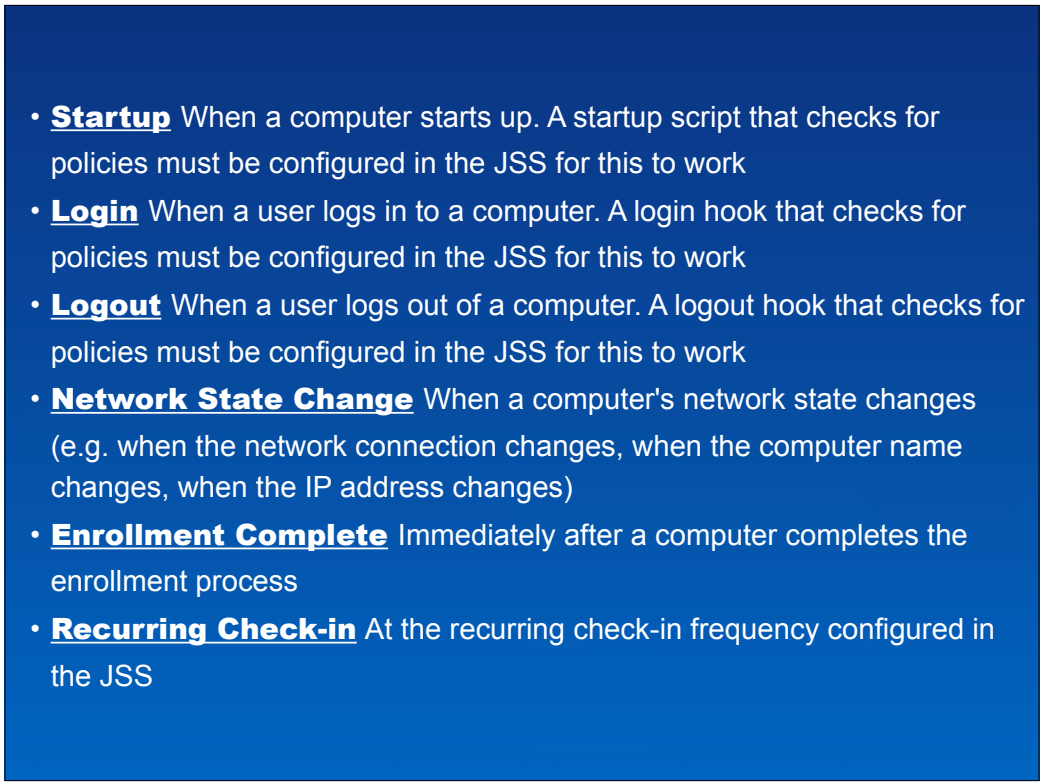
For example, we deploy a profile to all MacBooks which is a “User-Level” VPN profile with the username set to $USERNAME.

If something is missing you can create an "extension attribute" which is a script that runs when reconning a device & will report back it's result. This can then be used as a scoping mechanism for a Smart group.

For example, years ago I would scope wireless profiles to only devices with Wireless Cards, which would be scoped via an EA i created.. (now all Macs ship with wireless cards this is largely unneeded)

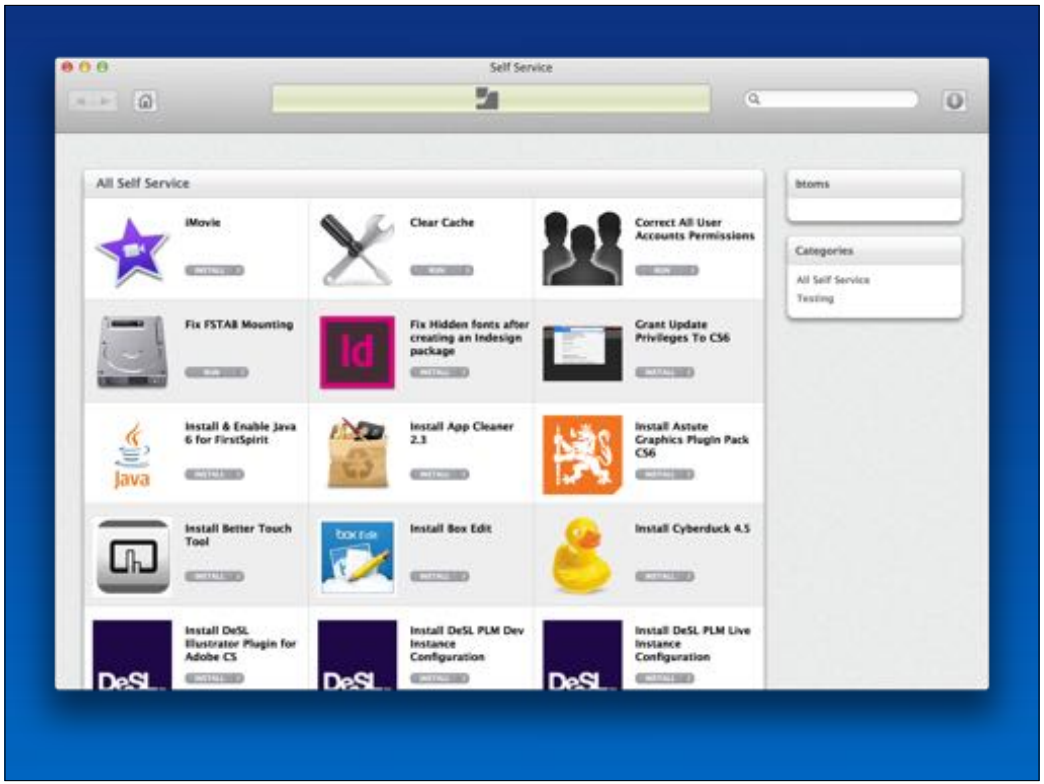
For example, years ago I would scope wireless profiles to only devices with Wireless Cards, which would be scoped via an EA i created.. (now all Macs ship with wireless cards this is largely unneeded)


NGRP
BAWK
© New Relic

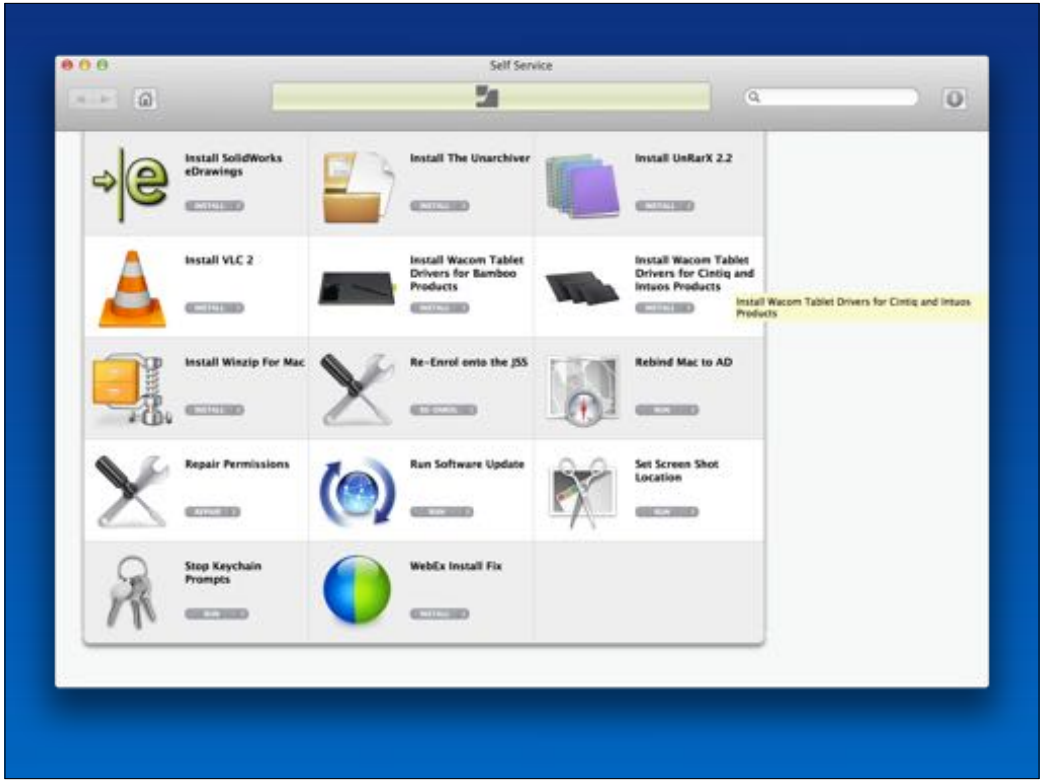


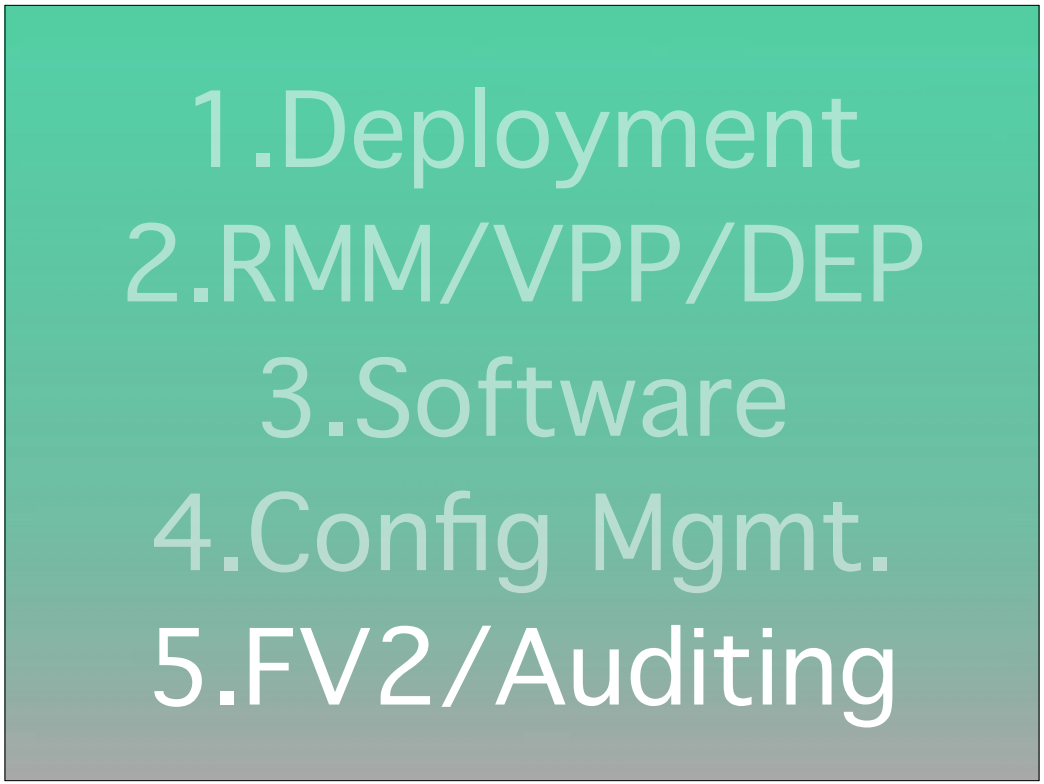
I've mentioned policies so far in regards to installing software, well scripts can also be run by policies & there are a number of triggers that can be used for both.

As well as custom triggers & even triggers via iBeacons.



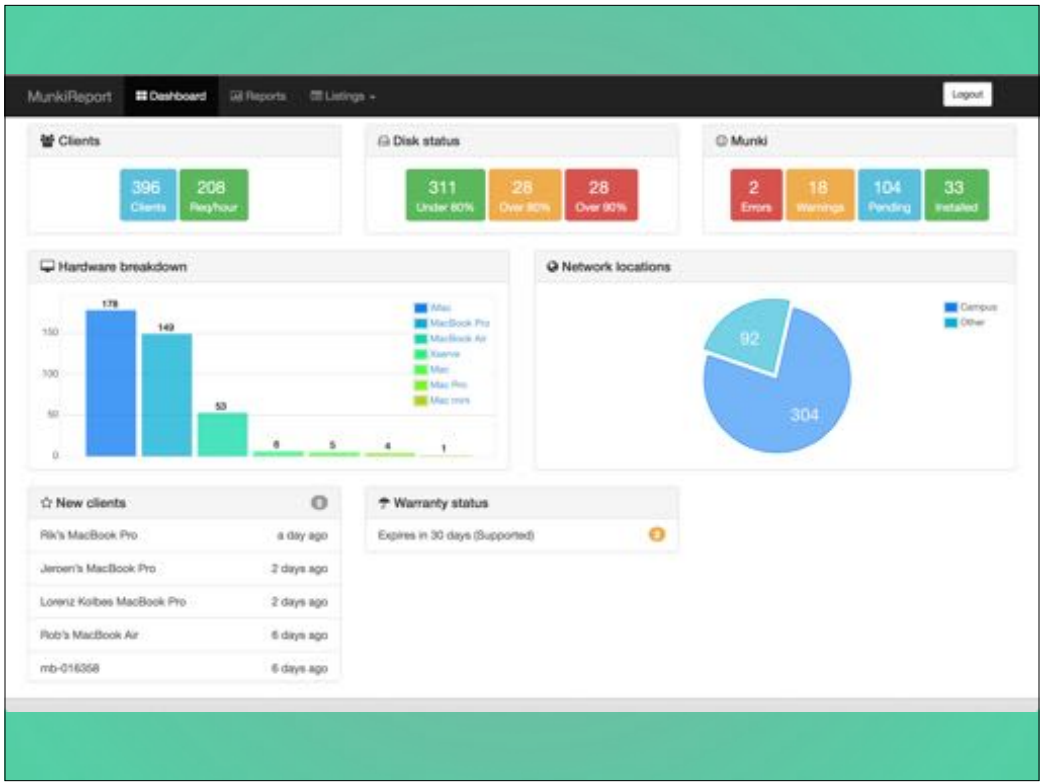
Policies can also be ran via Self Service (show examples of what we do), some examples here, our users can rebind to AD, correct home permissions etc.


Self Service
Install SolidWorks eDrawings
Install The Unarchiver
Install UnRarX 2.2
Install VLC 2
Install Wacom Tablet Drivers for Bamboo Products
Install Wacom Tablet Drivers for Cintiq and Intuos Products
Install Wacom Tablet Drivers for Cintiq and Intuos Products
Install Winzip For Mac
Re-Enrol onto the JSS
Rebind Mac to AD
Repair Permissions
Run Software Update
Set Screen Shot Location
Stop Keychain Prompts
WebEx Install Fix

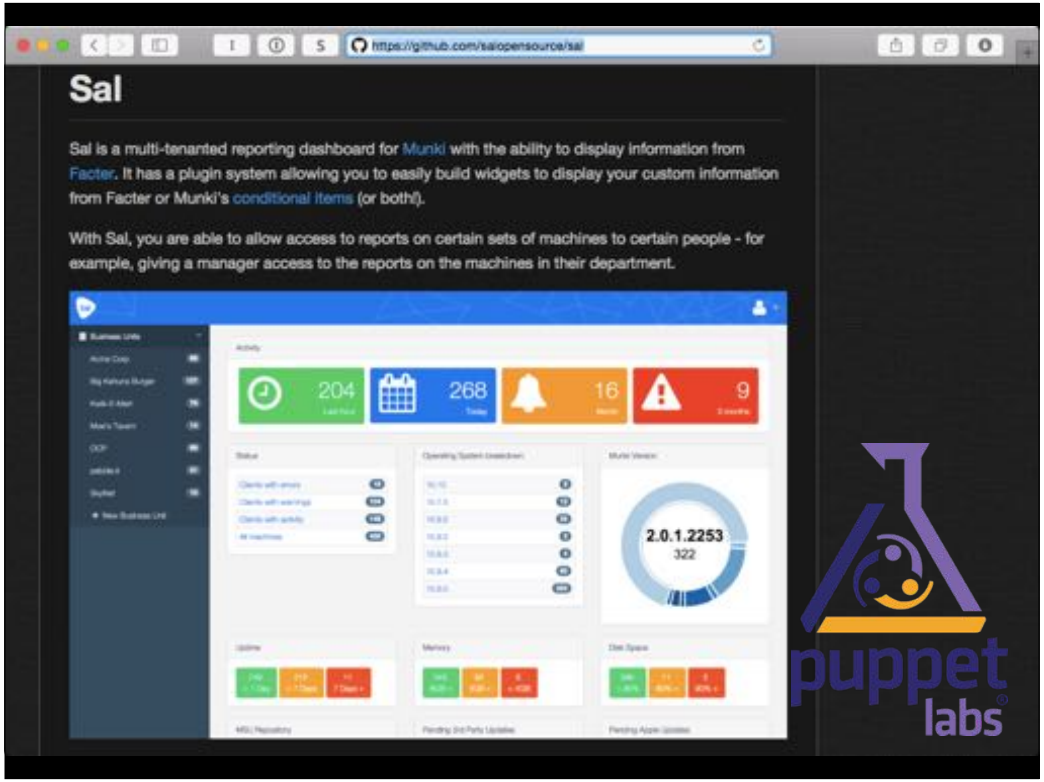


Oh man oh manischevitz.This is the last round and the last chance for Ben to drink my milkshake, but I must warn the faint of heart out there, there will be blood. Taking it slow, starting with basic inventory like reporting and licensing…

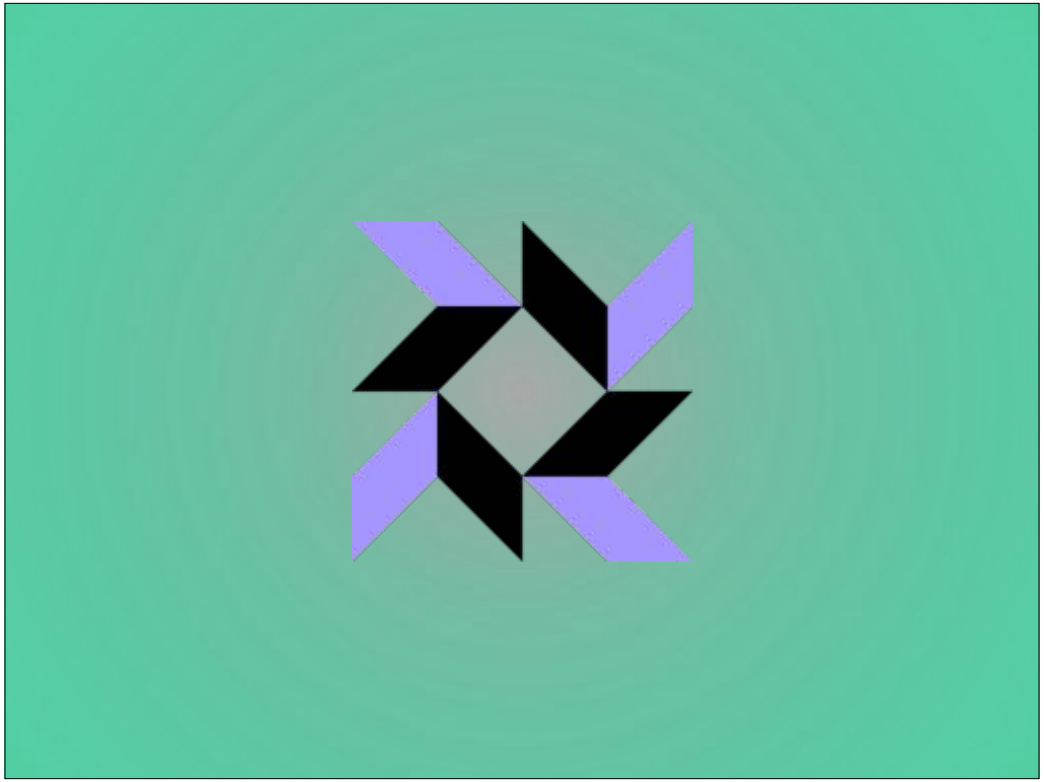
With the assumption you can install and manage an app on a server somewhere, this isn't too difficult of a prospect. Munki is a great foundation, and server-side Greg Neagle's own MunkiWebAdmin is a pretty comprehensive implementation of a console on what's going on out there on workstations that report in. It has support for tracking users and software licenses,



but MunkiReportPHP may be easier if you're used to PHP apps or you have a server team that can get those basics in place for you. It's also possibly easier if you want to customize it, to hire a PHP dev since that has historically been a large hiring market. I prefer Graham Gilbert's Sal but that's because I'm more of a python guy and because of the customization options it has.



Those aren't that hard to spin up, and can replicate if not far surpass the features and functionality of many commercial products. Client-side, puppet has a companion open-source product called Facter that just got an overhaul in C++, and while there are many 'batteries included' in the base pluthora of things it'll report about a computer, with their concept of 'external facts' you can easily display and report on those in a server product like Sal.

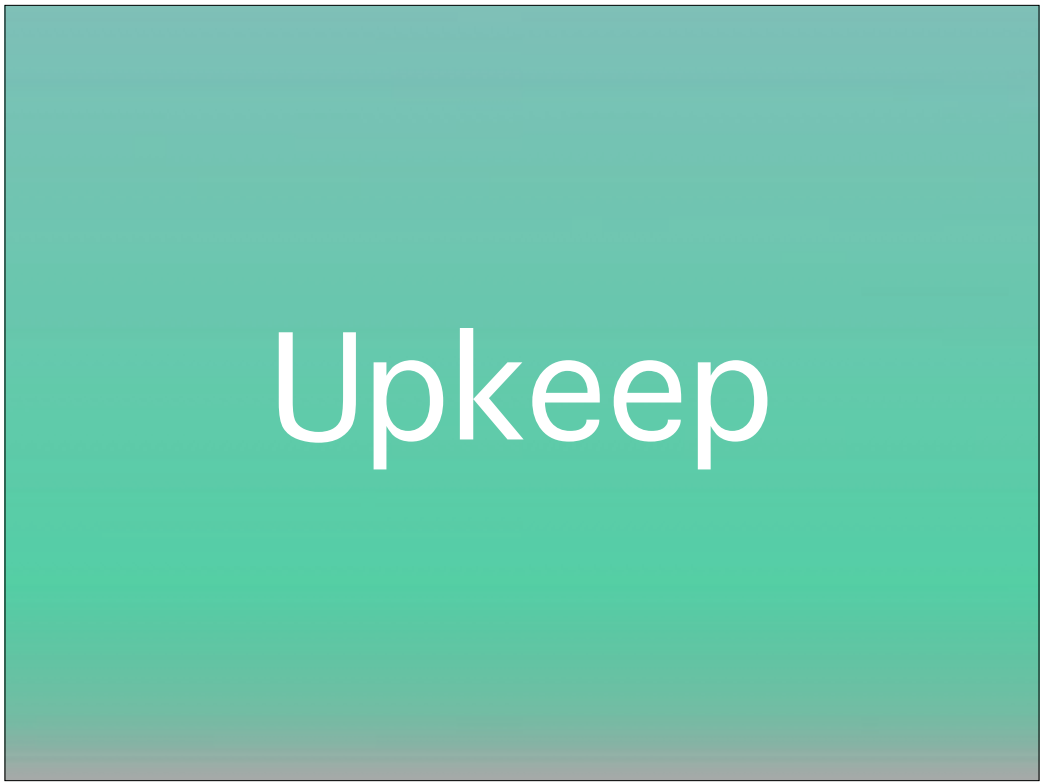


But the real up-and-comer is Facebooks osquery, which is super high performance, super platform-native, and comprehensive in it's way to proactively not just report when things are known to be exploitable, but keeps watch to clue you in before anything gets exploited in the first place. It's also seriously non-trivial to take advantage of at this point, BUT they've offered anyone who needs help getting started personalized assistance because they really want to see a community of users banging on it and helping supplement the experiences and feedback they get from the handful of macs they support. Which at last report was 15000, AND they mentioned just recently they also use it in production on the handful of servers they manage. I mean, I don't use facebook, I don't know how many servers they'd really need.

Now on to Filevault. This would be the time where Ben asks the audience to really believe in Fairies so I don't kill tinkerbell with the rage I have about how poor some vendors handle interactions with filevault.

But to focus on the solution I use, we customize Graham Gilbert's Crypt client to make it trustworthy to end users, and escrow the key to his cryptServer complement over https. The nice thing about having demo'd MunkiWebAdmin is once you're deployed on django app, things like Sal and CryptServer look pretty similar, in fact I deployed it with puppet by altering a module Nate Walck and Vincent Vranchan had collaborated on for MunkiWebAdmin. Building the Crypt client was probably the trickier thing to get right, but it really is a teensy app, there's not much reports from folks who've had issues.

Staying with filevault, end users encrypt the machine on their own time, so they don't have to wait for however long until we're done, they are therefore also the only user you see at the pre-OS EFI window where you authenticate to Filevault, so that makes it feel more like their computer. You can even open up logins to auditors or the customers themselves to obtain get out of jail free in case they forget their password or it goes out of sync - and the approval mechanism in cryptServer means you'll find out about it WHENEVER and WHOEVER asks for it, unlike some systems...
And with the inventory and auditing systems, when you have data tipping you off in advance of an issue, you have the possiblity of getting in front of a user, hopefully before they notice their search results are being hijacked or their HD is full or some



I didn't mention logging shippers like Logstash and greylog, but besides the relatively small and static data needed to escrow a filevault recovery key, these mainly server-focused processes do require pruning and attention. Notifications and monitoring are only as good as the fidelity of the data and the personnel to respond to it. logs that don't get viewed aren't useful to retain. Managing databases aren't necessarily why any of us got into this business, just like I don't assume everyone is comfortable with being programmers right off the bat. But the engines that are under the covers of the solutions I discussed really are well understood quantities for the most part, and one data point is Sal was maintained in a large deployment for some time and Graham says he performed database maintenance a total of zero times…

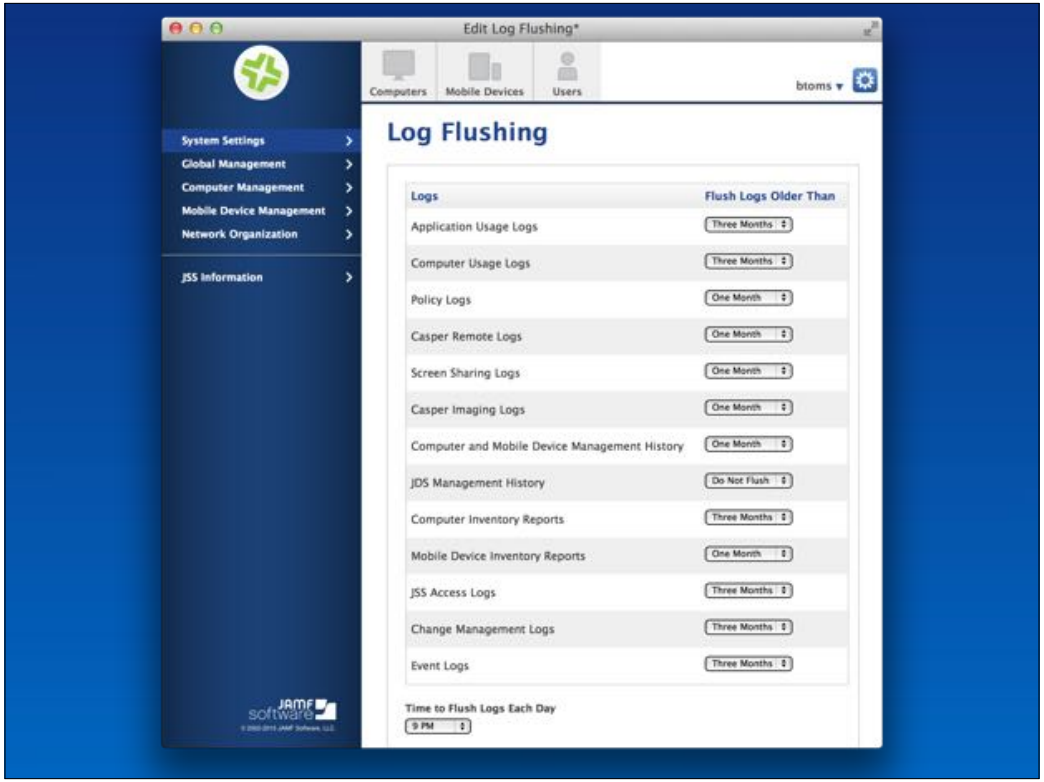
(Ben:)The JSS is really the brains of the suite…

It can be hosted Mac, Linux or Windows hosts.. People have even dockerised it.

There is a tomcat front end that can be clustered or load balanced..

& a MySQL backend..

The maintenance of which for a JSS admin is really just scheduling the backups of the DB as required.. using the JSS utility.jar



& setting the “log flushing” schedule.. hehe log flushing..

These things need to be set for your environment… & are often set during your jump start.

Anything else DB related & I call JAMF Support.

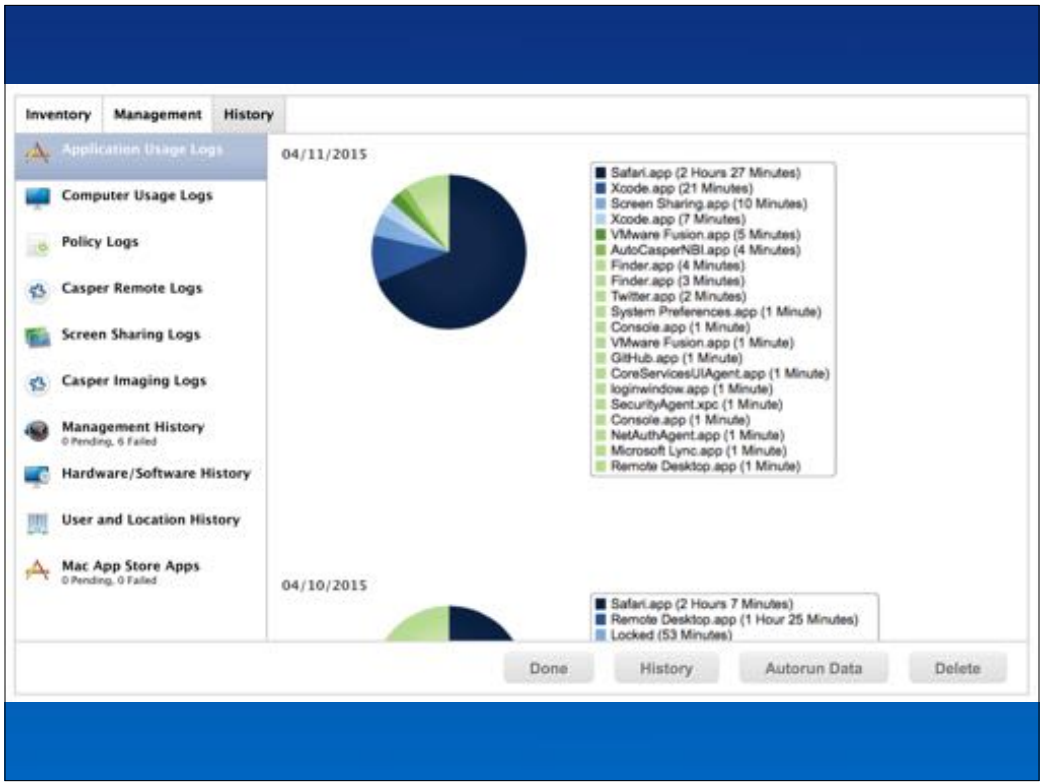
Soon into the call we might be going into the MySQL command line with me like:

ok.. well that's wrong.. I am a CCE..

it's more like:

I'm not a DBA & don't want to be.

When it comes the the JSS's MySQL db.. But that's theirs to support & for me to bust.

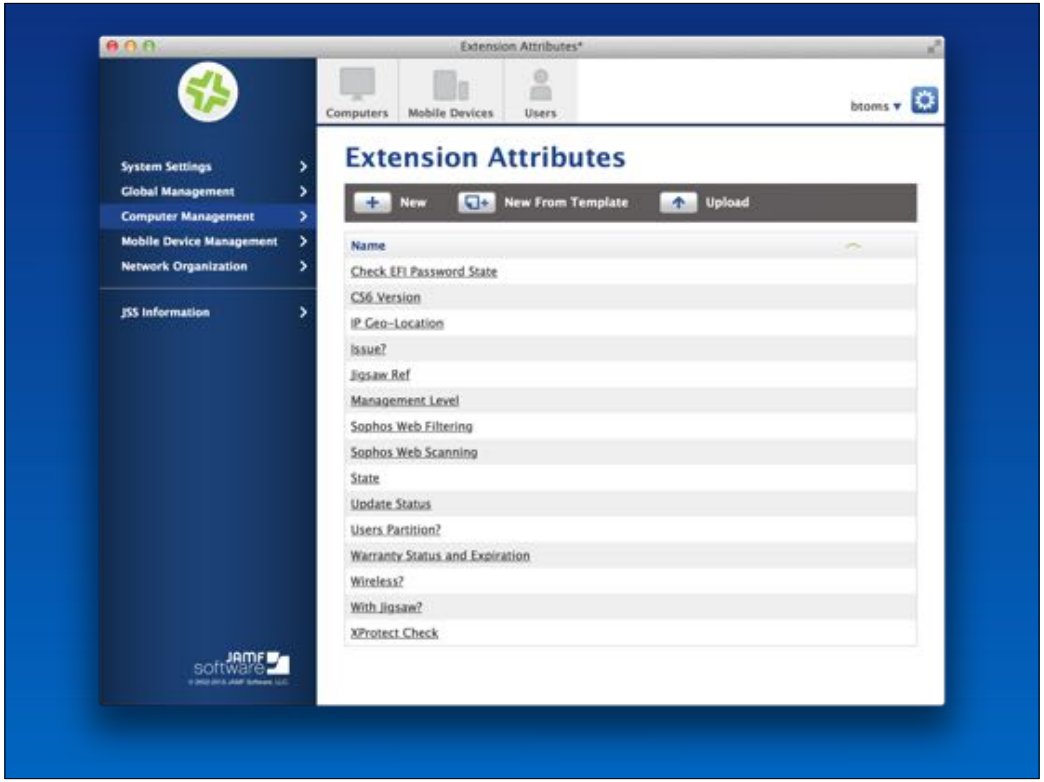


There are a number of logs available for each client within the JSS & they should go back as far as the “log flushing” level set for each log…

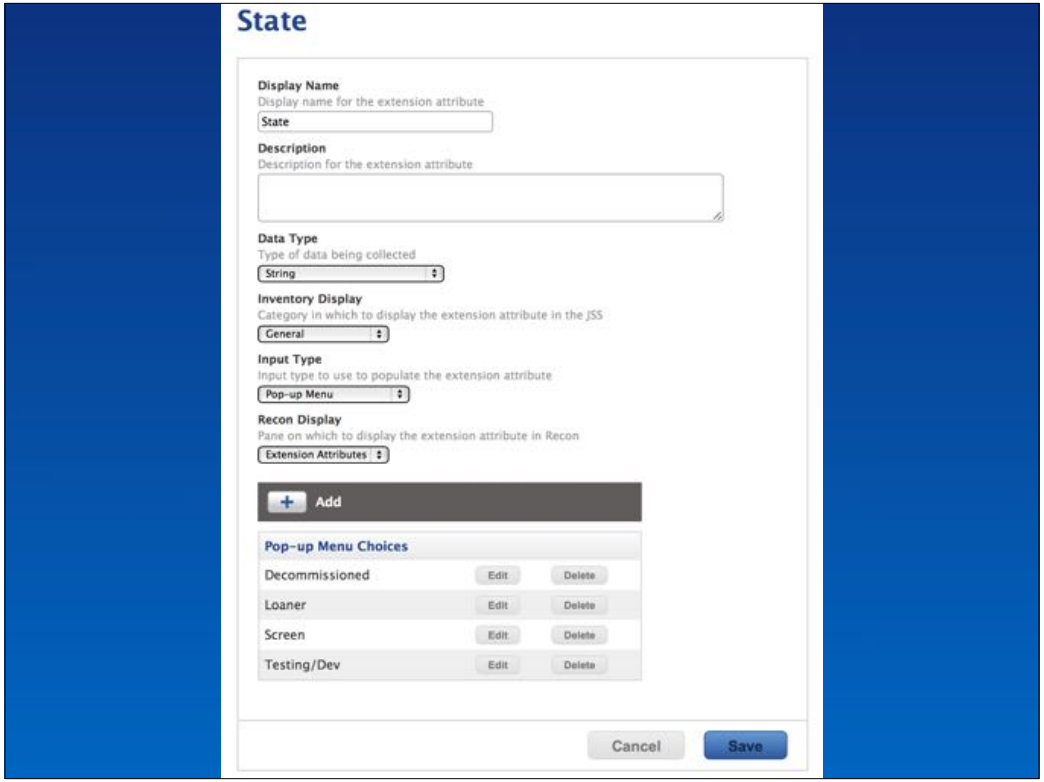
These can help give a per-device audit trail.

For example, if your JSS is pointing to your AD & you’re running recon via a policy at login… the JSS will update the Macs user & location history at login..

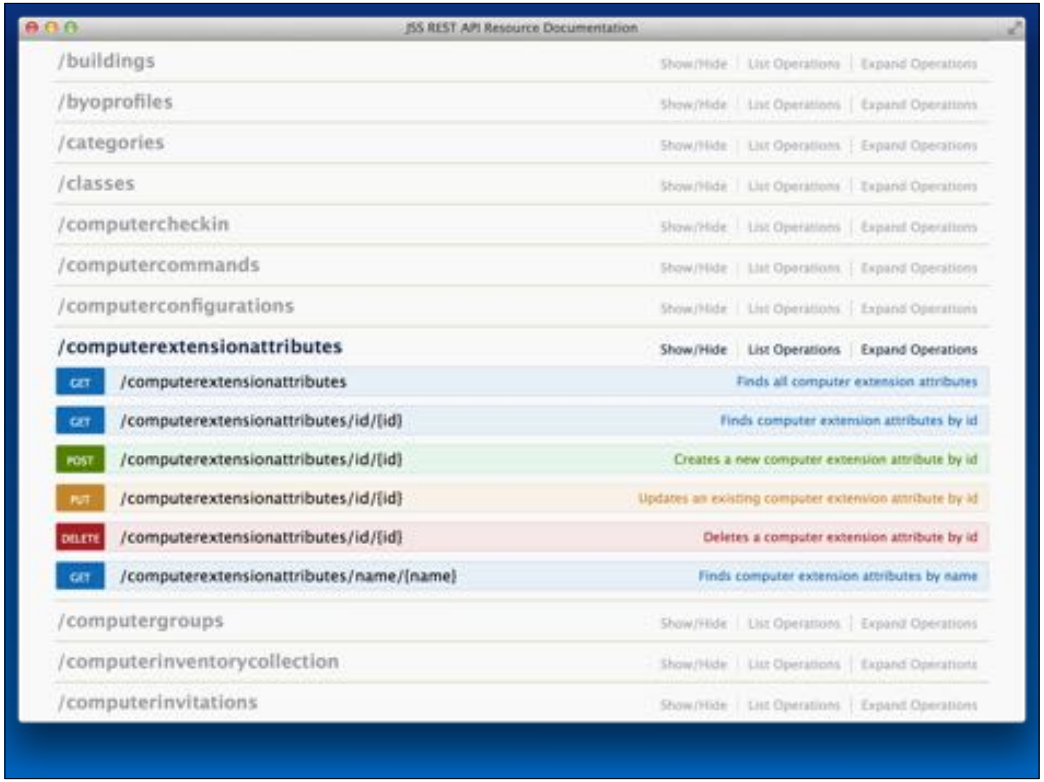
Which then updates the relevant Smart Groups..



As mentioned, Extension Attributes can be used to gather information via scripts to augment the existing inventory.



However, these can also be manually entered via things like pop-up menus… or



via the REST API..

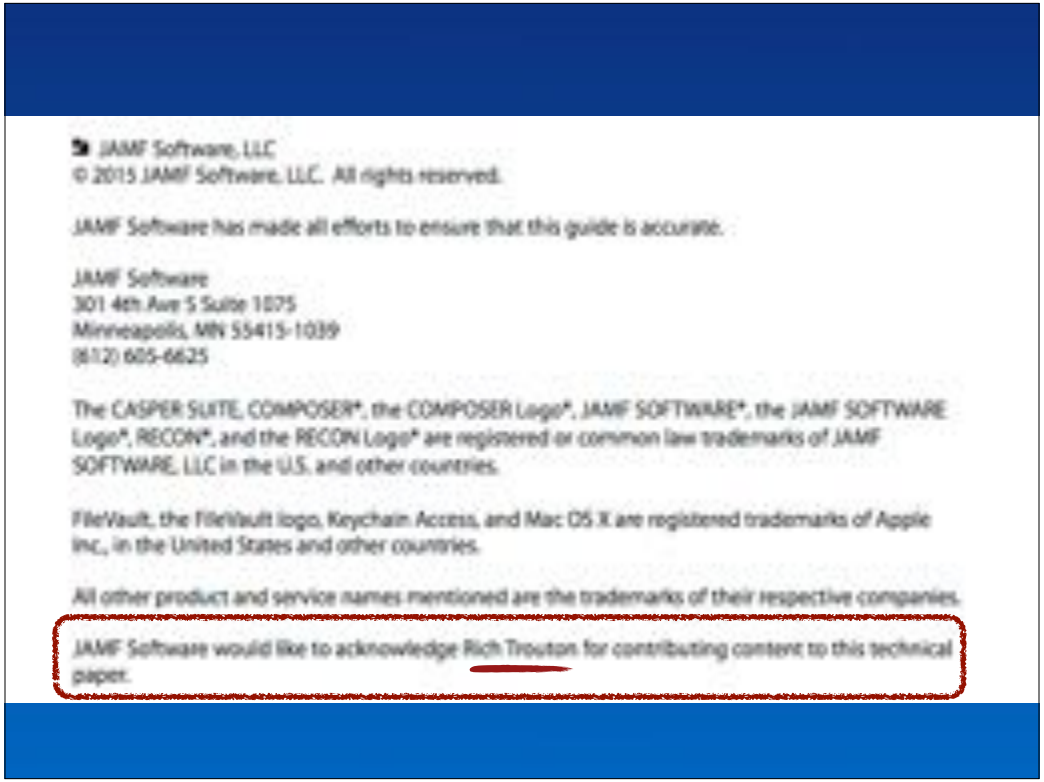
Updating the EA will then update any smart groups attached.

Just a quick shout out for Bryson's excellent blog posts on the JSS REST API..

At this point you'll probably expect me to breeze through the FV2 section.. & I will only because I don't use it.

But if I did, i'd go through Rich's blog.. & you don't need a bit.ly link for that.. if you're looking at FV2 .. you'll find your way there.

& go through the JAMF white papers..



JAMF Software, LLC

JAMF Software has made all efforts to ensure that this guide is accurate.

JAMF Software
301 4th Ave S Suite 1075
Minneapolis, MN 55415-1039
(612) 605-6625

JAMF Software would like to acknowledge Rich Trouton for contributing content to this technical paper.

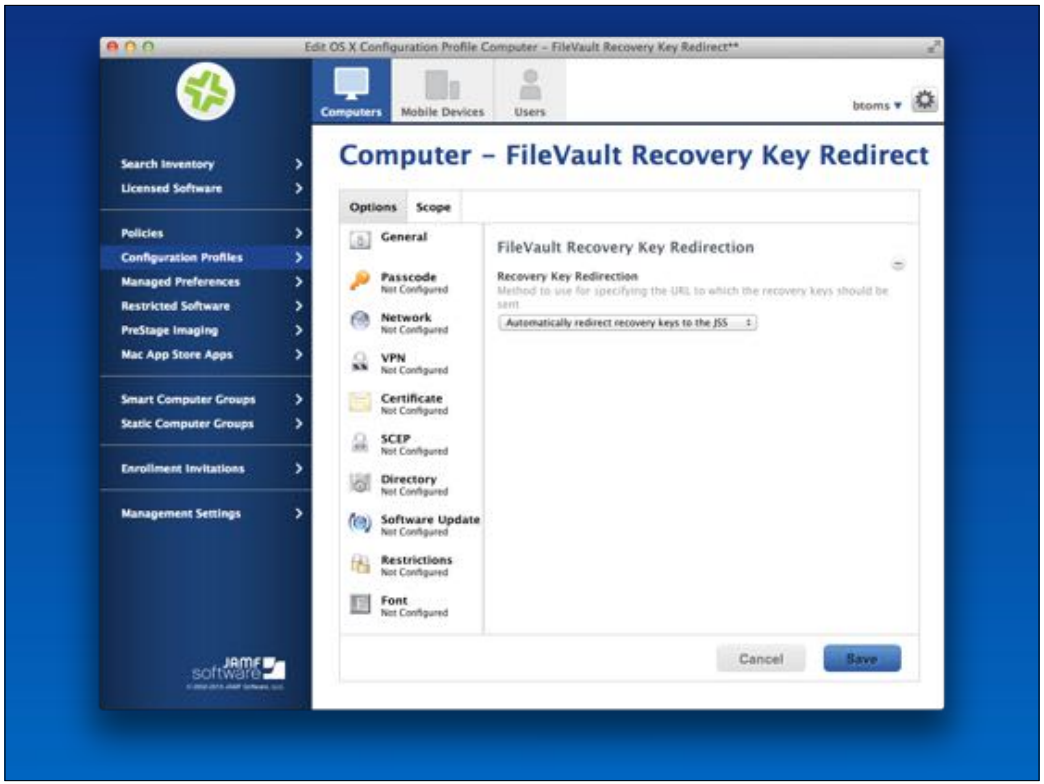
oh look..

But some webinars & videos from conferences later: The JSS can issue, renew recovery keys for individuals & institutions..

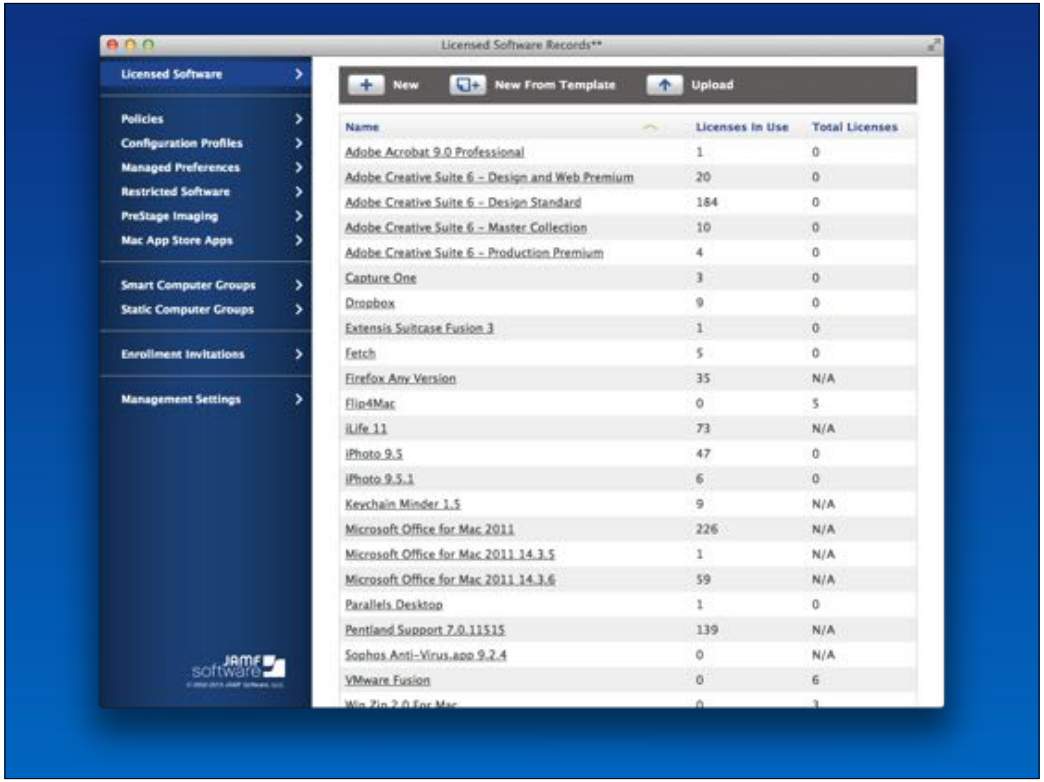
The devices key should be tied to the devices record in the JSS & the status of the encryption shown & be used as a scopable item for a smart group & so FV2 can be enforced via the JSS.

So you could create a “Smart Group” which contains only un-encrypted MacBooks & then have a policy that encrypts.

Once a MacBook starts the encryption it will be dropped out of the smart group.



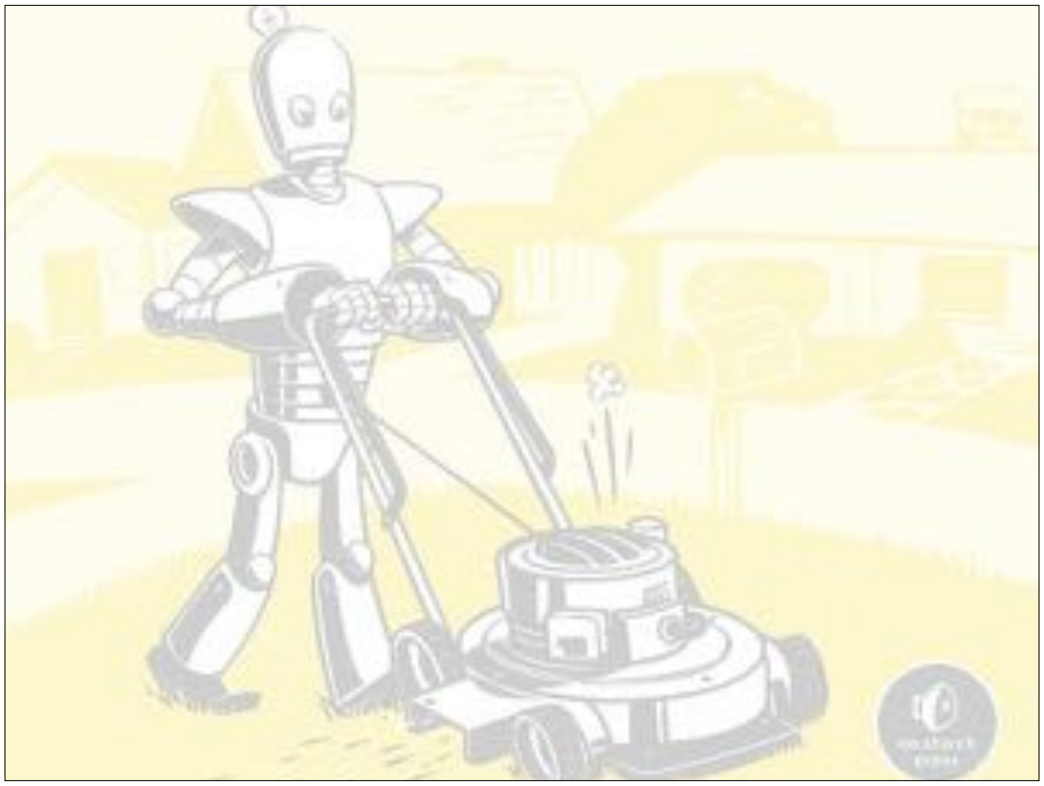
Another option available to the JSS as it's an MDM is the config profile that redirects the FV2 recovery key, escrowing it onto the JSS even if a user has self-enabled FV2 on a managed mac themselves.



The JSS can also be used to report on Software installed via “Licensed Software Records”

There are a number of templates available, & you can create your own.

So in summary, we have to pressure these vendors to suck less, Apple included. I'm only as glad to have all these options as I am with the fact that I can therefore deliver a better experience to my customers, and continuously improve as a sysadmin who writes code. The win I wanted out of this session was to convince you to not wait for some vendor to wrap your job in a bow and make you a GUI monkey, or have your job title incorporate the name of some vendors products: we're sysadmins.



Ben’s obviously not sticking his head in the sand, but you can also clearly see he’s not exactly best buy mobile material either. Intellectual curiosity is unfortunately something few of the representatives of our vendors exhibit, and I’d love for there to just be more of us folks who actually do care about what’s better. I think I can count on you kind folk to not wait, join me in collaborating on solutions to automate the dumb away.

Thanks to Ben, thanks to Penn State as always, thanks to all the folks who contribute to open source, thanks to the vendors who let us poke at them with a stick, and thanks to all of you for your time and attention, I look forward to your vote in the upcoming election!

Many thanks to you all for coming & watching the video.

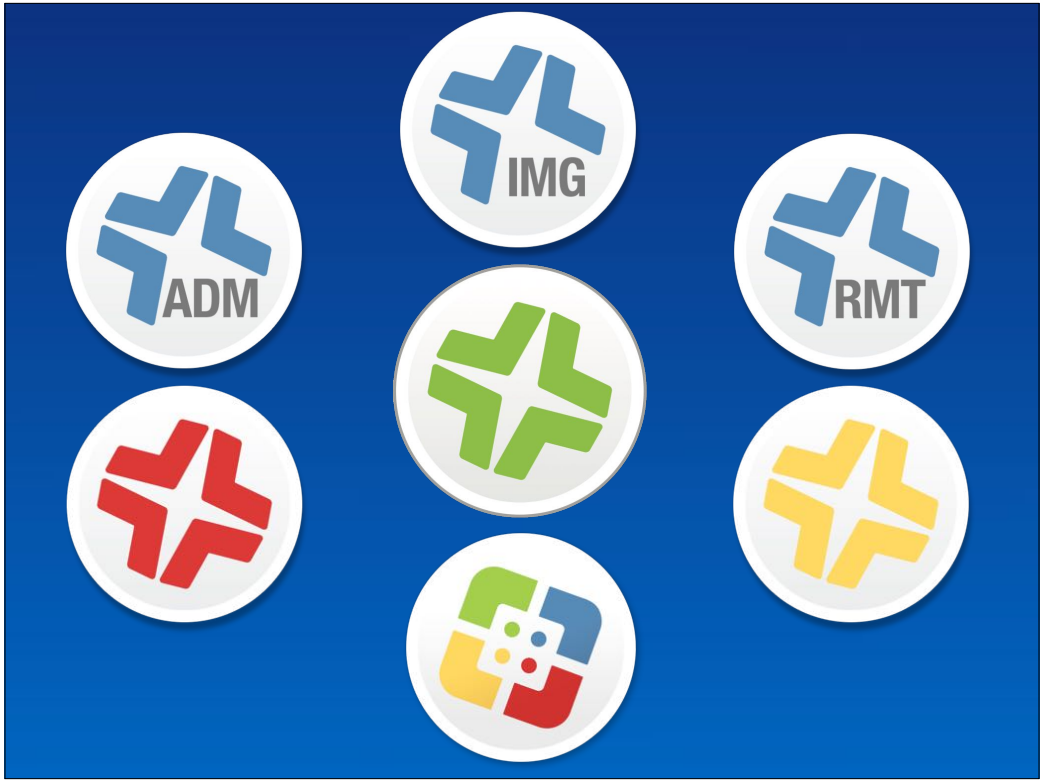
& many, many thanks to Allister for him so dutifully playing the part of the panto-villian.

I knew that Allisters passion for the open source tools he's mentioned would shine though & that I would take some licks..

But i'm big enough & ugly enough to take it, & will resist retorting

Let's make it clear: I am a fan of FOSS, I have contributed to a few projects & have a few of my own out there.

But it does take time, & these projects need to be properly maintained & supported.



The irony of this is that in me using a paid for solution, has freed up enough time for me to contribute & release a number of FOSS.

The holistic approach from a paid for solution really can be beneficial, although maybe not a flexible as a cobbled together solution from FOSS.

But when you deliver something or have a pull request merged.. it is a feeling of elation that you don't get from having a defect closed or a feature added with a closed source solution.

You truly are rocking out with your spock out..

Anyways.. Whichever camp you're in & however you vote.. I hope you found the talk entertaining, informative & enlightening.

The idea of this talk was to show both sides of the Mac admin coin & to try & bridge the divide.
At the end of the day, we're all working to the same end result.. we all want our user/colleagues/management to give our managed macs the thumbs up.

Thanks again.

Open (and/or Free) vs Closed Source

AKA Steel Cage Death Match

Ben Toms and Allister Banks

Feedback: http://j.mp/psumac2015-91

Worksheet: http://url.aru-b.com/openVs