MANAGING MULTIPLE MACHINES

FREEBSD & RADMIND

WARNING

AUDIENCE PARTICIPATION IS MANDATORY

SURVEY QUESTIONS

(PLEASE RAISE YOUR HAND IF...)

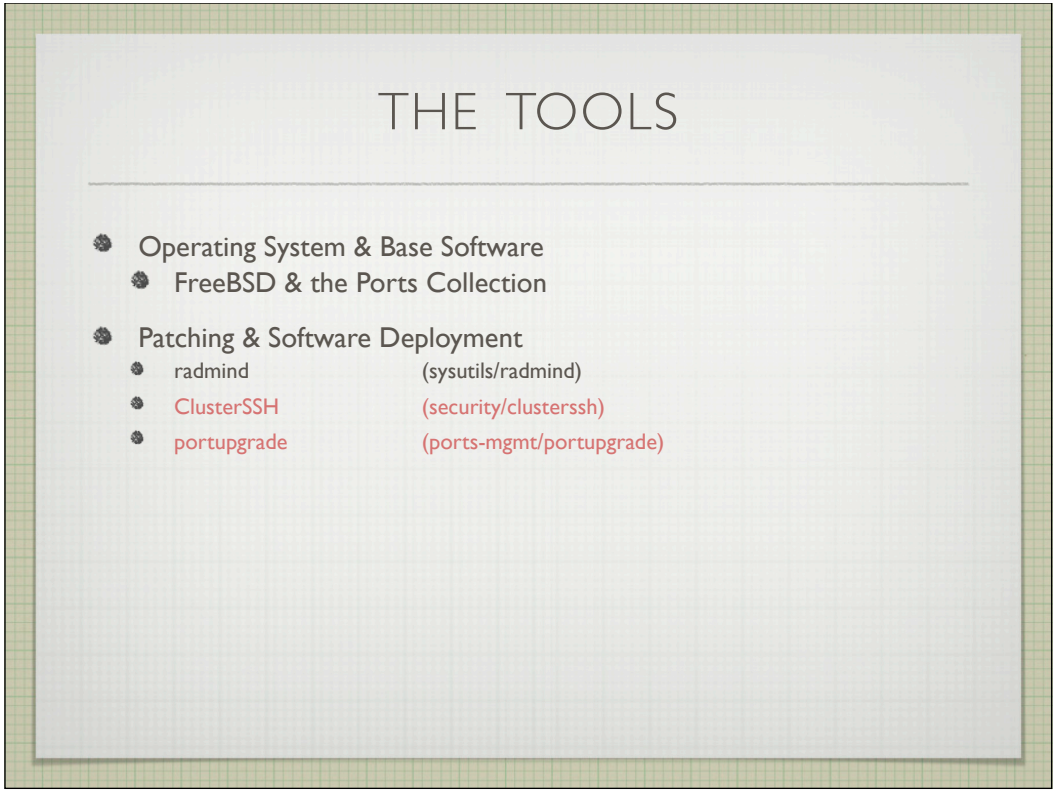
- You currently manage an environment with multiple *NIX machines
  - Keep your hand up if your environment is heterogeneous (Multiple OS or Architectures)
- You currently use an automated deployment/patching system
  - radmind?
  - Puppet / Chef?
  - something homegrown?
  - Something else?

THE TYPICAL PROBLEM

- Walked in to an inherited mess
- Multiple machines with different operating system versions
  - Sometimes different operating systems entirely
- No central patching/deployment system
- No redundancy
  - (This may not be such a bad thing…)

THE GOALS

- Simplify Management
  - Standardize operating system & "base software"
  - Centralize patching & software deployment processes
- Ensure Stability
  - Add redundancy where it is missing
  - Create identical Test and Development environments



Operating System & Base Software
For the sake of this discussion we're going to assume you're using FreeBSD and that you're installing most of your ancillary software from ports. None of that is written in stone however: If you're running some other *NIX system everything stays pretty much the same.

****

Patching & Software Deployment
We're going to talk about radmind as a tool for deploying and patching systems.

We're not going to talk about ClusterSSH, but it's a useful tool that lets you run one command on a bunch of systems interactively. Install it and play with it - you may find it useful.

We're not going to talk about Portupgrade or Portsnap or any other of the myriad ways you can update ports. They're all well-documented, they all work pretty well, and you can pick whichever one you like best.

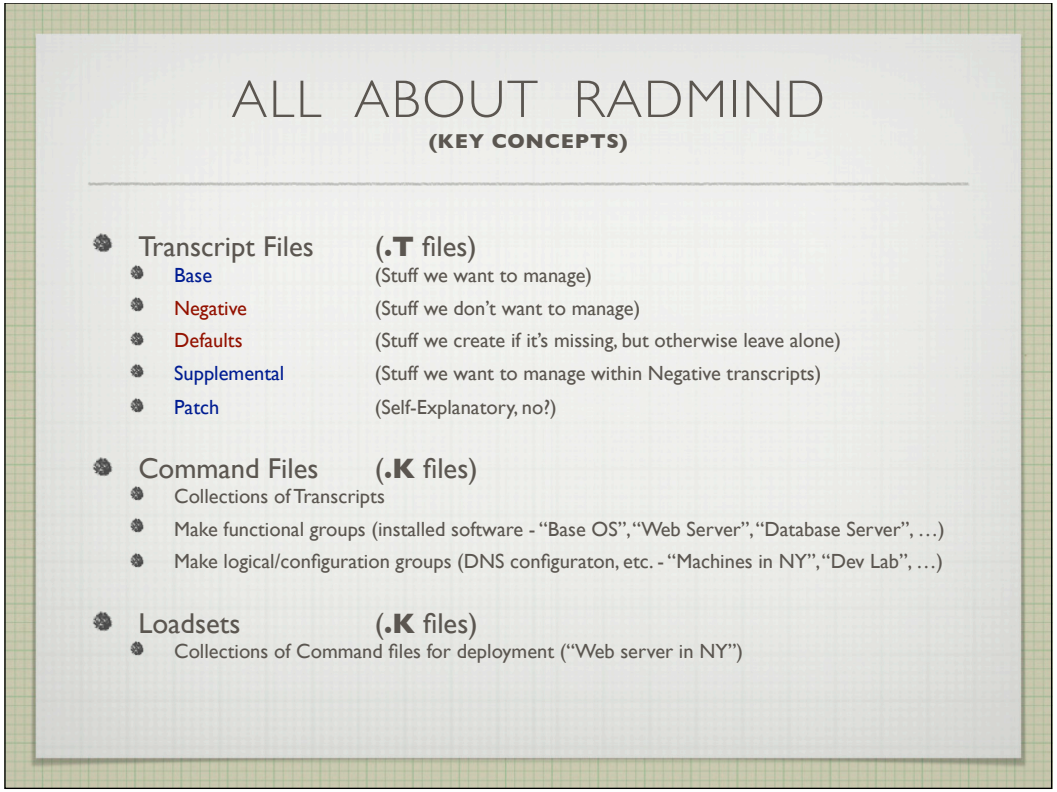
***

Authentication, Authorization & Password Management
We're not going to talk about any of this stuff, but please feel free to ask questions or corner me in the hallway after this talk if you're interested.

ALL ABOUT RADMIND

(WELL, A LITTLE BIT ANYWAY)

- Developed by RSUG at the University of Michigan
  - Originally intended to manage Mac systems, but friendly to all *nix
- Back under active development!
  - `1.14.0` Release Candidate came out this month
  - The original developers are committing time to improving it
  - The port maintainer is actually using it in production



Transcripts
Transcripts are nothing more than lists of files (with sizes, checksums, permissions, ownership, etc. info too). They come in a few flavors that we'll talk about in a minute.

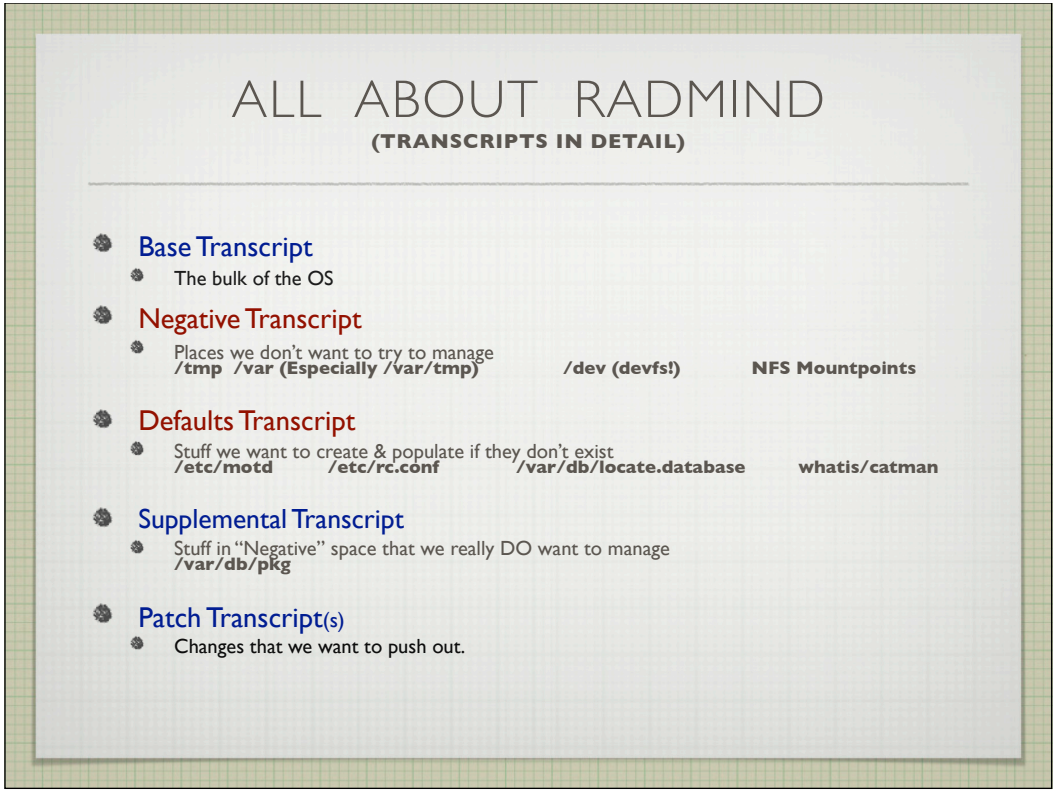
Command Files
Command files are collections of transcripts that make up a unit. For example:

The "Base OS" command file contains your base OS (duh), but also all the patches you have applied to it – You may have started with FreeBSD 8.0 & upgraded to 8.1, and that patch transcript would go in the "Base OS" command file.

Loadsets
Loadsets are Command Files that contain **other command files** – For example, if you have a webserver in NY your loadset for that machine would be:

1) The Base OS command file (because every machine needs the base OS)
2) The Web Server command file (because that's what this machine is)
3) The "NY" site command file (to point at the appropriate DNS servers, etc.)

Why this structure?
Easy – Because when I patch the OS all I have to do is add that patch transcript to the "Base OS" command file. All the Loadsets that include the Base OS will pick up the change.



Transcript Types

radmind really only has two transcript types -- Positive transcripts (in blue) list stuff we are managing, and Negative transcripts (in red) indicate stuff we aren't (with one caveat)

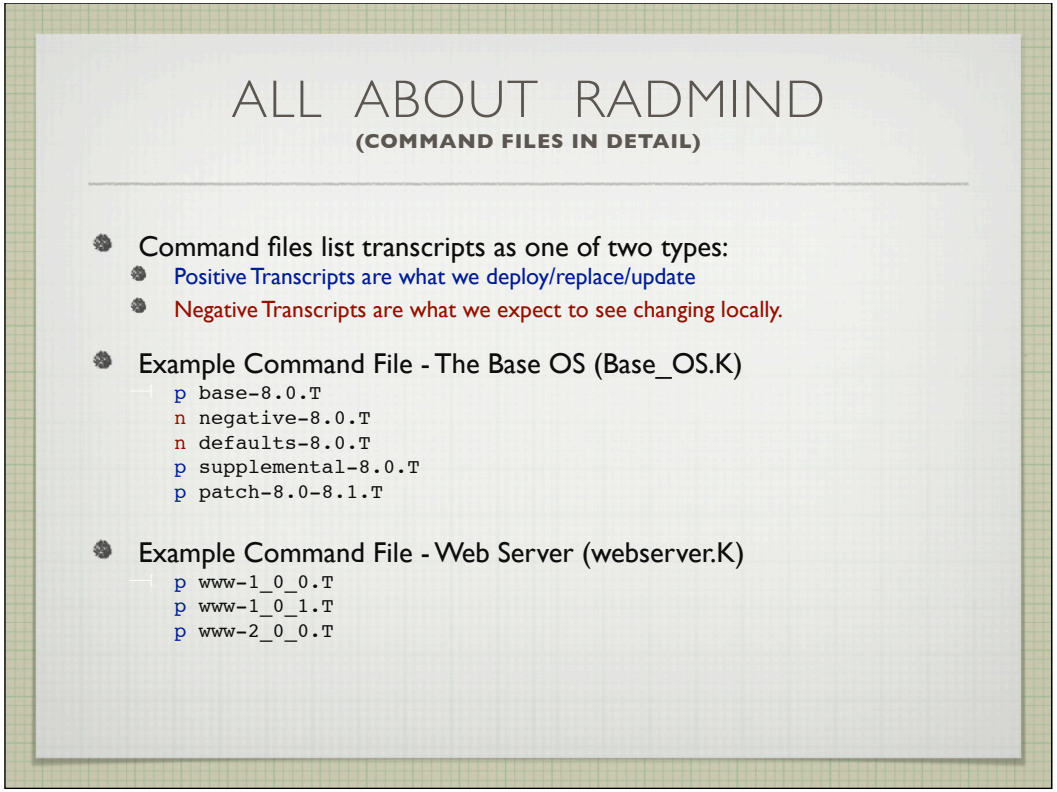
Base Transcripts are positive transcripts - for our purpose lets call this "the whole OS".

Negative Transcripts list "negative space" - The parts of the FS we don't want to manage. Usually this is a list of directories, but it can include specific files (like locate.db) that will change.
Files/directories listed in a negative transcript will be created if they don't exist.
Files listed in a Negative transcript are usually uploaded as zero-length files (magic flag "-N" to lcreate tells it the transcript will be negative, so send zero-length files)

Defaults Transcripts list files we want to create (and populate) if they don't already exist.
A Defaults transcript is really a Negative transcript, just create without the "-N" flag, so whole files are uploaded (& sent to the client if they don't exist)

Supplemental Transcripts are files (or directories) under negative space that we want to manage.

Patch Transcripts are self-explanatory, I hope.



radmind really only has one concept of a command file, but we break it down into two kinds – The first is a command file that contains transcripts ("command file"), the second a command file that contains other command files ("loadset").

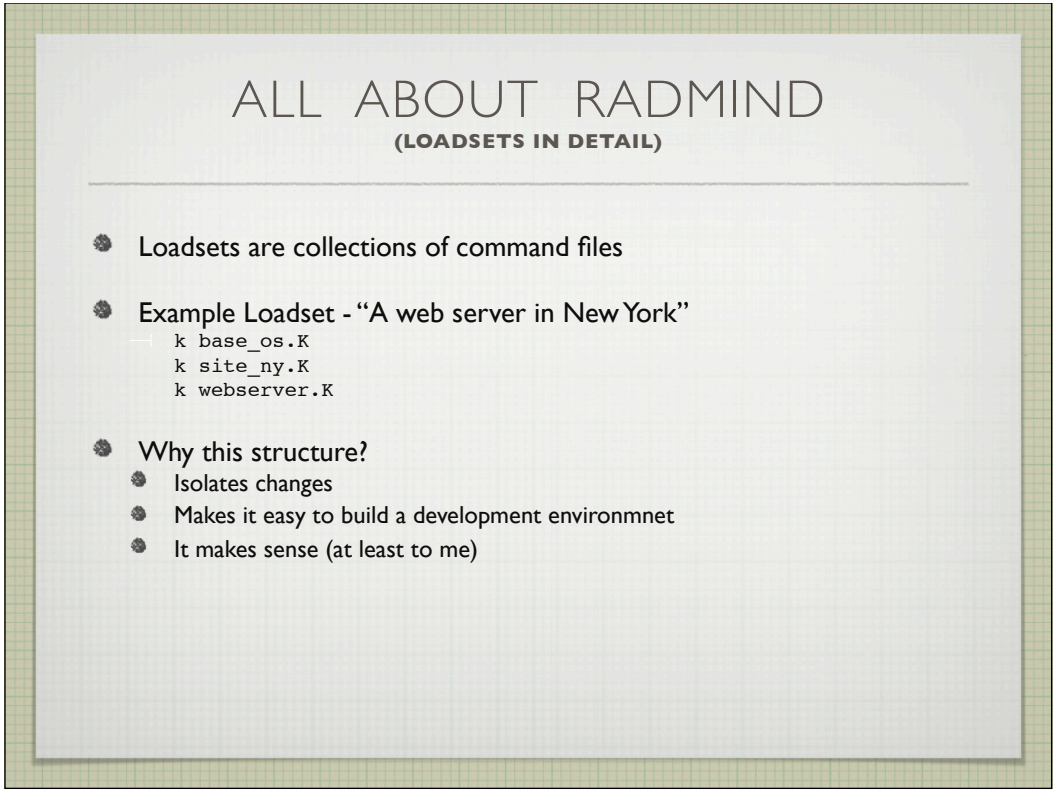
Example #1: The Base OS

The Base OS command file contains all the transcript types we talked about before – Base, Negative, Defaults, Supplemental & Patch

Example #2: Web Server

The Web Server command file only contains "patch" transcripts:

- The first describes the changes between the Base OS and a web server
- The rest describe changes from that point forward



Loadset files are just command files that contain other command files

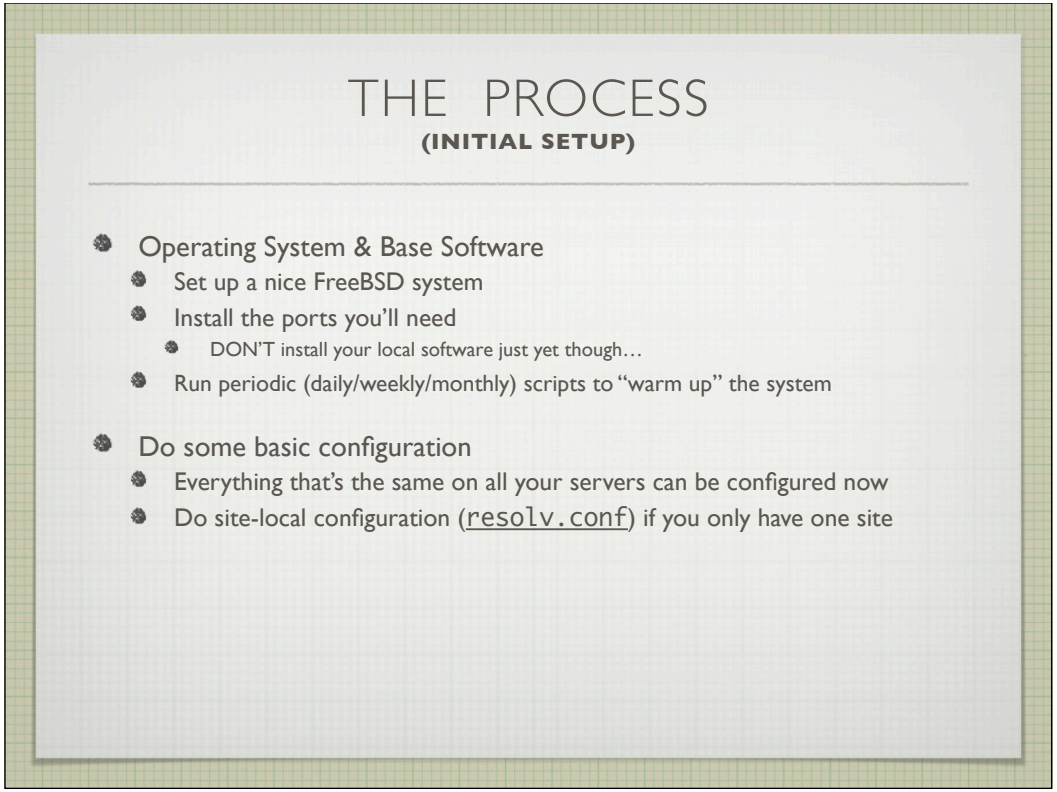
Command files look like what you see here.

Example : A web server in New York

- Contains the Base OS & patches (because all servers need an OS)
- Contain Site-Specific stuff for NY (like DNS configuration)
- Contains the webserver stuff

Why this structure? (again!)

- Isolates changes: Patching the web server stuff doesn't affect the base OS
- Makes dev environments easy to set up (site_ny -> site_dev and we're done!)
- It makes sense (to me anyway).



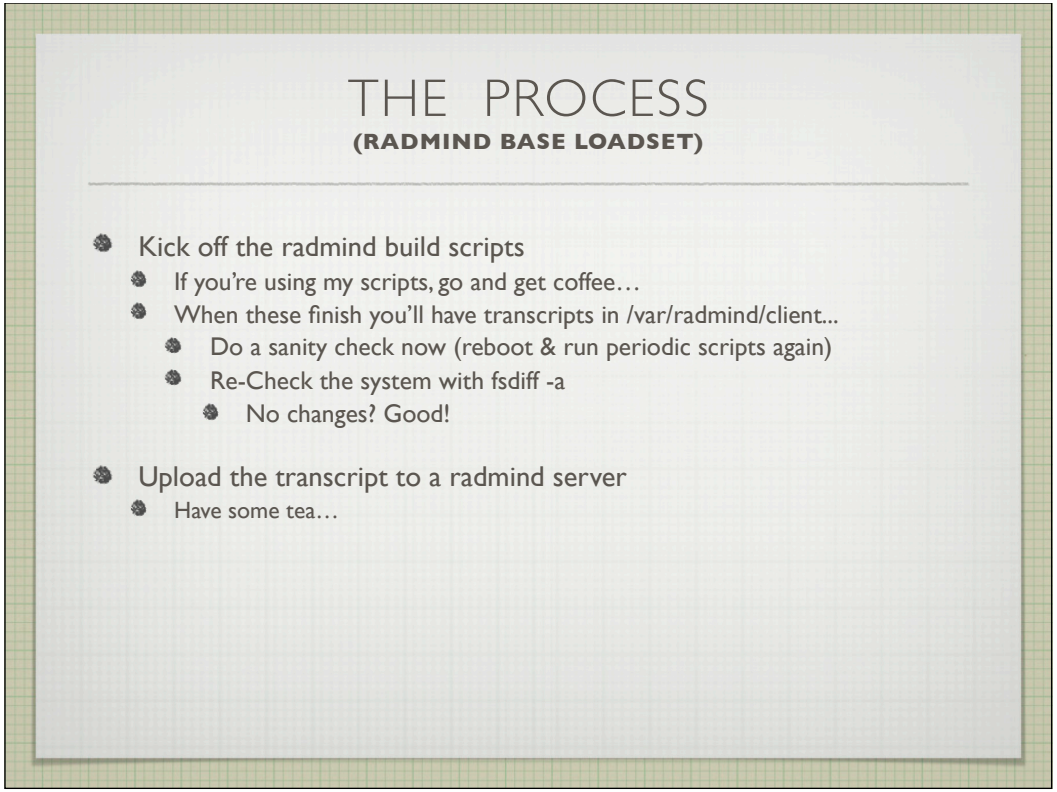
The initial setup for a radmind build system is pretty much the same as any other box: Install the OS and any extra software you'll need (ports).

If you have specific sub-types of servers (like a web server) you want to start with the base OS configuration first: For example, don't install your database software if you don't want it on all every machine. You're trying to create the most basic skeleton of a system here so that you can reuse it as a base for all the other systems you're going to build later.

One caveat

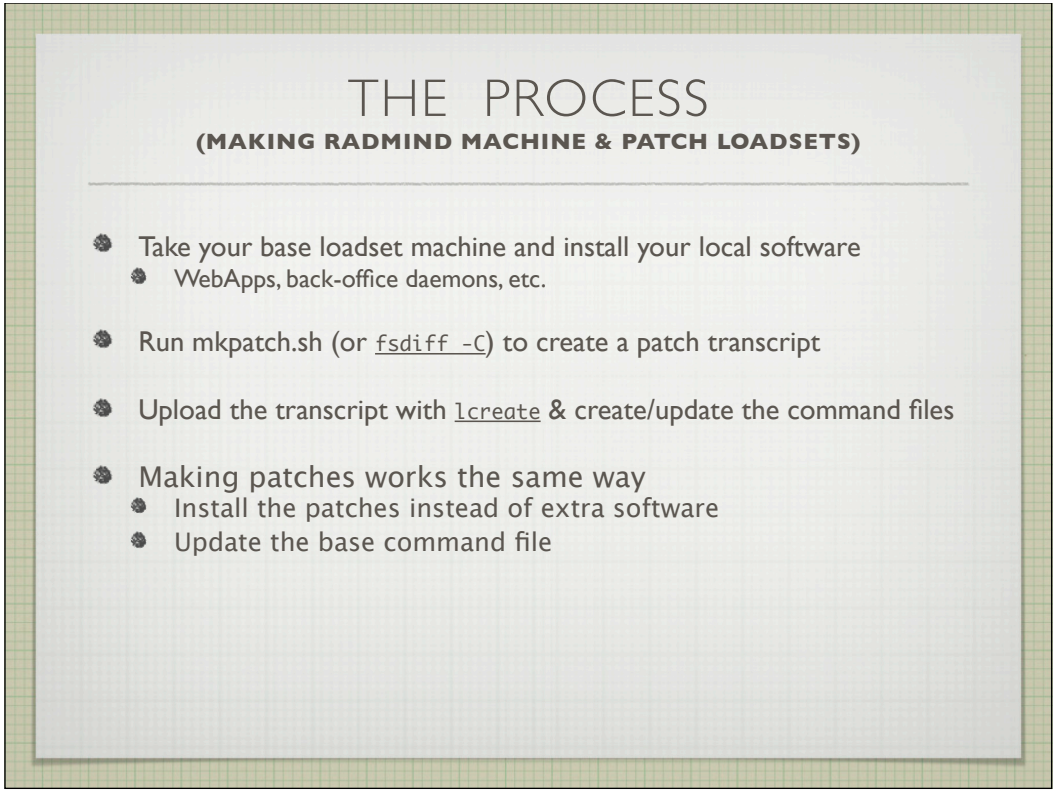
If you intend to manage `/var/db/ports` & `/var/db/pkg` you're going to want to install ALL of your ports as part of the base system (dependencies changing stuff in `/var/db/pkg` makes a mess out of things otherwise).

You could also not manage `/var/db/ports` & `/var/db/pkg` (leave them in negative space and don't add them to the supplemental list), but then `pkg_info` won't be accurate (and you'll need a dedicated build server so `portupgrade` et. al. will work correctly).



radmind Build Scripts

My build scripts are available



Say for example we have a web application we want to roll out. radmind makes this easy:

- Deploy a machine as a "base" server
- Install your webapp
- Make a patch transcript and upload it. Give it a good name like "webapp-1.2.3.T"

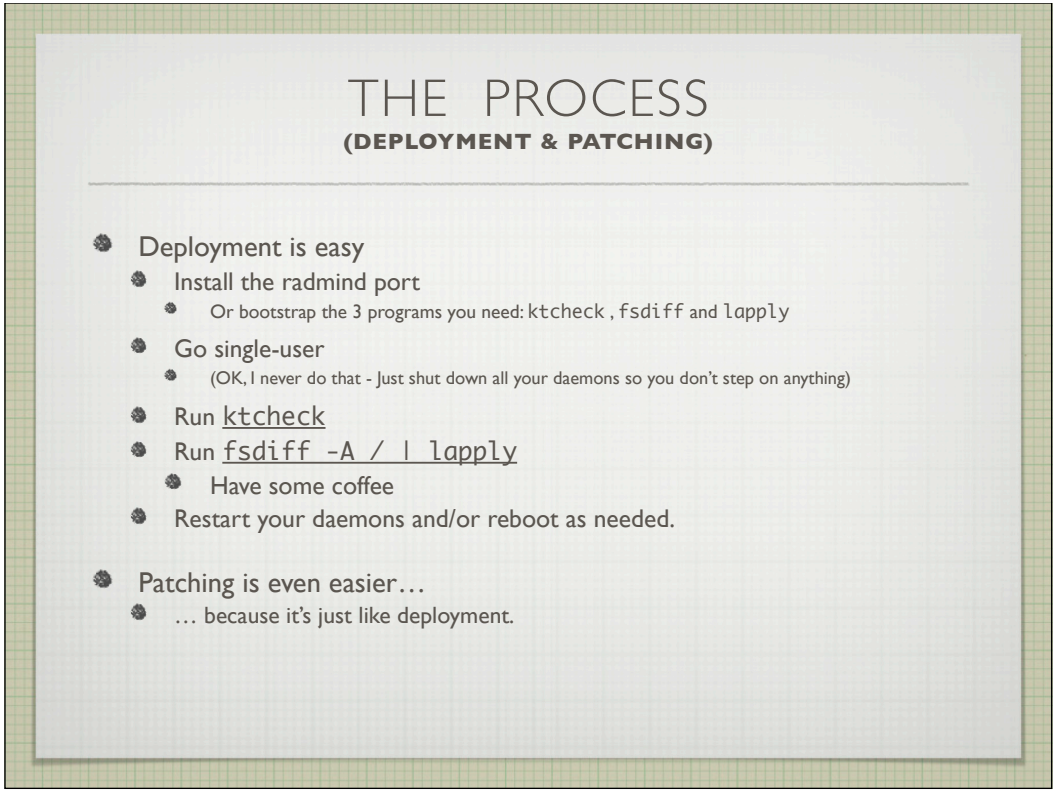
Now create a webapp.K command file that contains base.K (the base command file) and your new transcript file that you just created. That's the loadset you'll deploy to "webapp" machines.

***

For the sake of argument let's say that Apache is in the Base server loadset, and a new version comes out. How do we patch Apache?

- Deploy the "base" server somewhere
- Upgrade Apache the way you normally would
- Create a patch transcript (base-1.0-p1.T) & upload it.
- Update base.K to include the patch & re-deploy all your machines

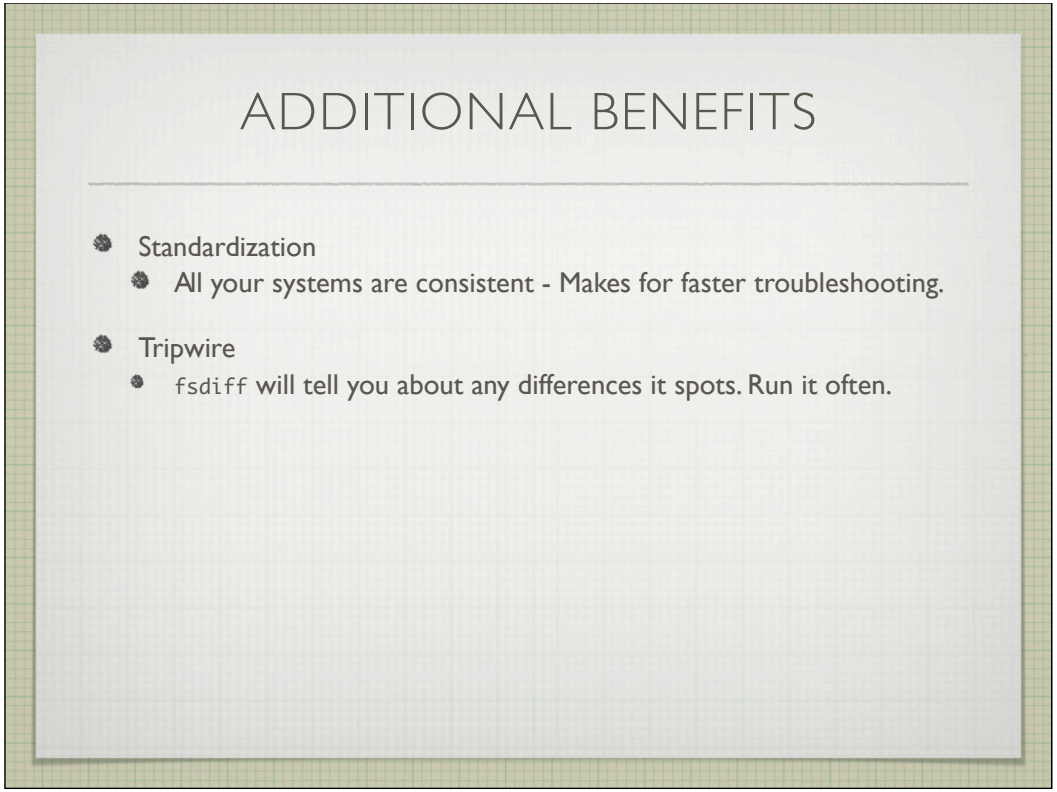
The webapp machine will get the patch because it incorporates base.K, so the Apache on your webapp servers is now all patched & up to date.



Deploying a system with radmind is simple -- Install the radmind port (it can be part of your default FreeBSD install process), run it, and let it do its thing.

You can use ra.sh for all of this too, but I like running the commands manually.

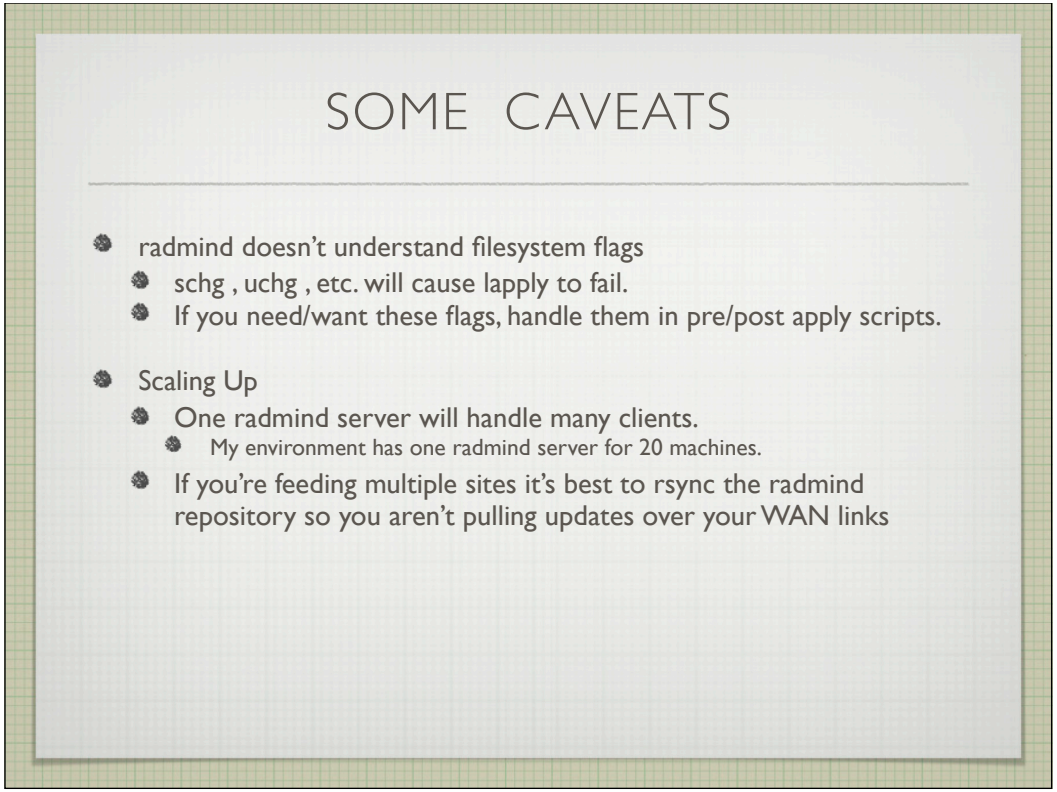
Patching with radmind is exactly the same process as deployment - You just don't have as many files to download.



Using radmind gives you a few ancillary benefits:

Standardization
All of your machines are running the same software, modulo some “extra” stuff.
This is the holy grail of managing an environment -- You know all the machines will behave roughly the same, so troubleshooting is made much easier.

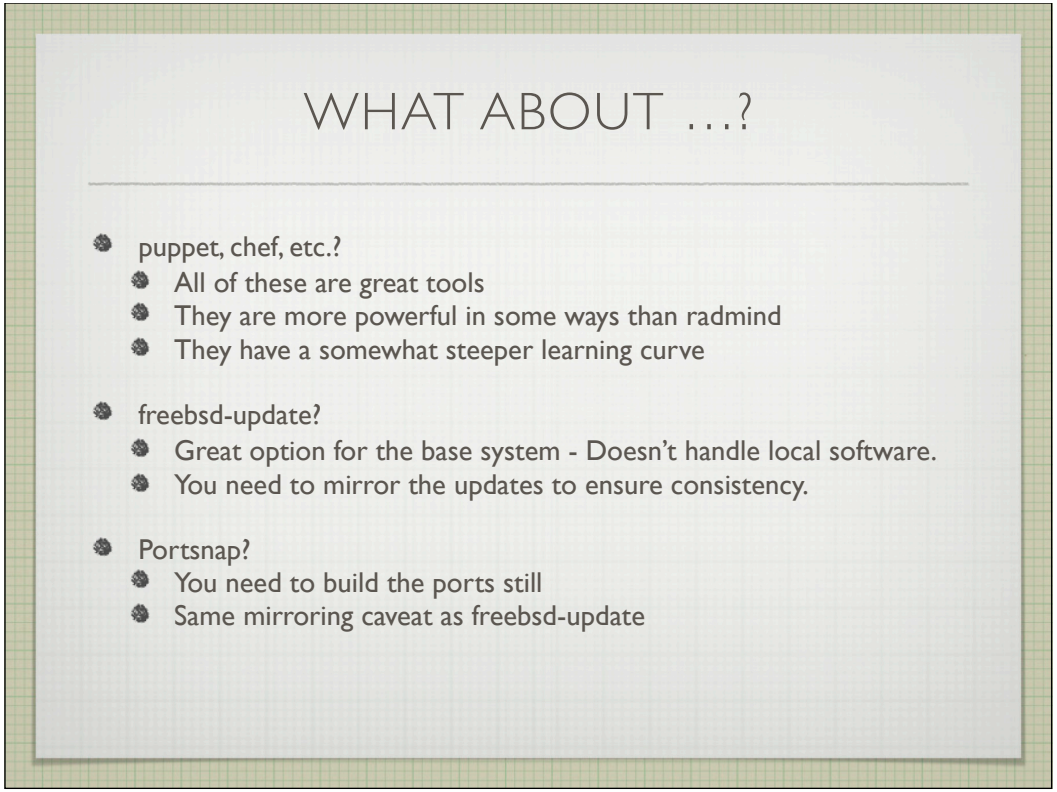
Tripwire
You’re probably running this already (and if not, why not?). radmind is essentially a centralized tripwire signature server: You download the file/signature list with ktcheck, then run fsdiff -A & anything that turns up has been modified somehow.



radmind has a few caveats

<u>Standardization</u>
All of your machines are running the same software, modulo some "extra" stuff.
This is the holy grail of managing an environment -- You know all the machines will behave roughly the same, so troubleshooting is made much easier.

<u>Tripwire</u>
You're probably running this already (and if not, why not?). radmind is essentially a centralized tripwire signature server: You download the file/signature list with ktcheck, then run fsdiff -A & anything that turns up has been modified somehow.



Some alternatives to consider

Puppet, Chef, etc…
These tools do the same thing as radmind -- They're all newer & more powerful (with the associated steeper learning curves), but if you're building a new system from the ground up you should consider them.

freebsd-update
Colin Percival created a great tool to distribute binary upgrades to the FreeBSD base system -- The problem is it only covers the base system. You need to run your own freebsd-update servers to make this work.

If all you're concerned about is the base system this is a great solution, but to ensure consistency in your environment you need to mirror the updates locally and install from your mirror - Otherwise you wind up with machine A starting its update a few seconds later & getting newer software than Machine B (update race).

portsnap
Portsnap is another great tool, but it only updates the tree: You still need to build and distribute/install those updates, which is where a tool like radmind comes in.

Also note that if you're portsnapping on every machine in your environment you have the same mirroring requirement as you did with freebsd-update.

SLIDES, SCRIPTS, ETC.

The slides from this presentation, including my notes, are available at http://www.bsd-box.net/~mikeg/talks/NYCBSDCon/2010/

Supporting materials, including my radmind build scripts, are available at the same URL.

More questions? Stuff I didn't cover?

email mikeg@bsd-box.net

Twitter/AIM/etc. voretaq7