provisioning

Grant Rettke

March 20, 2015

1 Reminders

- The means justify the ends
  - Makes entirety of the effort achievable, sensible, and valuable
- Be ready for surprises
  - You will learn new things every run
  - You will be surprised by things every run
  - You have to decide if it is an opportunity to better things or it is something to get upset about
    - Only you may decide
- Each top level section might be completed in entirety
  - Take brakes between them
  - Take over-night breaks
  - The whole thing takes a while
- Right-click to paste text into the terminal
  - If you Pretzel-v it, it doesn't work right
- Make scripts executable and run them
  - `sh` doesn't do what you want and you always forget that
- If one of your purchased apps doesn't show up in AppStore then just buy it again
  - The store will detect that you already own it and ask you to install it
  - Growl always has this problem
  - Used to reboot to fix it, but unreliable
- This whole thing only makes sense when you read each line
  - Partially because you forget after not doing it for a while
  - The order is critical
  - A lot of the steps are automated and only make sense after you understand the entire script
  - There are still a lot of manual steps though
  - For example automate app installs and manually configure them, **before** moving on to the next section
  - Everything must be completed before moving on to the next thing
- Disk image verification
  - Recipes exist to disable it
  - I don't use them; I want to know every time that all of my disk images are in a known good state
- Try running every program
  - Only way to identify failures
  - Only way to automate the correct way

  - Continue to find new issues and fixes
- Total transparency
  - I've run this 7 times already
  - Every time I run it I find new bugs and new improvements
  - Provisioning reveals just how much
    * One doesn't know about their OS
    * Things can go really well despite the fact we don't know how to do it right
- Automated provisioning is handled by kitchenplan
- Be thoughtful
  - This is not unique to this box, I just always forget
  - For example when I install emacs
    * I didn't think about using the main GNU release versus railwaycat's
      · That was dumb because I didn't think about the implications
    * First time, I installed all additioonal libraries
      · That was dumb because you should only install what you specifically need
    * Brew
      · It is very easy to use Brew
      · It will install a lot of stuff to make you happy
      · If you don't pay attention to what it does to make you happy then it will eventually make you unhappy
- Only run this setup with a reliable high-speed Internet connection
  - Ran this at off-site, and 40% of the downloads failed
- Once you get comfortable a couple of things happen
  - You try to paralellize stuff to save time and it is a big mistake which totally defeats the purpose of taking it slowly and easy because inevitably you will make mistakes
  - You forget that the means justify the ends; take it slowly and easy and perform the steps in that manner
  - You will form an opinion based upon experience rather than fantasy
    * Everyone is entitled to an opinion
    * Some opinions are more valuabe then others
    * Experience trumps fantasy **every** time
    * Take this dev-ops task as an opportunity to form your own person-preference
    * Valuable opinions come about as a result of a lot of good learning

2 Prerequisites, Manual

2.1 Scripts

- Log into BitBucket/GitHub using Safari
  - When I develop this system, do so out of GitHub and when I finished I publish it via GitHub.
- Download the hardware repo to get a current copy all the artifacts
- Keep the setup guide open both on a smartphone and on the computer to clearly track what is happening where
  - Seems like overkill but it is really simpler because it is easy to lose your place during reboots when OSX "forgets" the line that you were on
- Previously I copied the contents to a hard drive and this was useless because in order to provision a machine you require Internet access so nothing was made easier with this step

2.2 Image Disk

- Boot from EF4829
- Dismount C02M
- Erase C02M
  - You have the ability to specify the file system (FS) here
  - For this guide, it is not required, because CCC will restore a disk image that has a file system that is Mac OS Extended Journaled (MOSEJ)
  - The default is Mac OS Extended (not journaled)
  - I suspect though that if I don't specify it here, then at some point I will either copy or move this passage of text, and since it does not mention the desired file system (MOSEJ), that it will result in a documentation bug
  - When you do this, specify the FS and MOSEJ
- Mount 01
- Restore 01 to C02M
- Dismount 01
- Restore rescue partition on C02M
- Boot from C02M
- Dismount EF4829

2.3 Recovery Partition

- Pretzel-r on boot to enter it
- This makes sure that it works

2.3.1 Console Password (One Time Only)

- Set the firmware password

2.4 System Preferences

2.4.1 Keyboard

- Swapped caps lock and control

2.4.2 Displays

- Resolution: Scaled
  - More Space (1920x1200
- Arrangement
  - Mirror screens

2.5 Software Updates

- Installed software updates
  - Previously this was followed by manual reboot and that was needless
- On restart, check updates again by clicking on it to see if there are more and if there are then install them
  - There is a slight delay where the screen says that there are no updates *before* it refreshes and tellls you that it is again checking

2.6 Profile

Download the `~/.profile` and evaluate it. This may be a download, or just copy from the download of the repo. I am working through the best option, as both have seemed fine.

```
export PS1='\u@\h:\w> '
export HOMEBREW_NO_EMOJI=1
export INFOPATH='/usr/local/share/info:/usr/share/info'
export VAGRANT_DEFAULT_PROVIDER=vmware_fusion
source `brew --repository`/Library/Contributions/brew_bash_completion.sh
export EELIB="/Users/$(whoami)/EELIB"
export JAVA_HOME="/Library/Java/JavaVirtualMachines/jdk1.8.0_40.jdk/Contents/Home"
export MACTEX_BIN="/usr/local/texlive/2014/bin/x86_64-darwin"
export PATH=/usr/local/bin:$JAVA_HOME/bin:$MACTEX_BIN:$PATH
alias r='r --no-save'
alias R='r --no-save'
alias bbc='~/git/github-grettke/stathon/stathon.py ~/git/bitbucket-grettke/ | grep True'
alias gbc='~/git/github-grettke/stathon/stathon.py ~/git/github-grettke/ | grep True'
```

After setting up the system, just link it here:

```
cd ~/
rm .profile
ln -s ~/git/bitbucket-grettke/hardware/apple/macbookpro/c02m/.profile .profile
```

2.7 Install XCode

I want Xcode. I want the CLT. I want the IDE and more. I also want KitchenPlan. The installation order of these things was not obvious to me. I got it wrong 3 times. Now it works correctly. Install Xcode and then CLT and then everyone seems to be happy.

Go to the App Store and install it.

You must start Xcode and agree to its licensing. Afterwards, close it.

For reference, see where it lives.

```
xcode-select --print-path
```

```
/Applications/Xcode.app/Contents/Developer
```

Now install the CLT. Choose **Install**.

```
xcode-select --install
```

Note here that the Xcode path does not change. I don't understand this. What I think is happening is that Xcode is installed and that it is providing everything that is required and CLT piggy-backs on it.

```
xcode-select --print-path
```

```
/Applications/Xcode.app/Contents/Developer
```

2.8 Brew and BrewCask

Brew is here.

Run

```
brew doctor
```

BrewCask is here.

3 Git, Semi-Automated

After setting up this box, check out your stuff later before imaging.

3.1 Manual

Install DiffMerge.

Previously I used Meld. That worked fine. It was in Python. Brew installed it's own Python. I didn't like that. That is why I switched to DiffMerge. I didn't want to install Brew Python in addition to OSX Python. Personal preference.

My old setup had a problem. In Git I set up the external.diff to Meld. external.diff is only for command line tools. Meld is a GUI. That was wrong. I got a poor-user experience with git and vc-diff and Magit as a result.

This is the first BrewCask install, so it asks you for your password.

```
brew cask install diffmerge
```

Later in this setup, once the fonts are installed, set this font to DJSM 14. That is smaller than the Emacs default on my system. That is OK because I want to do side-by-side diffs in 50% of the screen. 14 is a fine balance between readability and fitting in that space.

3.2 Automated

Generate the key. There is no passphrase.

```
mkdir ~/.ssh
cd ~/.ssh
ssh-keygen -N '' -t rsa -C "gcr@wisdomandwonder.com" -f orion_gcr_rsa
```

Set permissions so that `ssh` will run.

```
chmod 600 ~/.ssh/orion_gcr_rsa
chmod 600 ~/.ssh/orion_gcr_rsa.pub
ssh-add -K ~/.ssh/orion_gcr_rsa
```

OSX creates this directory everywhere and it must be ignored.

```
echo .DS_Store >> ~/.gitignore_global
git config --global core.excludesfile ~/.gitignore_global
```

Add they key to Bitbucket and Github.

```
cat ~/.ssh/orion_gcr_rsa.pub | pbcopy
read -p "The new key is in your clipboard. Go and add this key to BitBucket and GitHub. When you are finished,
```

Set up ~/.ssh/config.

```
cat > ~/.ssh/config << EOF
Host github-grettke
     HostName github.com
     User git
     PreferredAuthentications publickey
     IdentityFile ~/.ssh/orion_gcr_rsa.pub
Host bitbucket-grettke
     HostName bitbucket.org
     User git
     PreferredAuthentications publickey
     IdentityFile ~/.ssh/orion_gcr_rsa.pub
EOF
```

Test each one out.

```
ssh -T github-grettke
read -p "Did it work? If not, fix it."
```

Separate them to allow for easier copy-pasting.

```
ssh -T bitbucket-grettke
read -p "Did it work? If not, fix it."
```

Set up my preferences.

```
git config --global user.name "Grant Rettke"
git config --global user.email gcr@wisdomandwonder.com
git config --global core.editor vi
git config --global color.ui true
git config --global core.autocrlf
git config --global alias.st status
git config --global alias.ci commit
git config --global merge.tool diffmerge
git config --global mergetool.diffmerge.cmd "/usr/local/bin/diffmerge --merge --result=\$MERGED \$LOCAL \$BASE
git config --global mergetool.keepBackup false
git config --global diff.tool diffmerge
git config --global difftool.diffmerge.cmd "/usr/local/bin/diffmerge \$LOCAL \$REMOTE"
```

Check out projects to get basic stuff working.

```
mkdir -p ~/git/bitbucket-grettke
mkdir -p ~/git/github-grettke
mkdir -p ~/git/github-anon
cd ~/git/bitbucket-grettke
git clone bitbucket-grettke:grettke/alec.git
git clone bitbucket-grettke:grettke/list.git
git clone bitbucket-grettke:grettke/hardware.git
git clone bitbucket-grettke:grettke/resume.git
git clone bitbucket-grettke:grettke/rule-engine-notes.git
cd ~/git/github-grettke
git clone github-grettke:grettke/home.git
git clone github-grettke:grettke/kitchenplan.gi
git clone github-grettke:grettke/stathon.git
git clone github-grettke:grettke/osx-provision.git
cd ~/git/github-anon
git clone git@github.com:tomislav/osx-terminal.app-colors-solarized.git
```

There are a lot of ways to manage libraries for Python. There are also many ways to manage Python installations. I am not considering any of them here. My approach is to use the base-box as a base. It needs to do certain things well. These directions are for my current approach.

Directions for installing Pip live here.

```
mkdir ~/tmp
cd ~/tmp
curl -O https://bootstrap.pypa.io/get-pip.py
sudo -H python ./get-pip.py
```

Install GitPython for Stathon.

```
sudo pip install GitPython==0.3.2.RC1
```

4 OSX GUI Config, Manual

Read the directions.

```
sudo gem install kitchenplan
```

Start the setup.

```
kitchenplan setup
```

When it asks, give my aliased URL so I can work right out of that directory.

```
github-grettke:grettke/kitchenplan.git
```

Provision it.

```
kitchenplan provision
```

Log in and out again to see changes. Previously I would reboot the machine, but that seems unnecessary.

The first two times I tested this, the desktop picture was set correctly. The 3rd time it was not. Also 4-6. I'm leaving that alone for now.

KitchenPlan may be run repeatedly. After completing various sections of this script I re-ran KitchenPlan. It worked correctly.

5 Emacs Environment Semi-Automated

This is what it takes to get an Emacs+Org development environment set up. Nearly everything is included but for LaTeX, which has an enormous download.

The way that I like to go through this is to:

- Perform the steps in the manual section
- Execute `emacs-run.sh` for the automated stuff
- Perform the manual configuration for everything that got installed
  - Directions are in `emacs-read.org`

5.1 System Software

5.1.1 Manual

1. Fonts

   ```
   Manually install fonts used for Emacs.

   Fonts that will be installed:
   - DejaVu
   - NotoSans
   - NotoSansSymbols
   - Quivira
   - Symbola

   They live in the font dir in the ALEC project.

   Steps to install them:
   - Extract them
   - Start Font Book
   - Create a new COLLECTION "Emacs"
   - Drag the top-level  directory into the collection
     - Font Book will search through it recursively for fonts and add them
   - Correct any resolution issues, which happen every time so far
   ```

2. Terminal

   Select the Solarized dark theme. It is checked out in Github-Anon. Go into preferences, choose it, and set it as the default.

   Set the font to DJSM 17.

3. Growl

   Install Growl via AppStore. Run it. Enable it to run on login.

5.1.2 Automated

Do the manual configuration of this stuff right away. The system is unusable without it. For some reason I find it easy to forget that.

1. XQuartz

   Why is this here? It used to be for Meld. Meld went away. It definitely *feels* right to be here. It is nice to have a X-Server. Stuff like the `dot` tool use X. I am staying true-enough here.

   This install asks for a password. I am done researching what asks for a password and what does not. Perfect world I would do that and break out the steps into things that ask for a password and things that to do not. In that perfect world, time does not exist. To do this in a world with time will take a lot of it.

   Asking for passwords appears to be an application-specific event and not based on order of Brew and BrewCask installs. Even if there is more to it, I am not researching it more now.

   ```
   brew cask install xquartz
   ```

2. Karabiner

   ```
   brew cask install karabiner

   Run it. Approve it. Configure it with the following. Look for the
   heading with the name below following "Modify".

   #+begin_src sh
   Modify "Change Return Key".
   Check:
   "Return to Control_L
   (+ When you type Return only, send Return)"
   #+end_src
   ```

3. Spectacle

   ```
   brew cask install spectacle

   Run it. Approve it. Set it to run at login.
   ```

4. Bartender

   ```
   brew cask install bartender

   - GENERAL
     - Launch Bartender at login: yes.
     - At bartender launch: show bartender bar: NO.
     - Bartender bar: autohides, YES.
   - Menu bar icon highlight
     - Show when bartender bar is open: YES.
     - Bartender menu baricon:
   - ADVANCED
     - Bartender menu bar icon: visible YES.
     - Clicking on bartender will: open bartender bar.
     - For everything moved into the gutter
     - I made it display in the main for 5m.

   - Ordered them (reorder by holding Pretzel and dragging them) (do this after installing everything)
     - little snitch
       - Can't move this one at all anyway
     - Date/Time
     - Volume
     - Keychain Access
     - Battery
     - Bluetooth
     - Wifi
     - Vagrant Manager
   ```

```
    - Dropbox
    - Fast User Switching Menu
    - Applications Menu Settings
    - Notification center
    - Spotlight
    - Bartender

    - Hide
      - Growl
      - Karabiner
      - Spectacle
      - Carbon Copy Cloner
      - AirPort Display
      - Time Machine (completely)
      - Google Chrome (Hangouts)
      - Microsoft Remote Desktop

  Arrangement-algorithm is most activity or cared about on the outside and less so
  in the middle.
```

5. Growlnotify

   ```
   brew cask install growlnotify
   ```

6. CCrypt

   ```
   brew install ccrypt
   ```

7. Aspell

   ```
   brew install aspell -all
   ```

8. Java

   At this point, test out Java and Langtool. The full Org stuff requires more configuration, like Ditaa. This assumes that you've done the Emacs final config stuff based on warnings at startup of Emacs; not on directions on this document.

   ```
   brew cask install java
   ```

5.2 Emacs

I quit using railwaycat's emacs macport. It was really nice. One night I reprovisionined my machine. railwaycat's repo was down. I learned that it was a one-man show. I never thought about it. It means that it is not supported well. Things that are supported well are likely to be available. I switched to the GNU Emacs app. It has those desirable traits.

ImageMagic is required for the Emacs install, so install it first.

```
brew install imagemagick --with-fftw --with-fontconfig --with-webp --with-x11
```

With all of Emacs prerequisites satisfied, install it now.

```
brew install emacs --cocoa --with-gnutls --with-imagemagick
brew linkapps
```

The installer lists caveats. Do I need to know of these for each installer???

Now link my .emacs.el and try tangling.

```
ln -s ~/git/bitbucket-grettke/alec/.emacs.el ~/.emacs.el
ln -s ~/git/bitbucket-grettke/alec/.aspell.en.pws ~/.aspell.en.pws
ln -s ~/git/bitbucket-grettke/alec/.aspell.en.prepl ~/.aspell.en.prepl
```

The eshell config lives under .emacs.d. It doesn't exist until Emacs is run. This setup creates a link to the real .emacs.d. Create the directory first and then link it.

```
mkdir ~/.emacs.d
ln -s ~/git/bitbucket-grettke/alec/eshell/ ~/.emacs.d/eshell
```

Start Emacs. Remember any Emacs fixes. Everything should work.

6 Productivity Semi-Automated

To be immediately productive with the most common use cases, I need all of these apps.
Use cases:

- Communications
- Productivity
- Word processing
- Productivity is pretty broadly defined here. It means everything.

The installation process is similar to Emacs up above:

- Install the software automatically
- Configure it manually
- Manually run it and make sure that it is happy
  - You will find new stuff everytime you do this and you will get better at dev-ops and automate as much of it as you can and want to do

I am curious about how best to automate this portion of the installation. My approaches were in order:

- Just let it run and try stuff later
  - Allows for installation failures which I don't like, it wastes time discovering the issue and resolving it
- Verify that each app starts
  - This is a fine approach. I would like to know at installation time though if there is a failure. Reading the transcript would be a good approach. Perhaps I should capture it, too.
  - Works easily for command line stuff via a script. GUI stuff is slower when I run it through Spotlight. Perhaps it is worth tracking down the installation directories of everything and starting them up via `open`?
    - Is this level of reproducibility required? Might need to quantify this.

Read through the installation messages at least once. You will learn things and then you will improve the installation process.

6.1 Files

I always end up keeping these temp files. Yes, temp files. They don't live in source control.

```
touch ~/tmp/post.org
touch ~/tmp/scratch.org
touch ~/tmp/todo.org
```

6.2 Dropbox

This one is first so that I'll have easy access to licenses.

```
brew cask install dropbox
```

```
Sign in.
In Login Items, start Dropbox.
When on highspeed, just sync everything after copying locally of course.
When on lowspeed, just sync required.
```

6.3 Little Snitch

```
brew cask install little-snitch

As of writing, the installer just gets downloaded and you need to run it like
this:

#+begin_src sh
open /opt/homebrew-cask/Caskroom/little-snitch/3.5.1/Little Snitch Installer.app/
#+end_src

Typical usage is to grant every program full access for any connection, forever,
until there is a reason not to do so.

- Gen
  - Show inactive warning
  - Silent mode: no
  - Show status in menu bar.
- Alert:
  - Detail level: Show full details
  - No: Confirm automatically
  - NO: Confirm with return and escape.
- Monitor:
  - Keyboard shortcut: On
  - Show network activity in menu bar.
    - Show data rates numerically. Monochrome.
  - Show auto when mouse enters. Hide in 2s.
- APS
  - No: Enable automatic profile switching
  - Yes: Save geolocation of networks.
- Security
  - Allow rules and profile edit.
  - Allow profile switch.
  - Allow preference editing
  - Respect privacy.
- Advanced
  - Approve rules automatically.
Update
  - Automatically check for updates daily
```

6.4 Chrome

```
brew cask install google-chrome

Sign into Chrome. Let the settings sync. All the JS disabling stuff needs
you to approve it. It is irritating. You always forget to approve it and make
it worse. Disable ScriptSafe right away. Turn it on as needed.
```

6.5 Firefox

```
brew cask install firefox
```

Sometimes this install fails. I checked the file download. The file exists. The name hasn't changed. Did a manual install instead.

```
Install the standard plugins: NoScript, Blur.
```

6.6 Filezilla

```
brew cask install filezilla

Set up Filezilla for WnW.
```

6.7 Skype

```
brew cask install skype
```

Got a surprise today. After doing 7 runs, Skype was installed but it was not sucessfully linked into /Application. That is surprising. It is installed under the BrewCask (cask room) location:

```
/opt/homebrew/Caskroom/skype/latest/Skype
```

The the link to Applications didn't exist so Skype was not added to Login Items. I manually added it. I clicked +. This Mac. Searched for "Skype". Then I chose it.

```
Log in automatically. Don't grant access to contacts.
Preferences General. Never set my status away. Don't show birthday notifications. Don't always keep Skype up to
Messaing: Don't use larg emotions. Use compact chat style. Don't show when I am typing.
Notifications: Do not notify when I sign in or out, and others, and when they become
available do nothing.
Advanced: Display technical call information yes. Do not collect
call information.
Login Items, add Skype in here.
```

6.8 SourceTree

```
brew cask install sourcetree
```

```
Log into SourceTree.
Let it scan the BitBucket folder to add those projects to SourceTree.
Preferences.
General: Disallow SourceTree from modifying yoru global Git config file
Diff: Set font to DJSM 14.
Manually:
- Go through the licensing process
- Old app just let you type it into the GUI, not seeing it here now
```

6.9 Racket

Need this for resume stuff. At this point at least Racket is installed. Here I found that Pandoc was not set up right. After everything is done, including manually installing MacTeX, I could build my resume here.

```
brew cask install racket
```

```
racket --version
```

6.10 Freemind

I use mindmaps.

```
brew cask install freemind
```

```
Run it.
```

6.11 VIM & MacVIM

```
brew install vim
```

```
brew cask install macvim
```

```
Copy over my .vimrc, once one exists.
Run it.
```

6.12 Virtualbox

```
brew cask install virtualbox
```

Sometimes this install succeeds but the application does not appear to be installed. Not sure if this is a Spotlight/FS thing or what.

```
Boot up at least one Vagrant box using Virtualbox to make sure it is happy.
The directions are with Vagrant.
```

6.13 VMWare Fusion

```
brew cask install vmware-fusion
```

Sometimes this fails, just like with VirtualBox. Perhaps it is a network speed thing?

```
License it.
Boot up at least one Vagrant box using VMWare Fusion to make sure it is happy.
The directions are with Vagrant.
```

6.14 Libre Office

```
brew cask install libreoffice
```

This failed on the slow network.

```
Run it.
```

6.15 Kindle

```
brew cask install kindle
```

```
Sign in. Make sure it works. Download everything so it is there.
You can see files downloaded versus avalable to be sure. Or not.
The upside is reading without Internet connectivity.
```

6.16 VLC

```
brew cask install vlc
```

```
Run it.
```

6.17 Programs That Don't Need Any Configuration

6.17.1 Part A

```
Remember to run these. Just do a version check or something.
```

```
brew install smlnj
brew install cvs
brew install bzr
brew install hg
brew install graphviz
brew install tree
brew install archey
brew install figlet
brew install pandoc
brew install gforth
brew install ffmpeg --with-fdk-aac --with-ffplay --with-freetype --with-frei0r --with-libass --with-libvo-aacer
brew linkapps
brew install povray --with-openexr
brew linkapps
```

6.18 Stuff That Asks For My Password

6.18.1 R

This is the first BrewCask install where I saw a checksum error:

```
==> Caveats
Cask r installs files under "/usr/local".  The presence of such
files can cause warnings when running "brew doctor", which is considered
to be a bug in homebrew-cask.

==> Satisfying dependencies
complete
==> Downloading http://cran.rstudio.com/bin/macosx/R-3.1.3-mavericks.pkg
######################################################################## 100.0%
==> Note: running "brew update" may fix sha256 checksum errors
Error: sha256 mismatch
Expected: bd150f488c36e3d793febd3b7f619c076fc3bccfe673592af3134c32118d1c5e
Actual: 28445419c73b03dd3e0e1199114e23c83e56a5140f8c43f37b63cb550dc0eba7
File: /Library/Caches/Homebrew/r-3.1.3.pkg
To retry an incomplete download, remove the file above.
```

The software was downloaded. Its checksum was wrong. Now what? This isn't the end of the world. I am glad it reported it. I am not investigating the mismatch. I suspect that there is a newer release and the formula wasn't updated. That is fine. I will delete that download and install it manually. This is a story I want to capture. Things aren't always perfect. There are so many moving parts that this is bound to happen. Patches are usually always welcomed, too.

To handle this manual install, I read the OSX page for R. They explain that we ought to compare the checksum of the download and show how. The current hashes are:

```
MD5-hash: 2f263bbb394361b5c3dd0d882d1d2e70
SHA1-hash: 200349fbcfd14b8b4769b52340164dd728c3995c
(ca. 68MB)
```

You may see the download size in MiB via:

```
cd ~/Downloads
du -h R-3.1.3-mavericks.pkg

 68M    R-3.1.3-mavericks.pkg
```

You may check the checksum via:

```
cd ~/Downloads
md5 R-3.1.3-mavericks.pkg

MD5 (R-3.1.3-mavericks.pkg) = 2f263bbb394361b5c3dd0d882d1d2e70
```

Install the software.

```
brew cask install r
ln -s ~/git/bitbucket-grettke/alec/.Renviron ~/.Renviron
ln -s ~/git/bitbucket-grettke/alec/.Rprofile ~/.Rprofile
ln -s ~/git/bitbucket-grettke/alec/.Rinstall ~/.Rinstall
rm -rf ~/.Rpackages
mkdir ~/.Rpackages
```

Explain what to do with it.

```
I haven't used R in a long time but I want to get it set up right again.
Link my configs from ~/ into where they live now.
This installer asks for my password, which halts the install. How will I deal
with this?
There is a =.Rinstall= file in my setup to get all of the packages installed. Be s
sure to run that, too. All of my notes live in R.org already.
```

1. R Studio

   ```
   brew cask install rstudio

   This doesn't need any config I think.
   Run it.
   ```

6.18.2 Vagrant

```
brew cask install vagrant

Install the VMWare provider for Vagrant and then license it.

Remember to [[http://docs.vagrantup.com/v2/other/debugging.html][log]] appropriately.

It goes something like this:

#+begin_src sh
vagrant plugin install vagrant-vmware-fusion
mkdir ~/.vagrant
cd ~/.vagrant
# copy that license file in there
vagrant plugin license vagrant-vmware-fusion license.lic
#+end_src

Test out VirtualBox box:

#+begin_src sh
cd ~/tmp
vagrant init hashicorp/precise32
vagrant up --provider virtualbox
#+end_src

The first time I tried this, it failed. I restarted OSX. Then it worked.

Test out VMWare:

#+begin_src sh
cd ~/tmp
vagrant init chef/ubuntu-14.04
vagrant up
#+end_src
```

1. Vagrant Manager

   ```
   brew cask install vagrant-manager
   ```

   Tonight I had a surprise. Vagrant reported a failure. It was trying to copy a machine into a non-existent directory ~/.vagrant. That is where I keep the VMWare Vagrant license. That was surprising. Support has see this before when Vagrant Manager is running. I closed it and tried again and it worked fine. The value of Vagrant Manager is that it makes visible machines that you have running. Support explained that Vagrant Manager might be doing it. It was. He opened a ticket. I added to it.

   ```
   - Terminal Preference: Terminal
   - Status Bar Icon Theme: Clean
   - Launch at login: Yes
   - Following settings: No
   - Refresh every: 5 seconds
   - Following settings: No
   - Allowed Updates: stable
   - Send anonymous profile data: Yes
   ```

7 Productivity, Manual

7.1 By Hand

7.1.1 Entropy

This is a download and manual install.

7.1.2 Guitar Pro

Here. Sign into MySongBook.

7.1.3 Cepestral Callie & David

Here.

You might need to log out and in again to make the "Cepstral Voices" appear in System Preferences.

License them. Test them.

Set Dictation & Text to Speech to: Callie

7.1.4 Microsoft Office

This is in BrewCask, but the download is waste of time. I have it locally.

I keep going back and forth on this one. Is it worth saving 15 minutes having to install it myself?

Start it. It asks for your name. It asks about updates and stuff. Get them all. Start Outlook. Enter in new license for that.

When activation occurs, the app always locks up then I quit it and it restarts after sending an error report.

Don't let it access Contacts.

When it checks for updates, tell it to check daily.

7.1.5 MacTex & GnuPlot

This is 2.4 GiB. That is not much. If you get the US mirror it is 20 minutes.

That is if you are lucky.

That makes the automated installer very slow. I just don't like it.

I will deal with it by installing it manually here.

At this point, definitely I can build my resume here.

Once that is installed, install gnuplot:

```
brew install gnuplot --with-latex --with-pdflib-lite --with-tests --with-x11
```

Build my resume at this point, too.

7.2 App Store Installs

- First check purchases, then the store
- Microsoft Remote Desktop
  - Don't open the session full-screen.
- Text2Speech PRO
  - Set default voice to Callie
- PixelMator

8 User Options, Manual

- Not really a user option, but open Keychain Access
  - Preferences
  - YES: Show keychain status in menu bar
    - This gives you a lock-screen GUI command

8.1 Desktop & Screen Saver

- Desktop

The automatic setting works intermittently. If it failed, set it manually. The image is here. I downloaded it, too.

```
cd ~/Pictures/
curl -O http://www.wisdomandwonder.com/wordpress/wp-content/uploads/2015/02/M101-ORG.jpg
cd ~/
```

- Screen Saver
  - Screen saver: Flurry. After 10 minutes.
  - Hot corner
    - Top left: Start screen saver
    - Top right: Put display to sleep
    - Bottom left: Disable screen saver

8.2 Energy Saver

- Energy Saver, Power Adapter
  - Automatic graphics switching: yes
  - Turn display off after minutes: 15
  - Prevent computer from sleeping automatically when the display is off: yes
  - Put hard disks to sleep when possible: yes
  - Wake for network access: yes
  - Enable power nap when plugged in: yes
  - Show battery status in menu bar: yes

8.3 Users & Groups → Login Options

- Automatic login: off
- Display login window as: Name and password
- Show the Sleep, Restart, and Shut Down buttons: yes
- Show input menu in login window: no
- Show password hints: no.
- Show fast user switching menu as: icon
- Use VoiceOver in the login window: no

8.4 Security & Privacy

8.4.1 General

- User: Require password _ after sleep or screen saver begins: yes, after 15 minutes
- Disable automatic login: yes
- Advanced → Require an administrator password access system-wide preferences: yes

8.4.2 Firewall

- Turn it on.
- Block all incoming connections: no
- Automatically allow signed software to receive connections: no
- Enable stealth mode: yes

8.4.3 HOSTS

Update it appropriately.

9 Full System Backups

Doing provisioning using Kitchenplan and Brew and BrewCask still takes at least 6 hours. You still need to perform manual steps. For me, it is unavoidable. This is painful. This is a waste of 6 hours. It is the waste of your evening. It is a waste of your weekend.

My new plan is to get the box provisioned and image it there. I want to minimize provisioning ever again. The opportunity here is that you learn new things each time. That is cool. The problem is that your goal is not to learn new things every time. It is a decision, and it needs to be thoughtfully made. This evidence is valuable. It has helped me understand my decision to work in virtuals here forward.

To image a full system:

- Note the base configuration, `01`
- Get system and application updates
- Trim contents of included-folders
  - Most of my stuff will never be in them anyway
  - More like DropBox and Git
    - Keep Git checkouts. Simpler. Update will get new stuff.
  - `/Users/gcr/Documents`
  - Empty the trash
  - Vagrant box definitions will exist under source control, but the individual boxes will only ever live in the storage locations configured for the provider
  - Run `bc` and `gc`
- Disk utility
  - Verify disk permissions
    - Logs a lot of info
    - Unsure if any of it is useful
  - Repair disk permissions if necessary
    - At the very bottom will log "Permissions repair complete"
  - Verify disk
  - Repair disk if necessary
- For the clone exclude
  - `/Users/gcr/Downloads`
  - `/Users/gcr/Dropbox`
  - `/Users/gcr/tmp`
  - `/Users/gcr/.vagrant.d/boxes`
    - `http://docs-v1.vagrantup.com/v1/docs/boxes.html`
    - How this works is that when CCC cloned that disk image back to the machine, this directory was present, and it was empty. I was curious about whether the directory would be present or not and happy to see it was.
    - After upping a box successfully, I verified that the base box was installed here as expected
  - `/VirtualBox VMs`
    - VirtualBox stores its VMS here by default
    - When you start making changes to a Vagrant box, they are saved here
    - For example I booted up the Vagrant box from "Getting started", created a hello world file, logged out and halted it, and this directory was `tmp_default_1426807980079_94539` was created in there and it contained all of the .vbox stuff for that machine.

        - Verified that `vagrant destroy` deletes that directory
    - `/Users/gcr/Documents/Virtual Machines`
        - `http://kb.vmware.com/selfservice/search.do?cmd=displayKC&docType=kc&docTypeID=DT_KB_1_1&externalId=2056798`
        - This seems to be the directory where VMWare creates machines when you do so through the GUI
    - When Vagrant VMWare Fusion creates machines it creates them inside of a `.vagrant` directory where the Vagrantfile lives.
        - The path is `<directory where the Vagrant file lives>/.vagrant/machines/default/vmware_fusion`
        - Found this info here and verified it on my machine
        - Verified that `vagrant destroy` deletes that directory
- Make sure that this configuration, including Kitchenplan, is tagged
- Note that tag here:
    - Machine name
    - Built from image
    - Provisioning Tag
    - Timestamp
    - c02M-01-v1.7-2015-03-17T18:58:40-0500
- On successful creation of the new image
    - Erase the machine and clone the new image to it
        - Address any image restoration issues now
    - Backup this new image to Stargate
    - Copy the Dropbox folder