

# Jenkins CI for MacDevOps

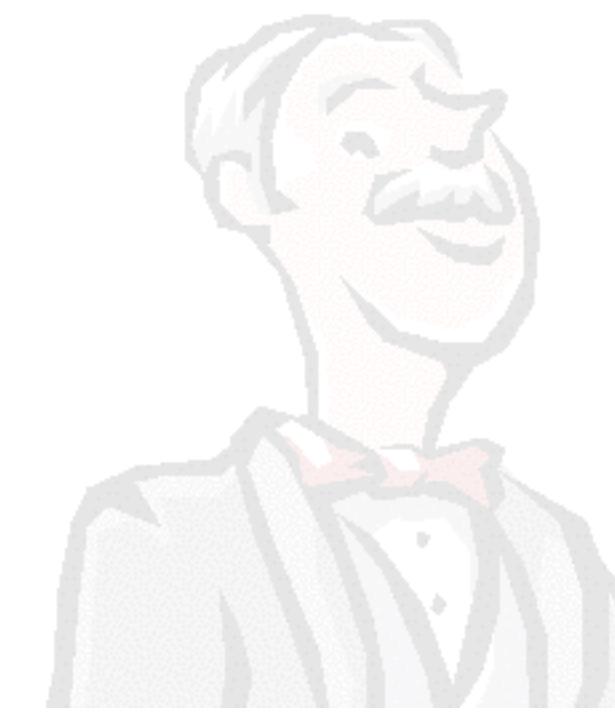
Tim Sutton

Concordia University, Faculty of Fine Arts  
Montreal

A story



svn update  
../configure  
make



Hudson

Hudson » Dashboard

All Dashboard

S W Job Last Success Last Failure Last Duration

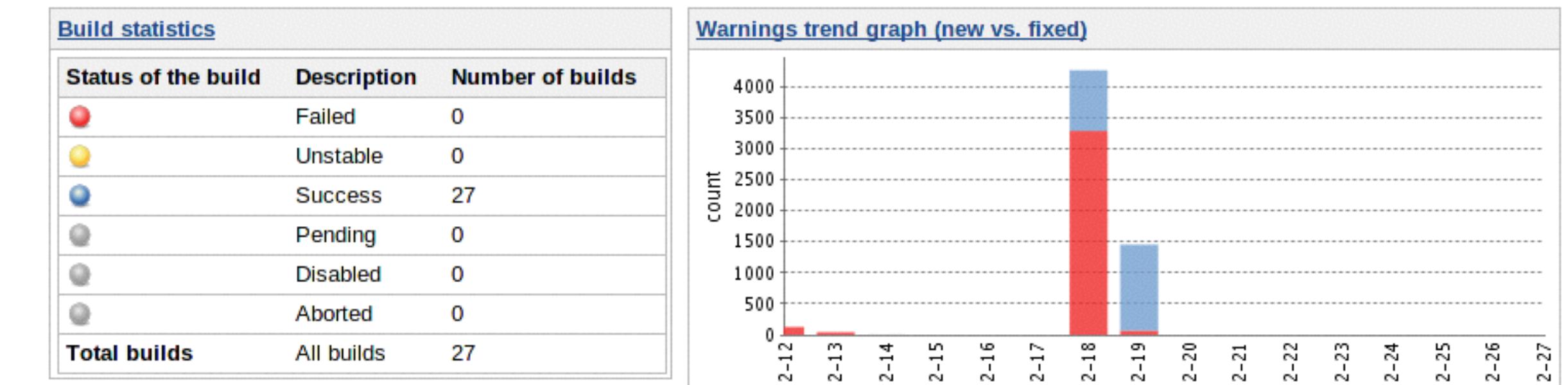
		<a href="#">Semantic Assistants</a>	6 hr 58 min (#30)	N/A	5 min 47 sec
--	--	-------------------------------------	-------------------	-----	--------------

Icon: [S](#) [M](#) [L](#)

Legend for all for failures for just latest builds

Warnings per project

Job	Checkstyle	Duplicate Code	FindBugs	PMD	Open Tasks	Compiler Warnings	Total
Semantic Assistants	<a href="#">2537</a>	<a href="#">20</a>	<a href="#">125</a>	<a href="#">271</a>	<a href="#">9</a>	<a href="#">83</a>	<a href="#">3045</a>
<b>Total</b>	<b>2537</b>	<b>20</b>	<b>125</b>	<b>271</b>	<b>9</b>	<b>83</b>	<b>3045</b>

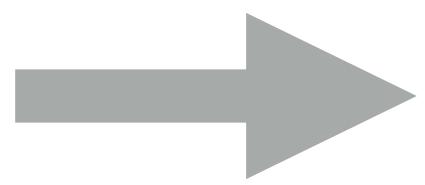


Jobs Grid

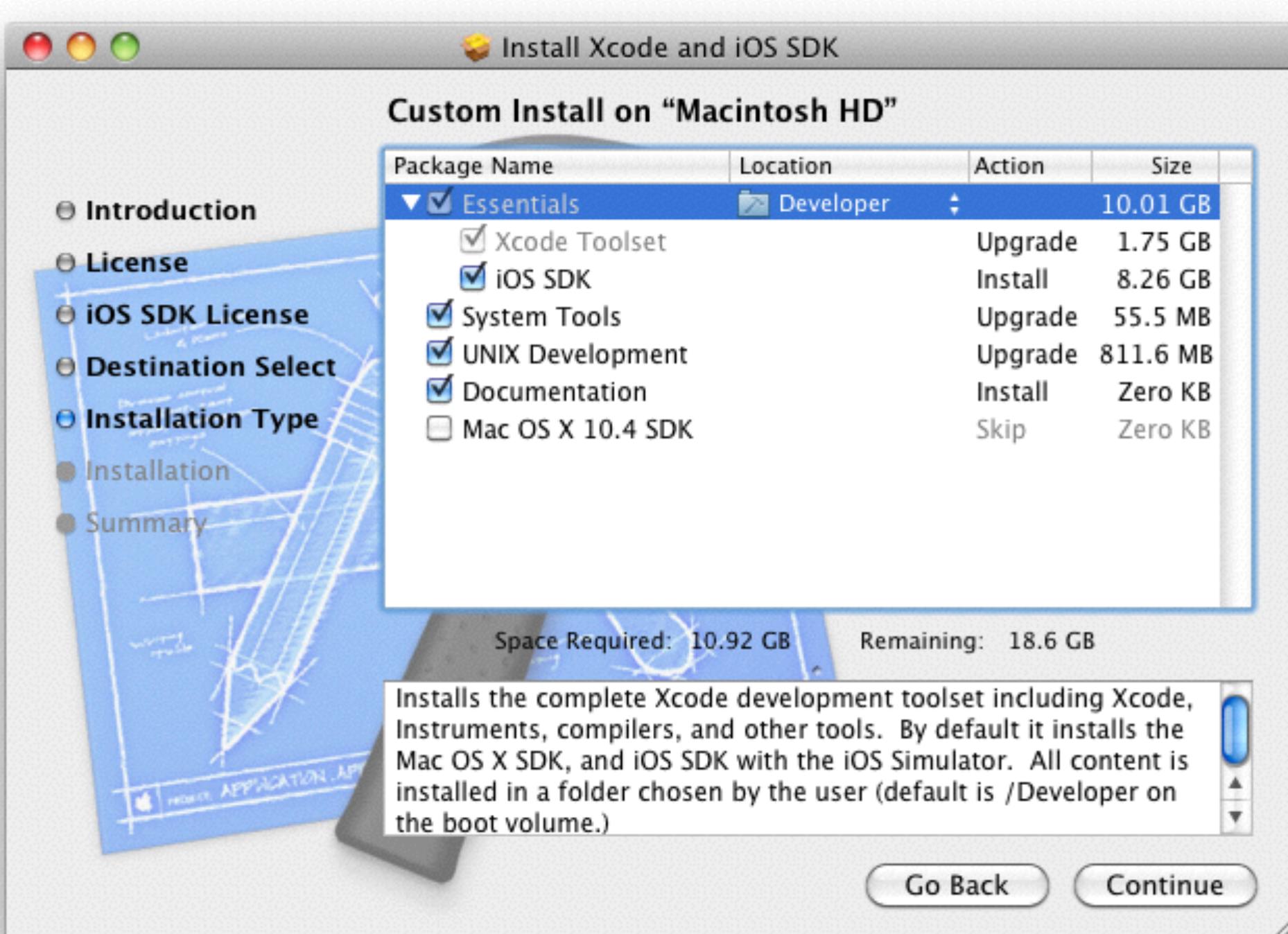
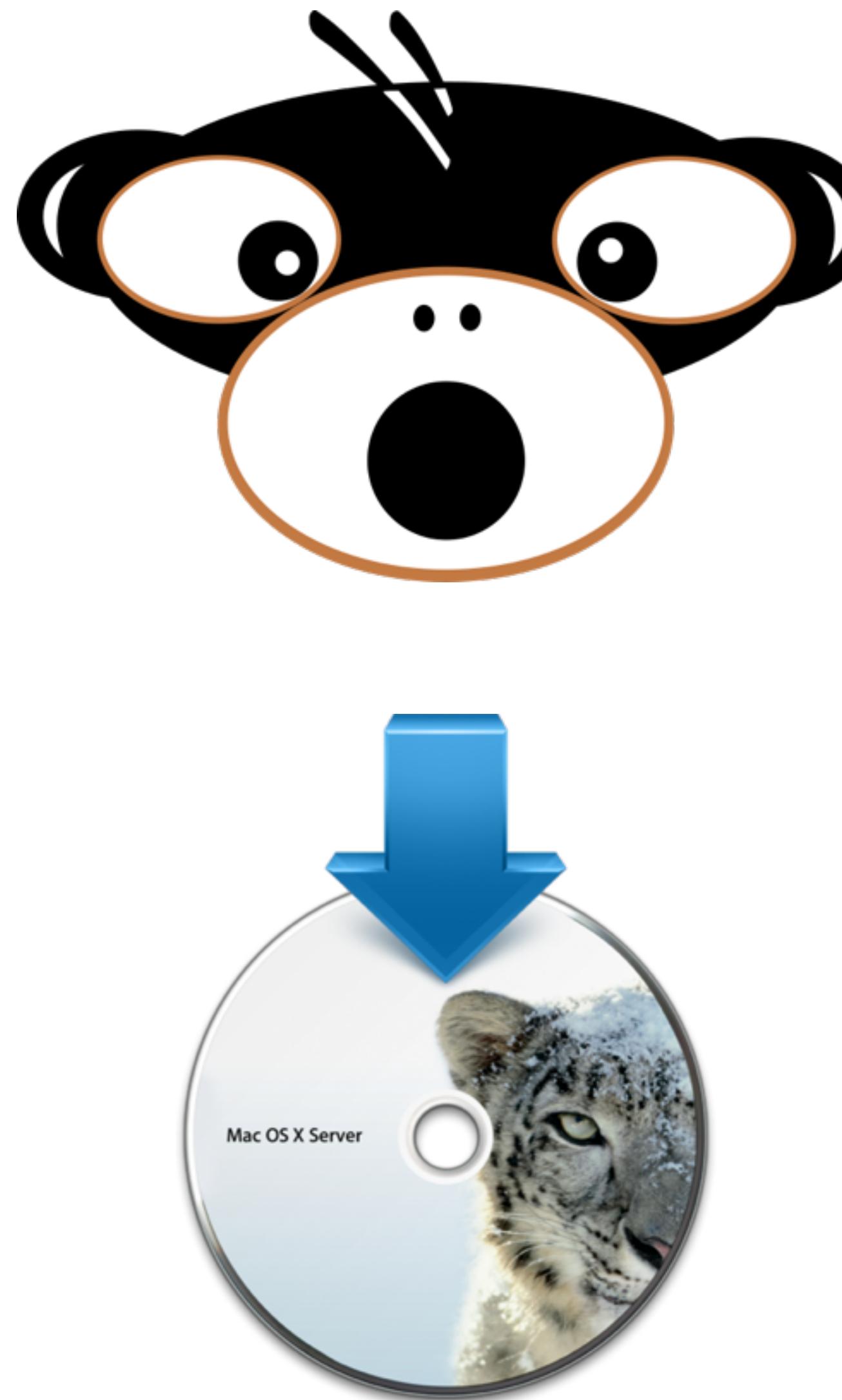
Semantic Assistants
---------------------



Hudson



Jenkins



```
git pull  
cd code/tools  
../make_munki_mpkg.sh
```

# munkibuilds.org



Munki auto-builds

*These are not official, stable releases, these are automated builds from the master Git branch for testing. Builds are performed on OS X 10.11.5, Xcode 7.3, using the build script [provided in the project repository](#).*

There is a download-and-install script available to get and install the tools in one step. Run the following command in a terminal session (once you've audited the script yourself):

```
curl https://munkibuilds.org/latest2.sh | sh
```

If you use [AutoPkg](#), there are also [recipes](#) to get the latest builds.

Munki is licensed under the [Apache 2.0 license](#). Visit the Munki [project site](#) on GitHub.

Maintained by [Tim Sutton](#)

Name	Last modified	Size	Description
<a href="#">munkitools2-latest.pkg</a>	2016-06-08 11:44	3.3M	
<a href="#">_branches/</a>	2016-06-08 11:44	-	
<a href="#">2.7.1.2764/</a>	2016-06-08 11:44	-	
<a href="#">2.7.0.2763/</a>	2016-06-02 17:13	-	
<a href="#">2.7.0.2762/</a>	2016-05-24 17:37	-	
<a href="#">2.7.0.2761/</a>	2016-05-23 21:42	-	
<a href="#">2.7.0.2757/</a>	2016-05-20 09:17	-	
<a href="#">2.7.0.2756/</a>	2016-05-15 21:57	-	
<a href="#">2.7.0.2755/</a>	2016-05-13 19:37	-	
<a href="#">2.7.0.2754/</a>	2016-05-11 13:22	-	
<a href="#">2.7.0.2753/</a>	2016-05-02 12:02	-	
<a href="#">2.7.0.2752/</a>	2016-04-28 12:52	-	

# Continuous Integration

# Continuous Integration

- Software Development
  - Run unit, integration, performance tests
  - Fix code style, handle compiler warnings
  - Define a single trusted build environment (no “works on my machine”)
  - Automate merging or pushing code if conditions pass
  - Orchestrate complex build tasks, shared dependencies

# Continuous Integration

- Operations
  - Run tests on configuration management tooling
  - A “build” == any long-running task ending in a product, and which needs a specific environment set up
  - Move “magic” you’ve internalized into shared code
  - Delegate access to others, remove bus factor

For example

# Auto-build your MacAdmin tools

Jenkins > Imagr >

ENABLE AUTO REFRESH

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Workspace](#)

[Build Now](#)

[Delete Project](#)

[Configure](#)

[Rebuild Last](#)

[Git Polling Log](#)

[Move](#)

[Project Imagr](#)

[add description](#)

[Disable Project](#)

[Workspace](#)

[Last Successful Artifacts](#)

[Imagr-1.2.1-403.dmg](#) 7.41 MB [view](#)

[Recent Changes](#)

**Permalinks**

- [Last build \(#144\), 3 days 11 hr ago](#)
- [Last stable build \(#144\), 3 days 11 hr ago](#)
- [Last successful build \(#144\), 3 days 11 hr ago](#)
- [Last failed build \(#30\), 1 yr 1 mo ago](#)
- [Last unsuccessful build \(#32\), 1 yr 1 mo ago](#)
- [Last completed build \(#144\), 3 days 11 hr ago](#)

Build History trend =  
find   
[#144](#) Jun 16, 2016 2:53 AM  
[#143](#) Jun 15, 2016 5:08 PM  
[#142](#) Jun 15, 2016 4:33 PM

## Project CDA-Base-Image

This build requires parameters:

OSX\_VERS  OS X version

OSX\_BUILD  OS X build version

BUILD\_NODE  master  elcap  mavericks  tim-win7-vm

SKIP\_ESD\_DOWNLOAD

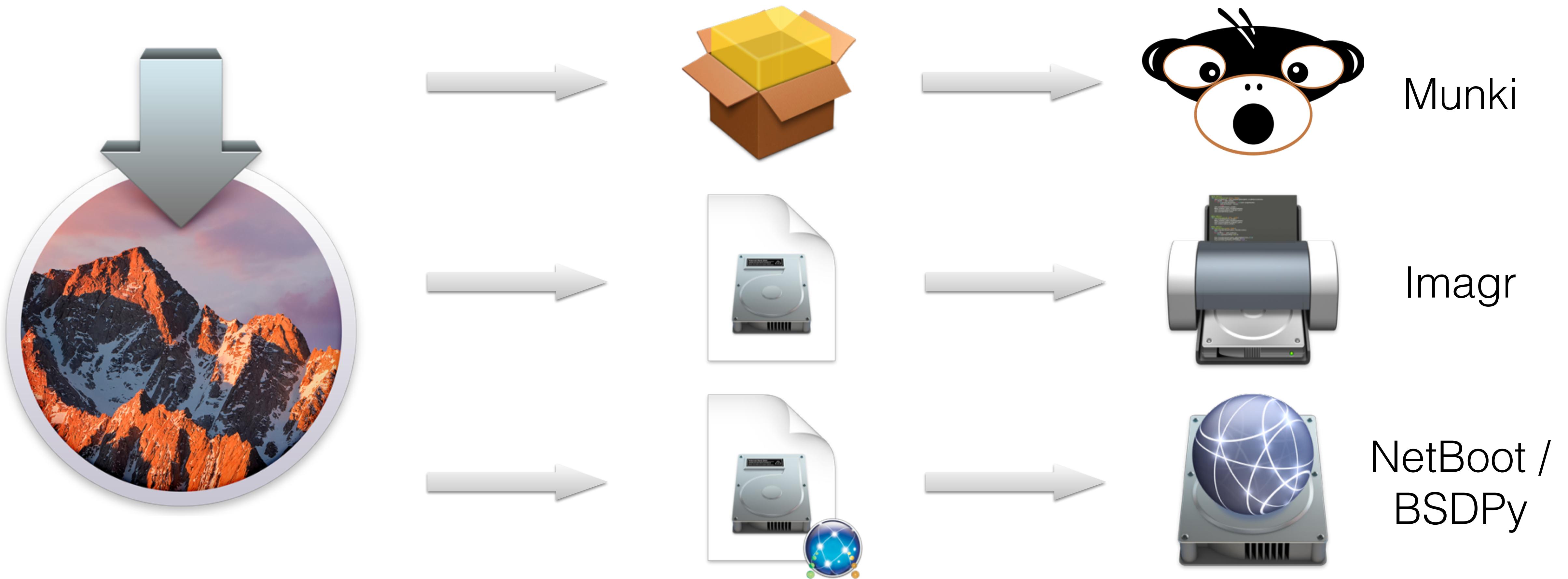
SKIP\_IMAGE\_BUILD

CLEAR\_SCRATCH\_DIRS\_FIRST

AUTODMG\_VERSION  Version of AutoDMG to use for the build. build-latest: compile from Git master, installed: use version installed in /Applications.

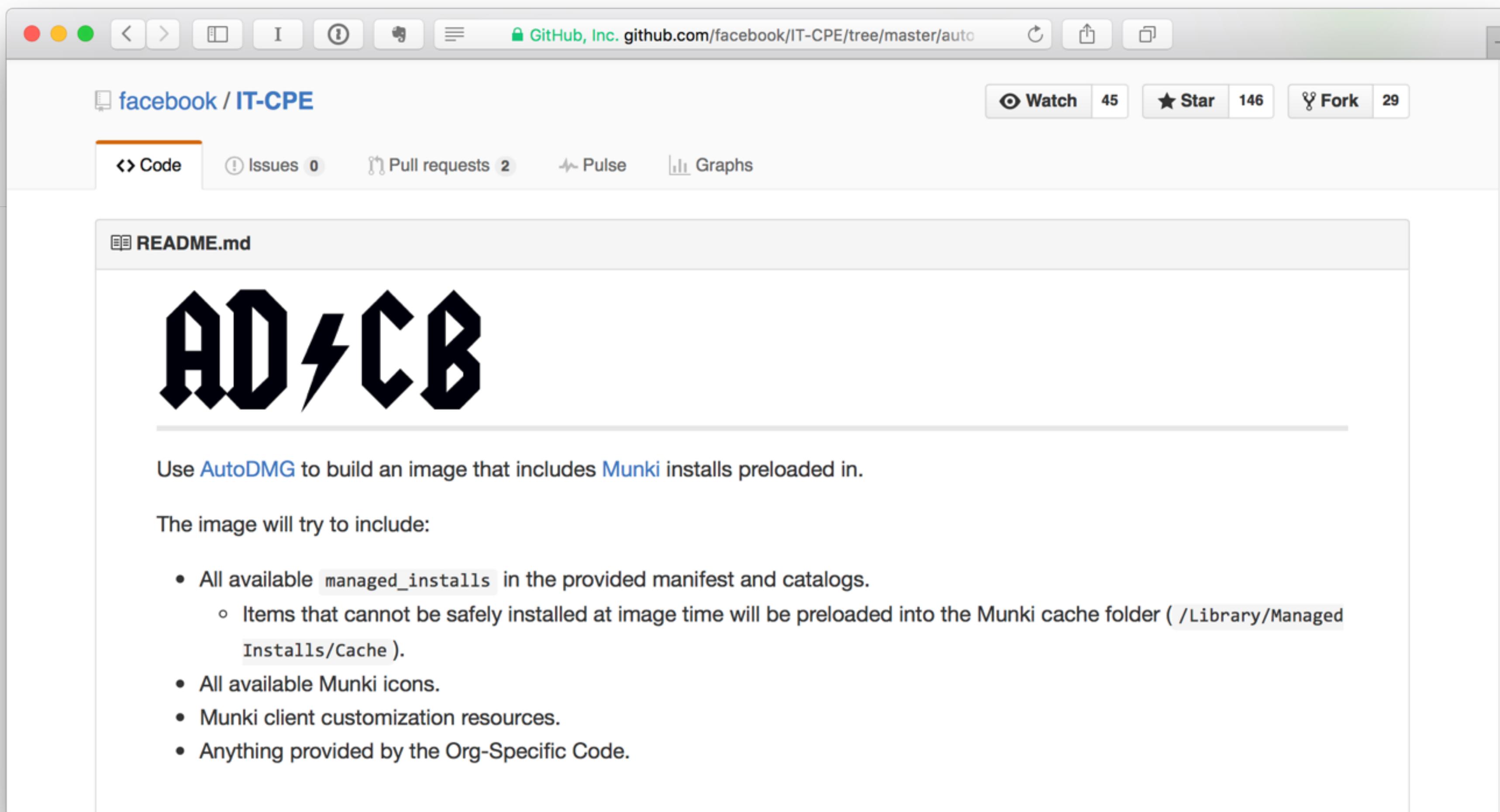
[Build](#)

# Single input, multiple builds



# Single input, multiple builds

AutoDMG Cache Builder (Nick McSpadden)



# Monitor external code

A screenshot of a GitHub repository page for `timsutton / DeployStudioDiffs`. The page includes a navigation bar with icons for back, forward, search, and refresh, and a URL bar showing `GitHub, Inc. github.com/timsutton/DeployStudioDiffs`. Below the header are buttons for `Watch` (6), `Star` (12), and `Fork` (1). The main content area shows tabs for `Code`, `Issues 1`, `Pull requests 0`, `Pulse`, and `Graphs`. A section titled `What changed?` displays summary statistics: `39 commits`, `1 branch`, `34 releases`, and `1 contributor`. Below this are buttons for `Branch: master`, `New pull request`, `Find file`, and `Clone or download`. The commit history table lists four commits by `timsutton` from April 5, 2014, to April 5, 2016.

File	Commit Message	Date
<code>Packages/Admin/DeployStudio Admin.app/Contents</code>	1.7.3-160404	2 months ago
<code>.gitignore</code>	Ignore build data, DS_Stores	2 years ago
<code>README.md</code>	Update README.md	2 years ago

# Monitor external code

A screenshot of a GitHub diff interface comparing two versions of a shell script. The left column shows the original code with line numbers 40 through 50. The right column shows the modified code with line numbers 31 through 45. A red box highlights the addition of a mobile configuration file at line 39.

```
40 "${VOLUME_PATH}"/etc/deploystudio/bin/ds_active_directory_binding.plist
41 -fi
42 -
43 VOLUME_SYS=`defaults read
44 "${VOLUME_PATH}"/System/Library/CoreServices/SystemVersion ProductVersion | awk -F. '{ print $2 }'` 
45 if [ -z "${VOLUME_SYS}" ]
46 then
47   VOLUME_SYS=`sw_vers -productVersion | awk -F. '{ print $2 }'` 
48 fi
49
50 - cp
51 "${SCRIPT_PATH}"/ds_active_directory_binding/ds_active_directory_binding.10.5.
52 sh "${VOLUME_PATH}"/etc/deploystudio/bin/ds_active_directory_binding.sh
```

31 VOLUME\_SYS=`defaults read
32 "\${VOLUME\_PATH}"/System/Library/CoreServices/SystemVersion ProductVersion | awk -F. '{ print \$2 }'` 
33 if [ -z "\${VOLUME\_SYS}" ]
34 then
35 VOLUME\_SYS=`sw\_vers -productVersion | awk -F. '{ print \$2 }'` 
36 fi
37 +if [ \${VOLUME\_SYS} -ge 11 ]
38 then
39 + if [ ! -e
40 "\${VOLUME\_PATH}"/etc/deploystudio/bin/ds\_active\_directory\_binding.mobileconfig
41 ]
42 + then
43 + echo "Command: \${SCRIPT\_NAME} \${\*}"
44 + echo "Usage: \${SCRIPT\_NAME} <volume name>"
45 + echo "RuntimeAbortWorkflow:
46 \\"\${VOLUME\_PATH}/etc/deploystudio/bin/ds\_active\_directory\_binding.mobileconfig
47 \\" configuration profile not found!"
48 + exit 1
49 + else

# Monitor external code

<http://swcdn.apple.com/content/downloads/47/02/031-63096/dejq3zu718qejaxr8xckqzuugus4qfinke/XProtectPlistConfigData.pkg>

```
<key>Version</key>
<real>2079</real>
<key>PlugInBlacklist</key>
<dict>
    <key>10</key>
    <dict>
        <key>com.microsoft.SilverlightPlugin</key>
        <dict>
            <key>MinimumPlugInBundleVersion</key>
            <string>5.1.41212.0</string>
            <key>PlugInUpdateAvailable</key>
            <true/>
        </dict>
```

```
<dict>
    <key>MatchFile</key>
    <dict>
        <key>NSURLNameKey</key>
        <string>CleanMyMac</string>
        <key>NSURLTypeIdentifierKey</key>
        <string>public.unix-executable</string>
    </dict>
    <key>MatchType</key>
    <string>Match</string>
    <key>Identity</key>
    <data>8aMuU00d0tyWejth+Qcd5sEPzk4=</data>
```

# Monitor external code

<http://swcdn.apple.com/content/downloads/19/38/031-61264/fn3uieyv462h1xqriahvvn0ri6niv60vb0/MRTConfigData.pkg>

```
$ strings MRT
```

```
...
OSX.XcodeGhost.A
OSX.Genieo.A
/Applications/InstallMac
/Applications/Genieo.app
~/Library/Safari/Extensions/Omnibar.safariextz
~/Library/Safari/Extensions/GoldenBoy.safariextz
...
```

# Iteration

# Iteration

- Start with a manual process that could be fully automated
  - Whether a big or small job, encapsulate it into a single command
- Now make the required environment reproducible
  - Add nodes (or “runners”) to isolate this environment as required
- Isolate the inputs and outputs and turn them into variables
  - Makes the build parameterized, allows for more flexible processing

# Iteration

- Don't let perfect be the enemy of good
- Fail fast (`set -eux`, exit non-zero on unexpected results)
- Entropy is inevitable, exercise workflows regularly
- Job configurations get dense, so keep build steps short..
  - ..and as much in SCM as possible

# Patterns and tools

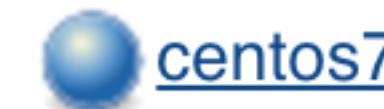
# Multiple configurations

Configuration Matrix	Debug	Release
	●	●
	●	●
	●	●
	●	●
	●	●
	●	●
	●	●
	●	●
	●	●
	●	●
	●	●
	●	●
	●	●
	●	●
	●	●
	●	●
	●	●

**Project osqueryPullRequestBuild**  
**Configurations**



[centos6](#)



[centos7](#)



[osx11](#)



[ubuntu12](#)



[ubuntu14](#)



[ubuntu16](#)

(<https://jenkins.osquery.io>)

# Triggers (SCM polling)

## Build Triggers

---

- Trigger builds remotely (e.g., from scripts)
- Build after other projects are built
- Build periodically
- Poll SCM

Schedule

H/3 \* \* \* \*

---

Would last have run at Thursday, June 9, 2016 11:21:00 o'clock AM EDT;

Ignore post-commit hooks

# Triggers (GitHub webhook on ‘push’ event)

Services / Manage Jenkins (GitHub plugin) Test service

Jenkins is a popular continuous integration server.

Using the Jenkins GitHub Plugin you can automatically trigger build jobs when pushes are made to GitHub.

## Install Notes

1. "Jenkins Hook Url" is the URL of your Jenkins server's webhook endpoint. For example: `http://ci.jenkins-ci.org/github-webhook/`.

For more information see <https://wiki.jenkins-ci.org/display/JENKINS/GitHub+plugin>.

**Jenkins hook url**

`https://5eb20bd9.ngrok.io/github-webhook/`

**Active**  
We will run this service when an event is triggered.

**Update service** **Delete service**

# Triggers (HTTP response)

## Build Triggers

- Trigger builds remotely (e.g., from scripts)
- Build after other projects are built
- Build periodically
- Poll SCM
- [URLTrigger] - Poll with a URL

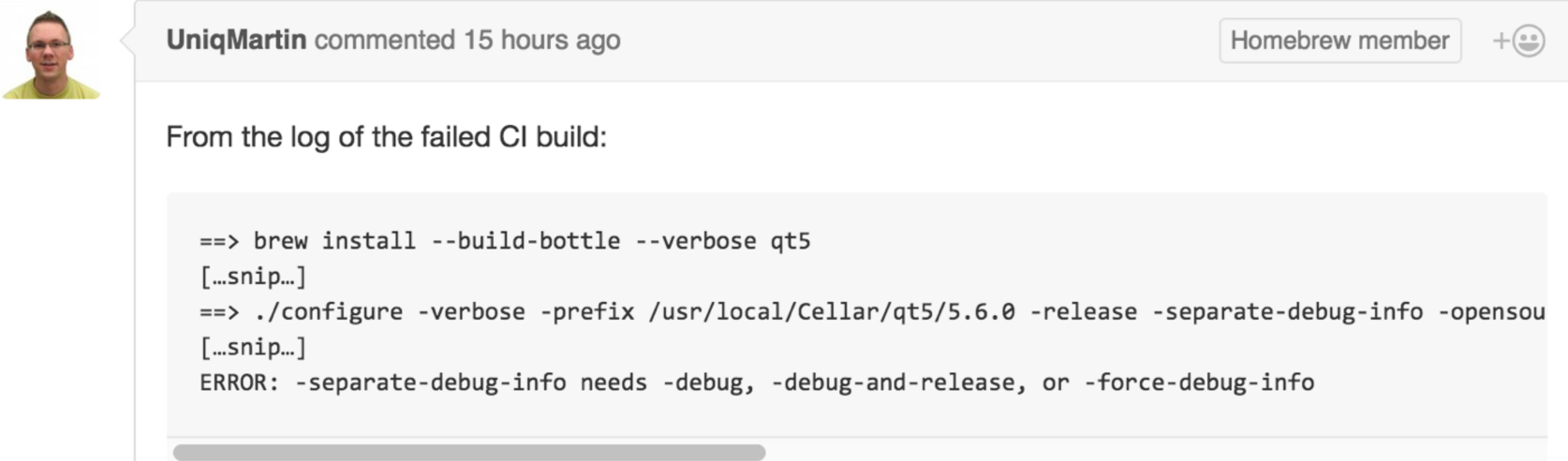
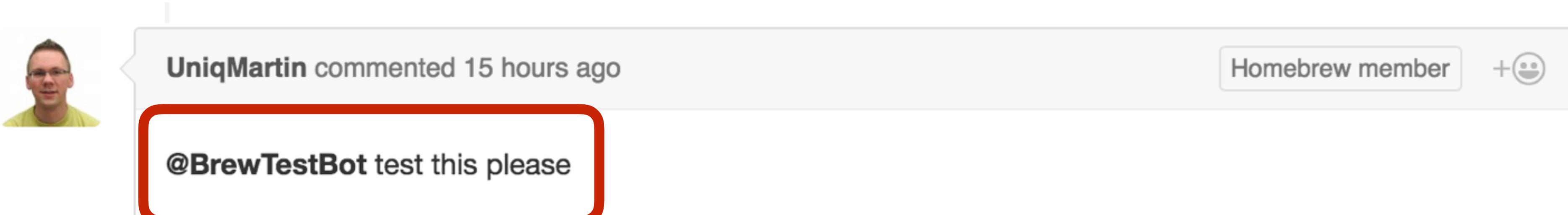
URL

http://www.deploystudio.com/Downloads/\_dss.current

## URL Response Check

- Check the status
- Check ETag
- Check the last modification Date
- Inspect URL content

# Triggers (GitHub issue/PR bot)



I have to admit that I have trouble understanding this given this checked item from the top comment:

Have you built your formula locally prior to submission with `brew install <formula>` (where

# Slave nodes

- Master -> Slave via SSH
- Slave -> Master via `java -jar slave.jar`
  - (either way, set `-Djava.awt.headless=true`)
  - (either way, slave nodes must be managed statically)
- Jenkins Swarm plugin
- Docker, EC2 dynamic slaves plugins

# Dynamic environment variables

```
# Store stuff in properties
cat > build.properties << EOF
"PKGVERSION=$PKGVERSION"
"APPNAME=$APPNAME"
"BUILDDIR=$BUILDDIR"
"BUILDLOG=$BUILDLOG"
"BUILTPKG_NAME=$BUILTPKG_NAME"
"SSH_TARGET_DIR=$SSH_TARGET_DIR"
EOF
```

See the [list of available environment variables](#)

## Inject environment variables

Properties File Path

Properties Content

EnvInject Plugin

## Send build artifacts over SSH

SSH Publishers

### SSH Server

Name

## Transfers

### Transfer Set

Source files

Remove prefix

Remote directory

Exec command

**Either Source files, Exec command or both must be supplied**

All of the transfer fields (except for Exec timeout) support substitution of [Jenkins environment variables](#)

SSH Publisher

# Publishing artifacts (SSH)

## Send build artifacts over SSH

### SSH Publishers

#### SSH Server

Name munkibuilds.org

### Transfers

#### Transfer Set

#### Source files

build/\*.pkg, build/\*.log, build/\*.txt, build/\*.html, build/MD5

#### Remove prefix

build

#### Remote directory

\$SSH\_TARGET\_DIR

#### Exec command

```
if [ $GIT_BRANCH = "origin/master" ]; then cd /var/www/munkibuilds.org; ln -f -s $SSH_TARGET_DIR/$BUILT_PKG_NAME
```

# Publishing artifacts (CIFS)

## Send build artifacts to a windows share

### CIFS Publishers

CIFS Share

Name

[Advanced...](#)

Transfers

Transfer Set

Source files

Remove prefix

Remote directory

All of the transfer fields support substitution of [Jenkins environment variables](#)

# Auth (GitHub OAuth)

## Security Realm

- Delegate to servlet container ?
- Github Authentication Plugin ?

## Global GitHub OAuth Settings

GitHub Web URI	<input type="text" value="https://github.com"/> <span style="float: right;">?</span>
GitHub API URI	<input type="text" value="https://api.github.com"/> <span style="float: right;">?</span>
Client ID	<input type="text" value="REDACTED"/> <span style="float: right;">?</span>
Client Secret	<input type="text" value="REDACTED"/> <span style="float: right;">?</span>

## Authorization

- Anyone can do anything ?
- Github Commiter Authorization Strategy ?

## Github Authorization Settings

Admin User Names	<input type="text" value="REDACTED"/> <span style="float: right;">?</span>
Participant in Organization	<input type="text" value="REDACTED"/> <span style="float: right;">?</span>
Use Github repository permissions	<input type="checkbox"/> <span style="float: right;">?</span>
Grant READ permissions to all Authenticated Users	<input type="checkbox"/> <span style="float: right;">?</span>
Grant CREATE Job permissions to all Authenticated Users	<input type="checkbox"/> <span style="float: right;">?</span>
Grant READ permissions for /github-webhook	<input checked="" type="checkbox"/> <span style="float: right;">?</span>
Grant READ permissions for /cc xml	<input type="checkbox"/> <span style="float: right;">?</span>

# Auth (Active Directory)

## **Authorization**

- Anyone can do anything
  - Legacy mode
  - Logged-in users can do anything
  - Matrix-based security

# Credentials

- Managed centrally, no need to manage on slave nodes
  - SSH keys
  - Service tokens
  - User authentication
  - etc.

# Plugins

.NET Development	Misc (ca-apm)
Android Development	Misc (cmp)
Artifact Uploaders	Misc (file)
Authentication and User Management	Misc (pipeline)
Build Notifiers	Misc (plugin-test)
Build Parameters	Misc (spot)
Build Reports	Misc (spotinst)
Build Tools	Misc (textfile)
Build Triggers	Miscellaneous
Build Wrappers	Other Post-Build Actions
Cloud Providers	Page Decorators
Cluster Management and Distributed Build	Python Development
Command Line Interface	Ruby Development
Database	RunConditions for use by the Run Condition plugin
Deployment	Scala Development
DevOps	Security
External Site/Tool Integrations	Slave Launchers and Controllers
Groovy-related	Source Code Management
iOS Development	Source Code Management related
Library plugins (for use by other plugins)	Testing
List view columns	Uncategorized
Maven	User Interface
Misc ()	Views

# Demo

Management, automation

# Job DSL Plugin

```
streamFileFromWorkspace("${RECIPE_LIST_FILE}").eachLine {
    def recipeName = it
    job {
        name "${recipeName}"
        logRotator(30, -1, -1, -1)
        label('macpro')

        multiscm {
            git('git://github.com/autopkg/autopkg.git', 'master')
        }
    }

    triggers {
        cron('H H(0-7),H(8-15),H(16-23) * * *')
    }

    steps {
        shell("echo ${recipeName} > recipe.txt")
        shell(readFileFromWorkspace('autopkg-ci/steps/autopkg_run.py'))
    }
}

configure { project ->
    def setter = project / publishers / 'hudson.plugins.descriptionsetter.DescriptionSetterPublisher'
```

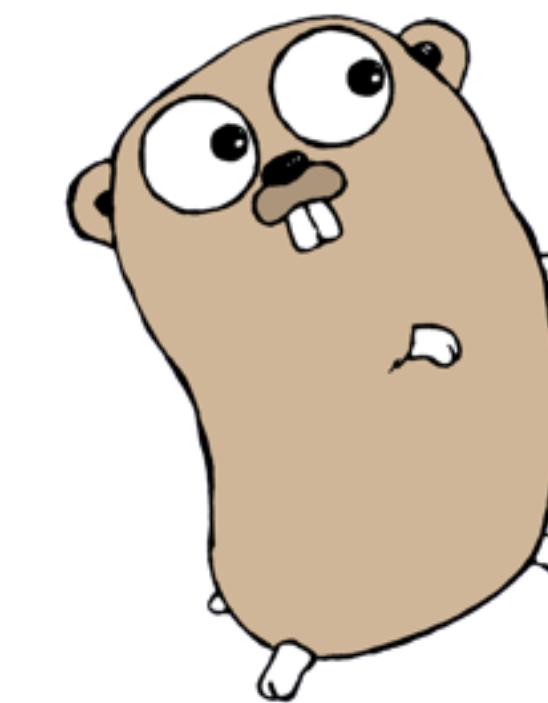
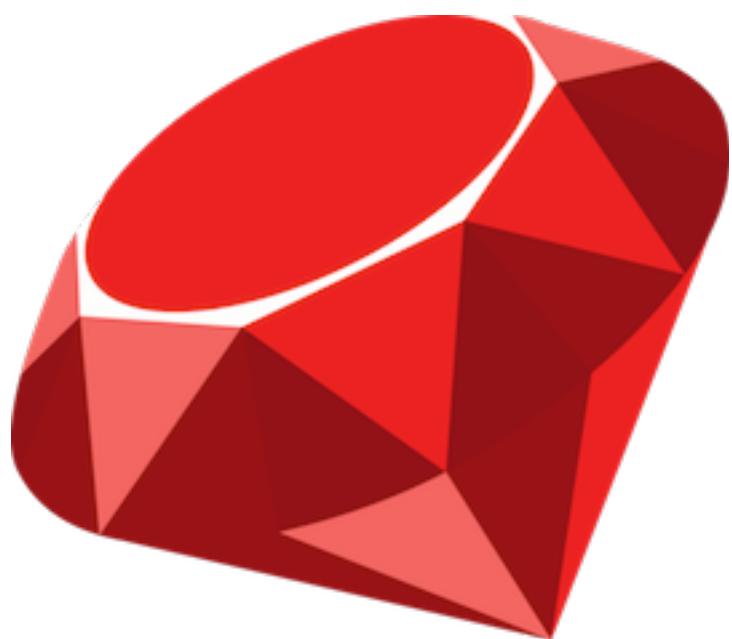
# Job DSL Plugin

All	autopkg-dsl	autopkg-recipes-all	autopkg-recipes-munki	autopkg-recipes-pkg	munki
S	W	Name ↓		Build description	
●	●	<a href="#">autopkg-Adium.munki</a>		1.5.7	
●	●	<a href="#">autopkg-AdobeAcrobatPro9Update.munki</a>		9.5.5	
●	●	<a href="#">autopkg-AdobeAcrobatProXUpdate.munki</a>		10.1.7	
●	●	<a href="#">autopkg-AdobeAir.munki</a>		3.8.0.1430	
●	●	<a href="#">autopkg-AdobeFlashPlayer.munki</a>		11.8.800.168	
●	●	<a href="#">autopkg-AdobeFlashPlayerExtractPackage.munki</a>		11.8.800.94	
●	●	<a href="#">autopkg-AdobeFlashPlayerRepackage.munki</a>		11.8.800.168	
●	●	<a href="#">autopkg-AdobeReader.munki</a>		11.0.04	
●	●	<a href="#">autopkg-BBEdit.munki</a>		10.5.5	
●	●	<a href="#">autopkg-Coda2.munki</a>		2.0.11	
●	●	<a href="#">autopkg-Cyberduck.munki</a>		4.3.1	
●	●	<a href="#">autopkg-Dropbox.munki</a>		2.0.26	
●	●	<a href="#">autopkg-Evernote.munki</a>		5.2.1	
●	●	<a href="#">autopkg-Facter.munki</a>		1.7.3	
●	●	<a href="#">autopkg-Firefox.munki</a>		23.0.1	
●	●	<a href="#">autopkg-Flip4Mac-2.munki</a>		2.4.4.2	
●	●	<a href="#">autopkg-Flip4Mac-3.munki</a>		3.2.0.16	
●	●	<a href="#">autopkg-GoogleChrome.munki</a>		29.0.1547.65	
●	●	<a href="#">autopkg-GoogleEarth.munki</a>		7.1	
●	●	<a href="#">autopkg-Handbrake.munki</a>		0.9.9	
●	●	<a href="#">autopkg-Hierra.munki</a>		1.2.1	

# Jenkins Job Builder

```
- job:  
  name: job-name  
  project-type: freestyle  
  defaults: global  
  description: 'Do not edit this job through the web!'  
  disabled: false  
  display-name: 'Fancy job name'  
  concurrent: true  
  workspace: /srv/build-area/job-name  
  quiet-period: 5  
  block-downstream: false  
  block-upstream: false  
  retry-count: 3  
  node: NodeLabel1 || NodeLabel2  
  logrotate:  
    daysToKeep: 3  
    numToKeep: 20  
    artifactDaysToKeep: -1  
    artifactNumToKeep: -1
```

# API Wrappers



# Mature workflows

```
==> git checkout origin/master
==> brew pull --clean https://github.com/Homebrew/homebrew-core/pull/1853
==> brew doctor
==> brew --env
==> brew config
==> brew readall --aliases homebrew/core
==> brew uses wimlib
==> brew fetch --retry makedepend openssl pkg-config
==> brew fetch --retry wimlib --build-bottle --force
==> brew install --only-dependencies --build-bottle --verbose wimlib
==> brew install --build-bottle --verbose wimlib
==> brew audit wimlib
==> brew bottle --verbose --json wimlib
==> brew bottle --merge --write --no-commit ./wimlib-1.9.2.el_capitan.bottle.json
==> brew uninstall --force wimlib
==> brew uninstall --force makedepend pkg-config
==> brew install ./wimlib-1.9.2.el_capitan.bottle.tar.gz
==> brew test wimlib --verbose
==> brew uninstall --force wimlib
==> brew uninstall --force openssl
==> git checkout master -f
==> git reset --hard
==> brew cleanup --prune=7
==> git clean -ffdx
HEAD is now at 11d47e8 boneyard-formula-pr: add new command.
Removing Cellar/
Removing Library/Locks/
Removing etc/openssl/
Recording test results
Archiving artifacts
```

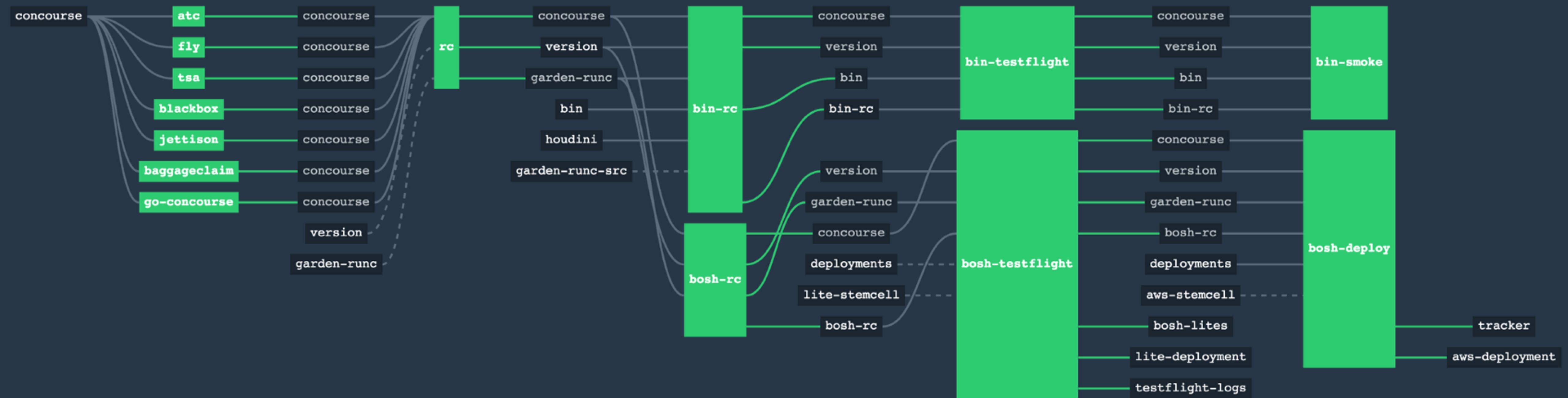
 **Build el\_capitan (9-Jun-2016 3:53:00)**

 **Build Artifacts**  
[wimlib-1.9.2.el\\_capitan.bottle.json](#) 425 B   
[wimlib-1.9.2.el\\_capitan.bottle.tar.gz](#) 463.18 KB 

 No changes.

 Started by upstream project [Homebrew Core Pull Requests](#) build number  
originally caused by:

# Pipelines



# Concourse (Pivotal) pipeline

# Jenkins Blue Ocean Project

Jenkins / Blue Ocean #423

Branch master  
Commit #601366d

Changes by Michael Neale, Ben Waldo and Ivan Meredith

3 minutes and 42 seconds  
14 minutes ago

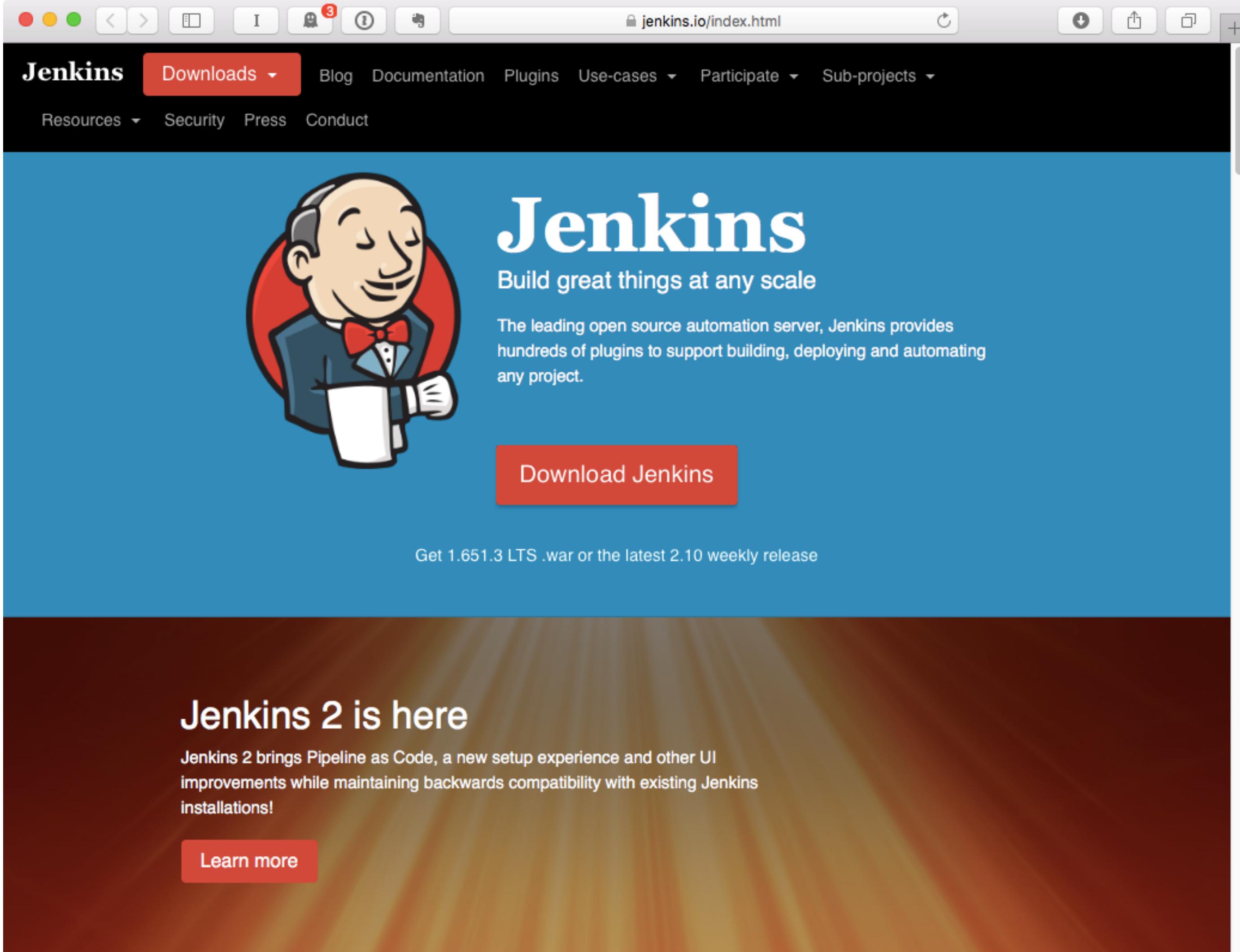
Pipeline Changes Tests Artifacts Re-run

The diagram illustrates a Jenkins pipeline structure. It starts with a 'Build' stage (green checkmark), followed by a 'Test' stage which branches into 'JUnit' (green checkmark), 'DBUnit' (green checkmark), and 'Jasmine' (green checkmark). From the 'Test' stage, the flow continues to 'Browser Tests', which further branches into 'Firefox' (green checkmark), 'Edge' (red X), 'Safari' (red X), and 'Chrome' (red X). Finally, the pipeline moves through 'Dev', 'Staging', and 'Production' stages.

Build log – Edge

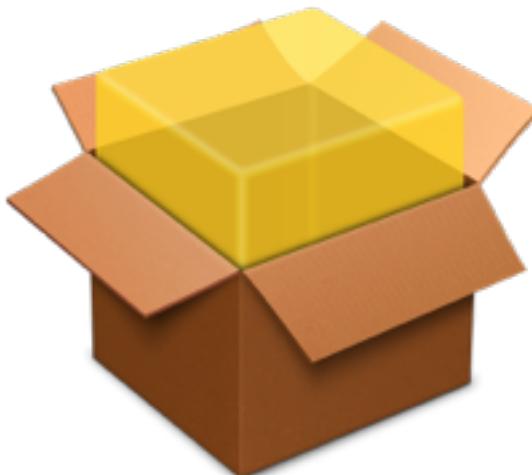
✓ > Start Docker container	3 minutes and 42 seconds
✓ > Warm maven caches	5 seconds
✓ > Install Java Tools	8 seconds
✗ > Maven	6 minutes and 12 seconds

# Jenkins



A screenshot of a web browser displaying the Jenkins website at [jenkins.io/index.html](https://jenkins.io). The page features a large blue header with the Jenkins logo (a cartoon character in a tuxedo holding a coffee cup) on the left. To the right of the logo, the word "Jenkins" is written in large white letters, followed by the tagline "Build great things at any scale". Below this, a paragraph describes Jenkins as "The leading open source automation server" and mentions its support for building, deploying, and automating projects via hundreds of plugins. A prominent red "Download Jenkins" button is centered below the text. At the bottom of the page, there is a section about Jenkins 2, which includes a "Learn more" button.

<https://jenkins.io>



`brew install jenkins`

# Other CI platforms

Name	Self-hosted / Enterprise	Cloud-hosted	Bring your own runner
buildbot	✓		✓
Go (Thoughtworks)	✓		✓
Concourse (Pivotal)	✓		✓
Buildkite		✓	✓
Travis-CI	✓	✓	
Appveyor		✓	
GitLab CI	✓	✓	✓
Circle CI		✓	
Bamboo	✓	✓	
BitBucket Pipelines		✓	
buddybuild		✓	

# Thank you!



@timsutton



@tvsutton

<https://macops.ca/macdevopsyvr-2016>