

Advanced Munki Infrastructure



Rick Heil - Senior IT Manager @ Myelin



@refrshingapathy



@refreshingapathy



github.com/rickheil

2017
MACADMIN'S
CONFERENCE

AT PENN STATE®

Slides, resources, and links are available now at
rickheil.com/munkipsu2017

We will cover:

- My Prime Directive: Why Cloud Munki?
- The Power of Middleware
- Tales from Production: Ways to use Cloud Services with Munki
- Teamwork in Cloud-Based Munki
- Using CI to Automate Your Cloud
- Questions / Comments / Tomato Throwing

THE PRIME DIRECTIVE

**No, not this
directive**





How do we manage Macs in five diverse offices?

Five offices and remote workers challenged our existing Munki model.

“The munki server is just a
web server!”
~Greg



Scaling Approaches

- Host the munki server out of one office
- Host a munki server in each office
- Host the munki server on a cloud server somewhere
(Rackspace, Amazon EC2, physical hardware in one of the company colos)
- Something else?

**Then I found out
about middleware.**

**With middleware, the munki
server doesn't have to be
just a web server anymore.**





munki

?



Cloud Leak: How A Verizon Partner Exposed Millions of Customer Accounts

Updated on July 12, 2017 by Dan O'Sullivan

Filed under: [cloud](#), [data breaches](#), [misconfiguration](#)

Security



Adding security to S3

- Lock down access to IAM accounts
- Calculate a signature for the requests with a pair of keys
- Signature is added to “Authorization” header
- If the signature is valid, S3 allows the request

```
219
220     options = {'url': url,
221                'file': tempdownloadpath,
222                'follow_redirects': follow_redirects,
223                'ignore_system_proxy': ignore_system_proxy,
224                'can_resume': resume,
225                'additional_headers': header_dict_from_list(custom_headers),
226                'download_only_if_changed': onlyifnewer,
227                'cache_data': cache_data,
228                'logging_function': display.display_debug2}
229     display.display_debug2('Options: %s' % options)
230
231     # Allow middleware to modify options
232     if middleware:
233         display.display_debug2('Processing options through middleware')
234         # middleware module must have process_request_options function
235         # and must return usable options
236         options = middleware.process_request_options(options)
237         display.display_debug2('Options: %s' % options)
238
```



```
110 def process_request_options(options):
111     """Make changes to options dict and return it.
112     This is the fuction that munki calls."""
113     if S3_ENDPOINT in options['url']:
114         headers = s3_auth_headers(options['url'])
115         options['additional_headers'].update(headers)
116     return options
```

Options: {'logging_function': <function display_debug2
at 0x11172b9b0>, 'ignore_system_proxy': None,
'additional_headers': {'User-Agent':
u'managedsoftwareupdate/3.0.2.3347 Darwin/16.6.0'},
'file': u'/Library/Managed Installs/catalogs/
production.download', 'cache_data':

Options: {'logging_function': <function display_debug2 at 0x103dd49b0>, 'ignore_system_proxy': None, 'additional_headers': {'x-amz-content-sha256': 'e3b0c44298fc1c142jfdqkd358996fb92427ae41e4649b934ca495991b7852b855', 'x-amz-date': '20170701T212047Z', 'Authorization': 'AWS4-HMAC-SHA256 Credential=AKIAJHJGSDKFJDRSK5QQ/20170701/us-east-1/s3/aws4_request, SignedHeaders=host;x-amz-date, Signature=a584c61729318348ajfkaw39jfsdfjedk07db0d325c98906103c86e386d9a769ec5', 'User-Agent': u'managedsoftwareupdate/3.0.2.3347 Darwin/16.6.0'}, 'file': u'/Library/Managed Installs/catalogs/production.download',

**With middleware, the munki
server doesn't have to be
just a web server anymore.**

Middleware Options

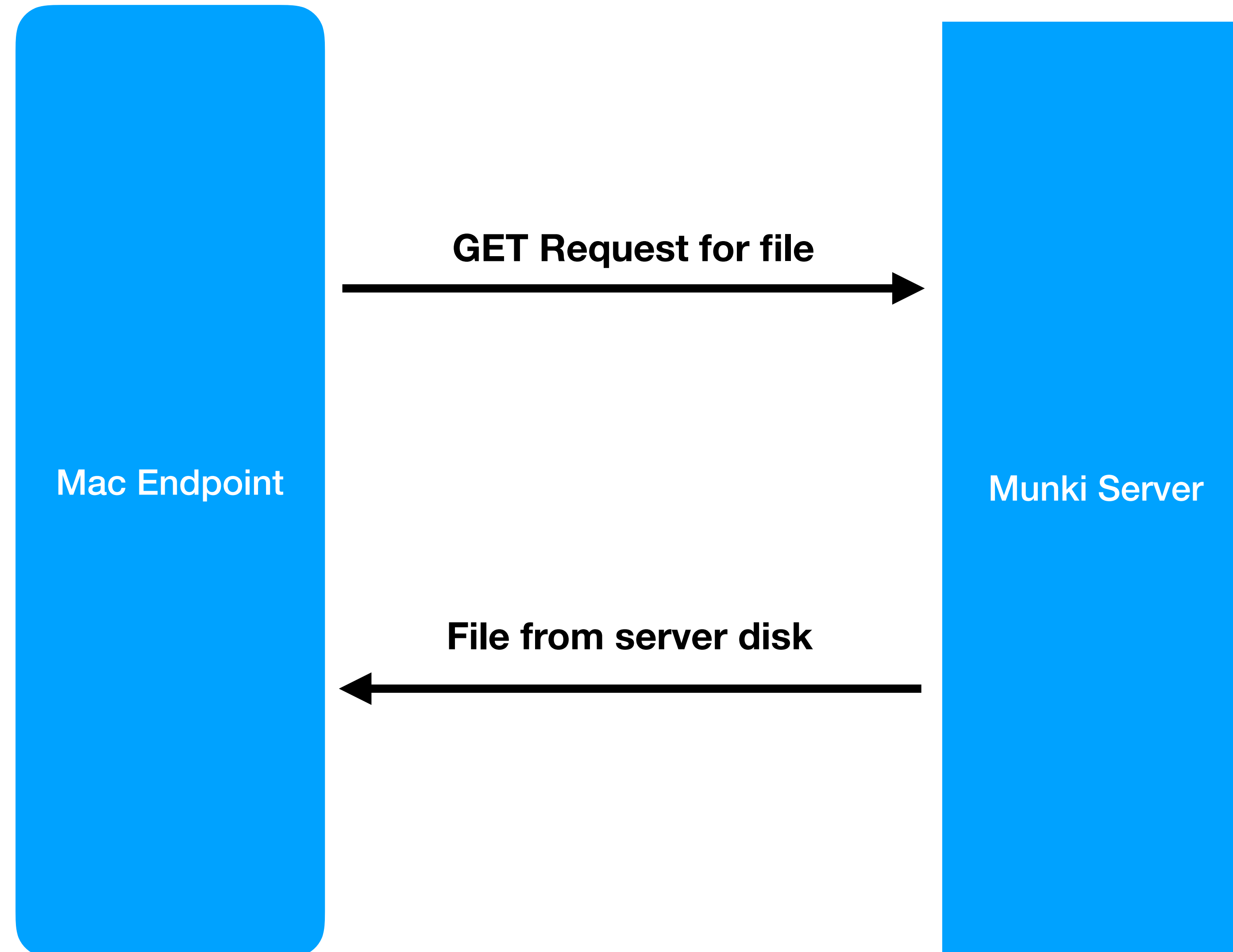
- Amazon S3 (Wade)
- Amazon CloudFront (Aaron Burchfield)
- Google Cloud Storage (Wade)



Tales from Production

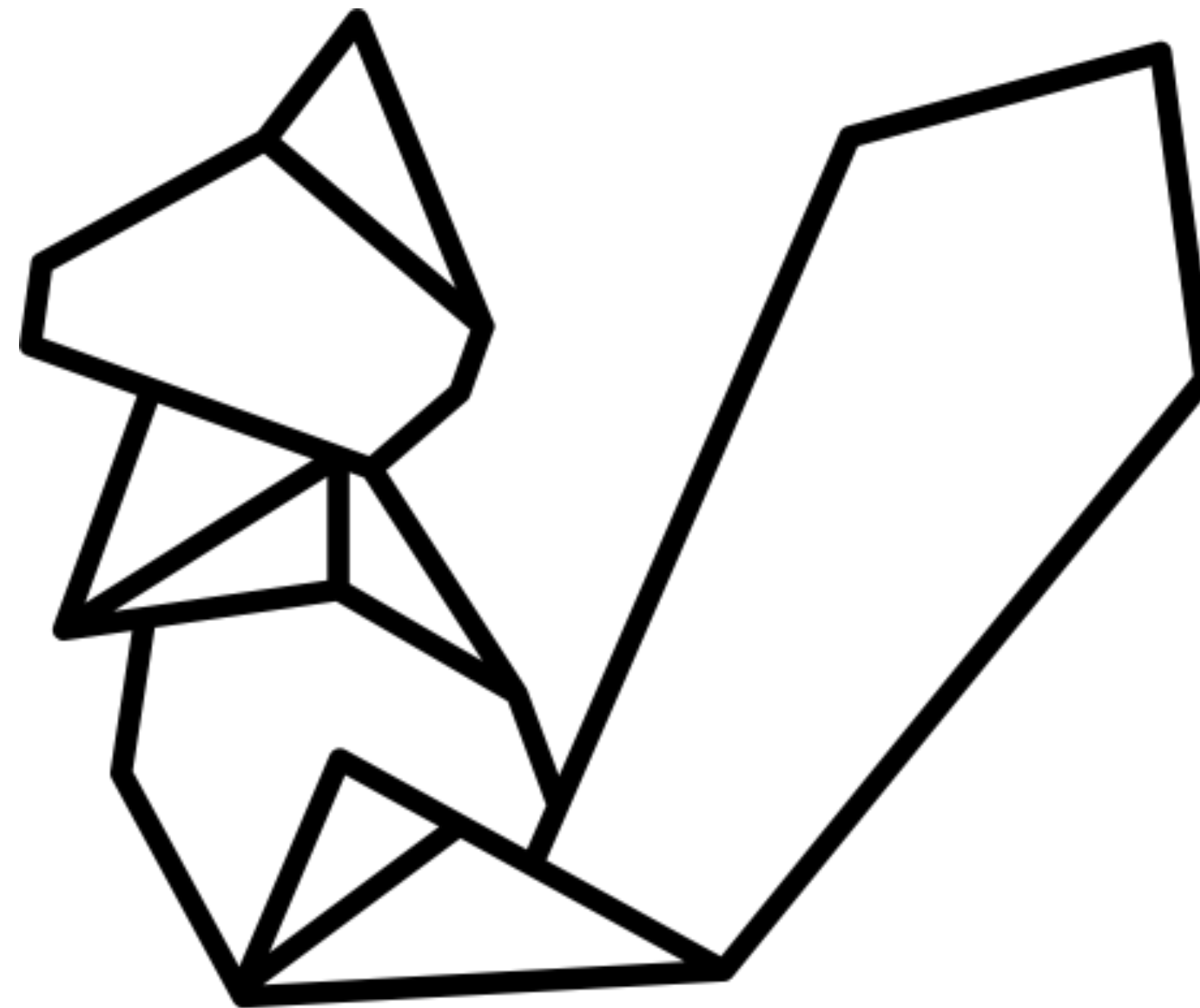
or: tell me how I can use this!

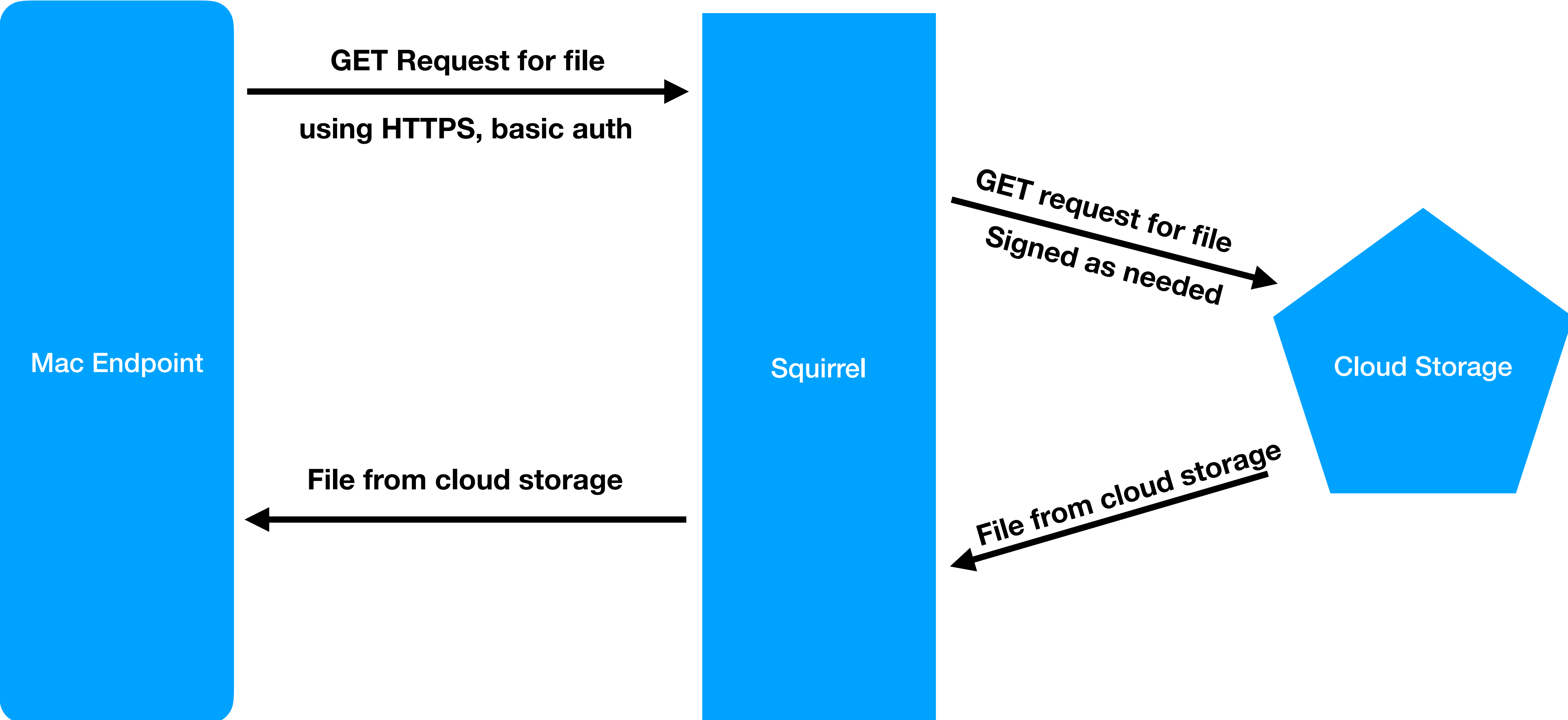
A “Normal” Munki Server



1: Use Squirrel

- open source server written in Go by Victor (@groob)
- built in HTTPS via LetsEncrypt
- can serve files from a local disk, S3, or GCS
- no special configuration needed on clients beyond basic auth





Squirrel

Pros

- incredibly easy to set up
- only need one cloud account
- keys are never on clients
- basic auth + HTTPS = good security out of the box

Cons

- each request fetches the file
- this adds up meaning you will pay more bandwidth charges
- large updates could saturate your outbound network connection, causing slowdowns.

2: Amazon S3 (Direct)

- uses middleware so machines talk directly to S3
- built-in HTTPS certificate from Amazon
- leverage Amazon's reliability and scale

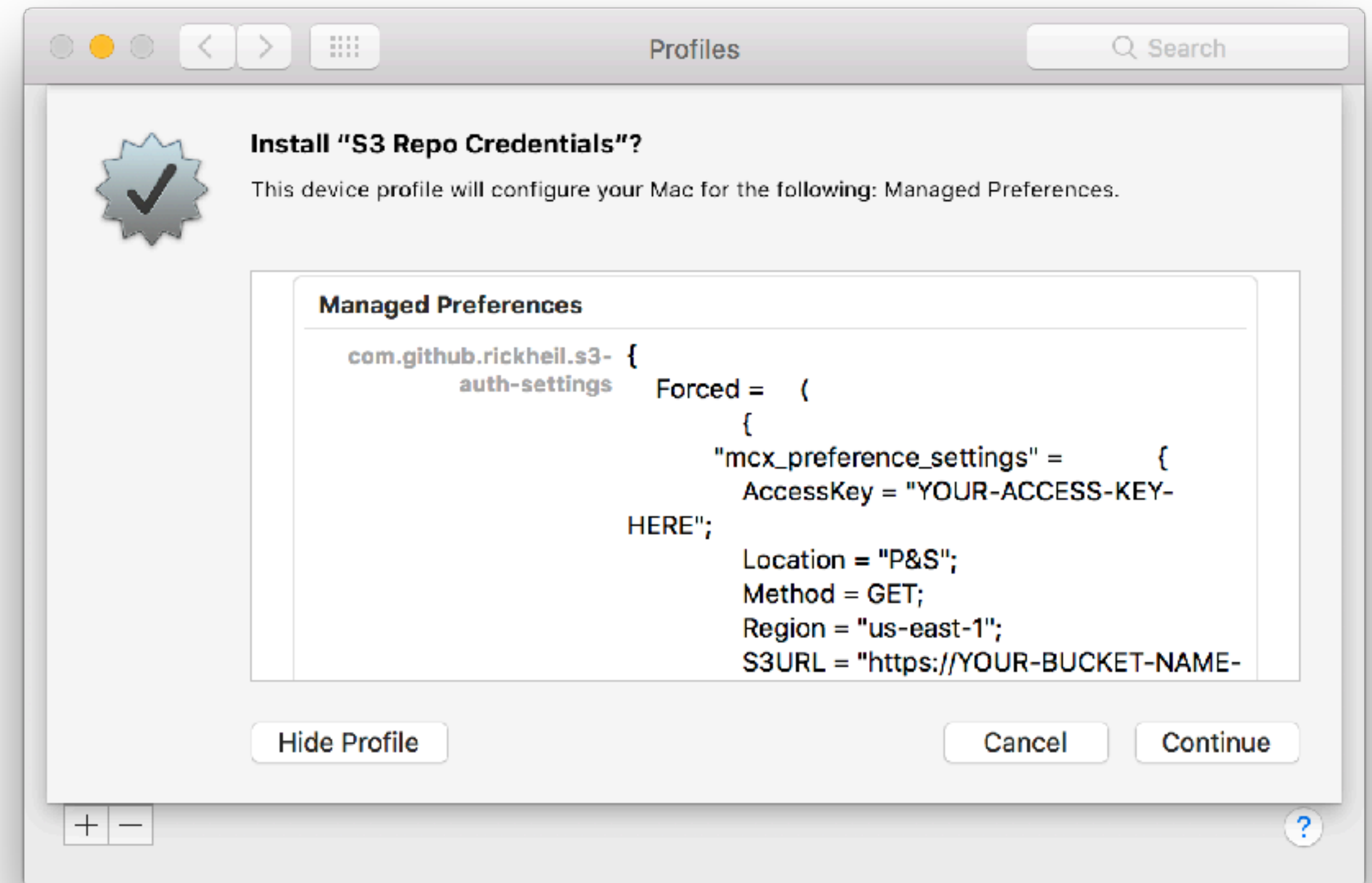


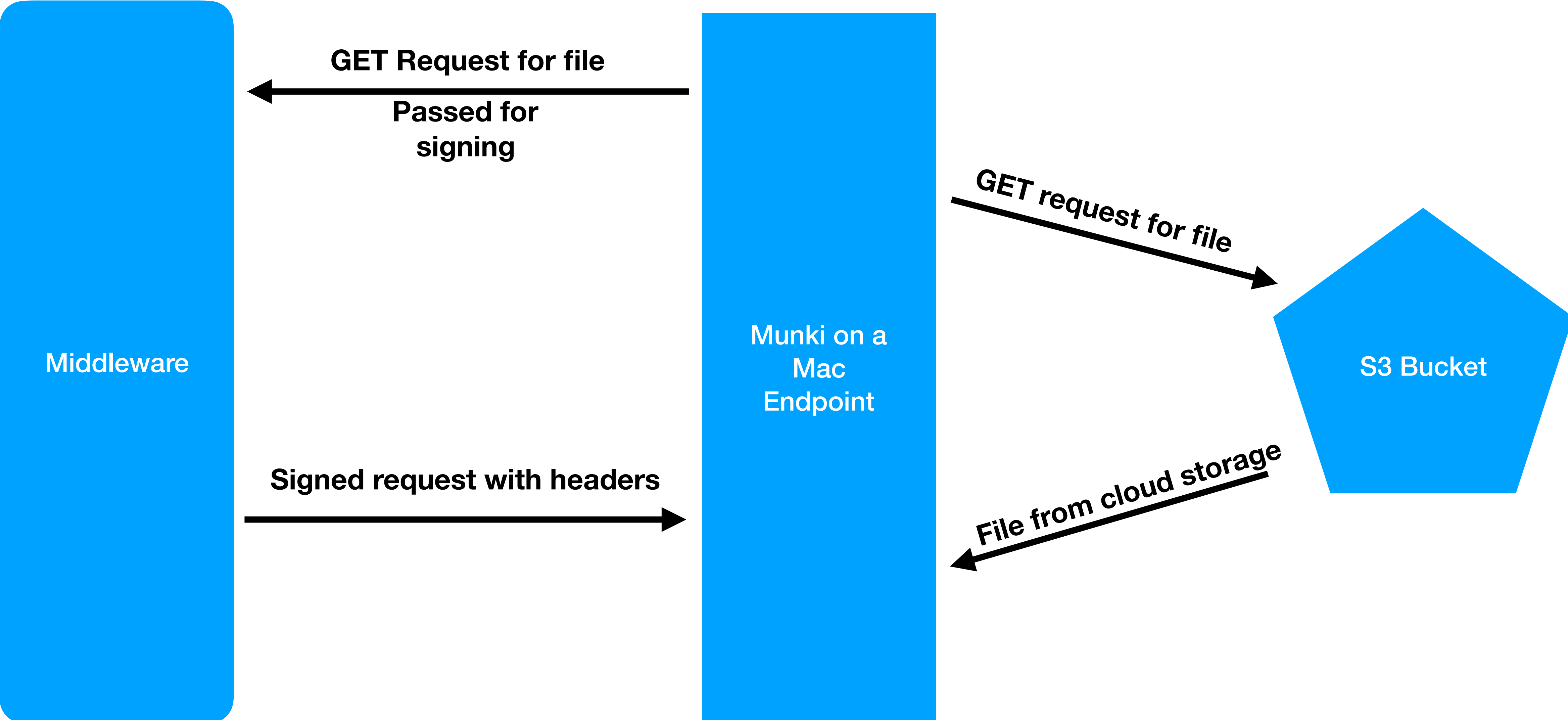
Setup on the S3 Side

- create your bucket, disable public access ACLs
- create a read-only IAM user for the clients
- create a read/write IAM user for yourself

Installing S3 Middleware

- create a pkg to drop the script at /usr/local/munki/middleware_s3.py
- create a profile with the IAM credentials and any other middleware settings needed





Amazon S3 (Direct)

Pros

- no local infrastructure / server needed at all
- straight forward to set up and troubleshoot
- Amazon buckets come with HTTPS by default
- S3 is generally pretty reachable

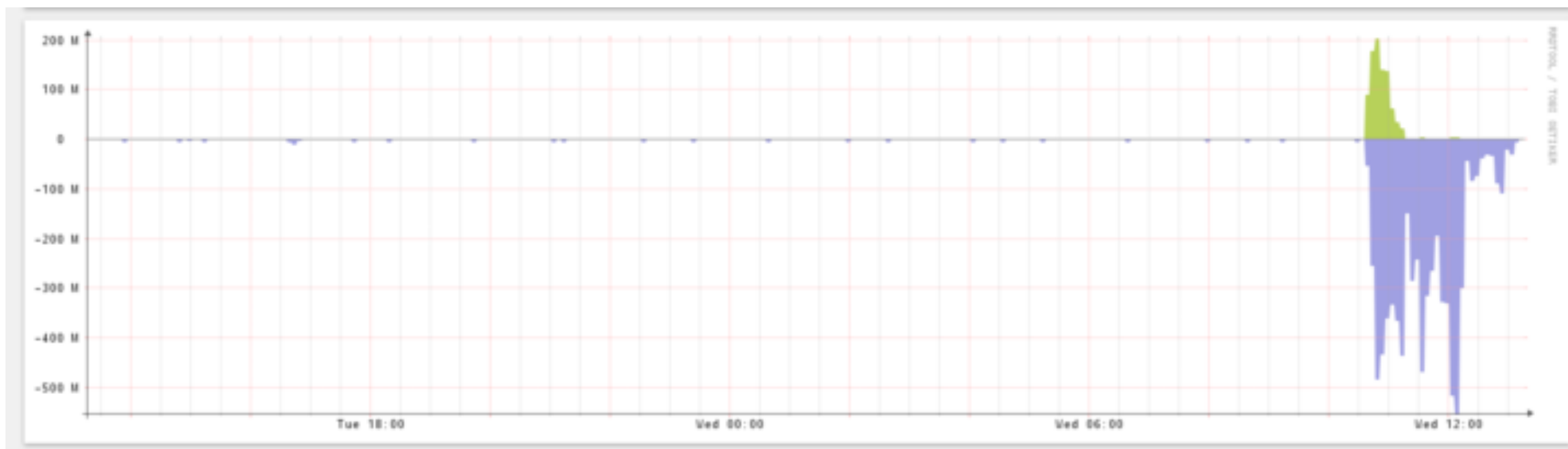
Cons

- IAM keys are on the client
- you're still paying for each file retrieval
- large updates could saturate your network connection, causing slowdowns.

3: Hybrid S3 (Cached)

- uses a combination of S3 middleware and on-prem reverse proxies
- HTTPS on local proxies and direct to S3
- allows granular cache control
- super-fast LAN downloads



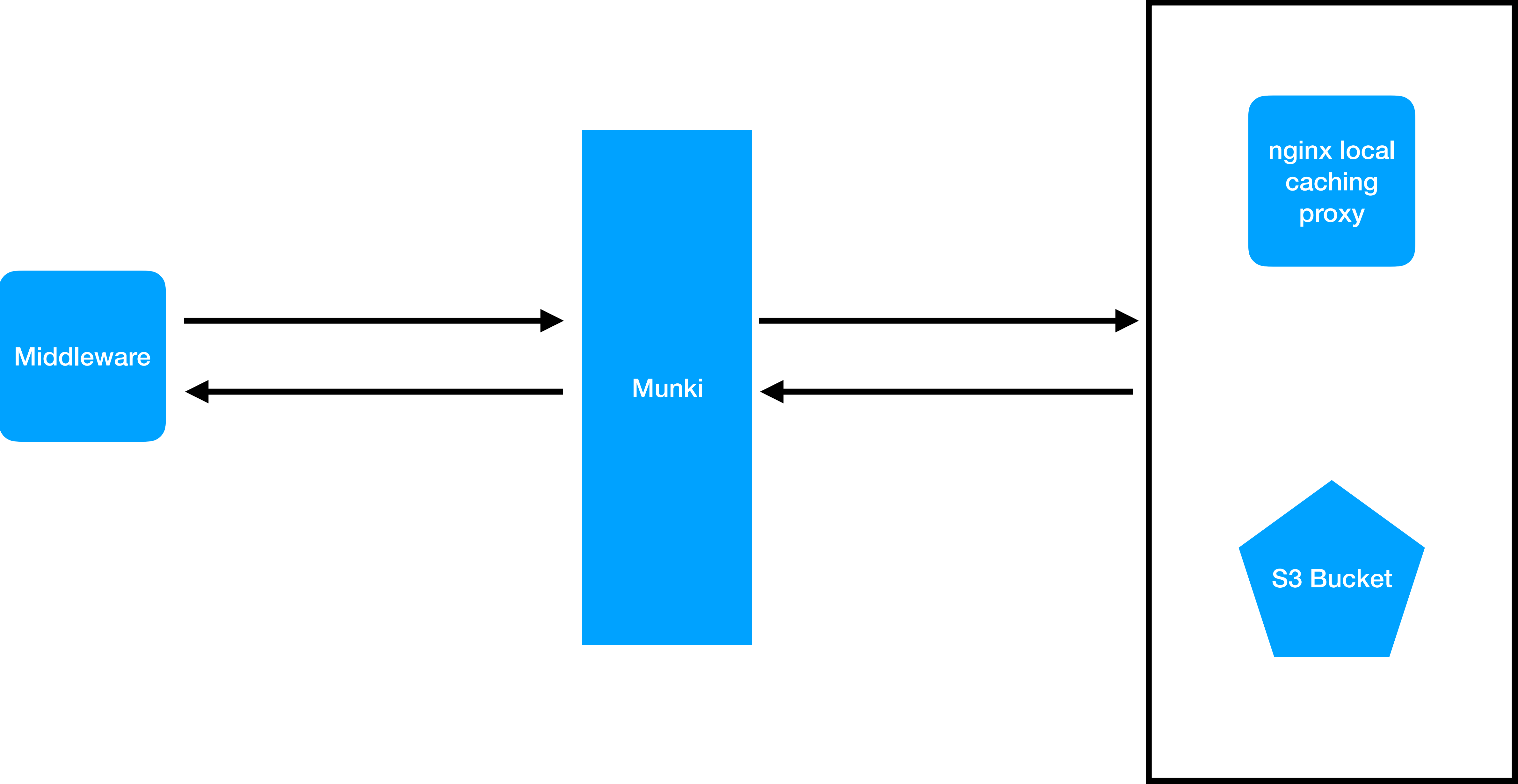


But why local proxies?

- significant bandwidth savings
- significant cost savings
- wire speed delivery of updates to LAN clients

What's in a proxy?

- Small VM - 1 vCPU, 1-2GB RAM
- hosted locally in each office
- run your favorite flavor of OS with nginx installed
- ngx_aws_auth module
- cronjob to re-generate keys (remember, signing keys expire!)
- enough disk space to hold your munki repo once (plus a little bit)



Hybrid S3 (Cached)

Pros

- FAST updates on LAN
- saves you on both bandwidth and S3 transaction costs
- All HTTPS all the time
- Caching proxies keep you going while S3 is down
- Caching proxies are super tunable to your needs

Cons

- IAM keys are on the client
- Requires local infrastructure (small server or VM)
- Requires DNS changes
- More complexity means more things to troubleshoot

Other fun proxy tricks

- Due to historical reasons, our old munki URL had /repo/ in the path. We can redirect this easily with the proxies.
- We can efficiently serve specific static content (e.g. pictures that are in descriptions) easily with the same system
- Set up alerts through our log shipping system if a client requests the site_default manifest
- Use the stub status module to track various metrics for making pretty graphs

Back to the prime directive:

How do **we** manage Macs in five diverse offices?







git

Git: The Good

- all changes (commits) tied to a specific person - great for audit-able logs
- follows the “infrastructure as code” push we are making on all fronts
- provides a centralized repository but allows independent local work

Git: The Bad

- LFS doesn't have object expiration (yet)
- Team members need to have an understanding of git
- Branching strategy wars

Tips on getting started...

- Put your catalog files in .gitignore and generate them when you deploy
- Use Git LFS (or Git Fat, if you like) for DMG, pkg, and mpkg. Let regular git track everything else
- Use “git lfs prune” on your local copy to keep things under control
- Agree on a branching strategy ahead of time


```
1. less
vim  #1 less #2
* 2e0f669 - (HEAD -> develop, origin/develop) Merge branch 'hotfix/fix-2011' into develop (2 days ago) <Rick Heil>
| \
| * 317a50f - fix office 2011 old dep (2 days ago) <Rick Heil>
| * 0b59b4b - (tag: ci-changes-july-6) Merge branch 'hotfix/ci-changes-july-6' (3 days ago) <Rick Heil>
| | \
* | | 863f7eb - clean up older packages with more aggressive repoclean, remove dupe xcodes (2 days ago) <Rick Heil>
* | | fa718c2 - Merge branch 'hotfix/ci-changes-july-6' into develop (3 days ago) <Rick Heil>
| \ \ \
| | | /
| | /
| * | 2a090b1 - remove extraneous steps and commented out lines, remove explicit git lfs as recent gitlab release fixed
the need for manual pull (3 days ago) <Rick Heil>
| | /
| * 6dbba7e - Merge branch 'develop' into 'master' (3 days ago) <Rick Heil>
| | \
| | /
| /
* | 954de8b - Merge branch 'hotfix/merge-manifests-round-1' into develop (3 days ago) <Rick Heil>
| \ \
| | /
| * d7ac604 - (tag: merge-manifests-round-1) Merge branch 'develop' into 'master' (3 days ago) <Rick Heil>
| | \
| | /
| /
* | 317333d - add fep PPT template as optional for all users (4 days ago) <Rick Heil>
* | b935168 - Merge branch 'feature/VLC-2.2.6' into develop (4 days ago) <Rick Heil>
| \ \
| * | 537090f - vlc to prod (4 days ago) <Rick Heil>
| * | 7969dc0 - (origin/feature/VLC-2.2.6) Updating VLC to version 2.2.6 (7 days ago) <AutoPkgr>
| | /
* | 3fd945b - Merge branch 'feature/Spotify-1.0.58.573.g57c9cd87' into develop (4 days ago) <Rick Heil>
| \ \
| * | 282dfba - spotify to prod (4 days ago) <Rick Heil>
| * | 0980200 - (origin/feature/Spotify-1.0.58.573.g57c9cd87) Updating Spotify to version 1.0.58.573.g57c9cd87 (4 days ago) <AutoPkgr>
| | /
| * cb6ee6e - Merge branch 'master' of gitlab.pas-digital.com:ps-it/munki (9 days ago) <Rick Heil>
| | \
| | * 68a7407 - Merge branch 'hotfixVictorFlaviusBoot' (9 days ago) <matt>
| | \
: |
```



GitLab

What is CI?

- Stands for “Continuous Integration”
- Uses a “runner” to accomplish tasks
- Can run scripts directly on shell (bash on Linux/macOS, cmd on Windows) or using a docker container

Why use CI with Munki?

- Allows for continuous deployment - whatever is on git's master branch is a mirror of production repo
- No need to generate and monitor tons credentials for production - provides a single path to deploy
- Repeatable build environments (using docker containers)
- Logged builds and tests for troubleshooting
- Run scheduled pipelines

CI Pitfalls

- Will do exactly what you tell it to do
- Can be slow to work with large repos
- Another moving part in the process
- Temptation to script literally everything in your life


```
image: rickheilps/munki-ci:v0.2
```

```
build:
```

```
  stage: build
```

```
  variables:
```

```
    GIT_STRATEGY: clone
```

```
  script:
```

```
  - python /builds/ps-it/munki/tests/munkilinter /builds/ps-it/munki/
  - python /builds/ps-it/munki/build/makecatalogs /builds/ps-it/munki/
  - aws s3 sync --delete catalogs/ s3://$AWS_BUCKET/catalogs
  - aws s3 sync --delete icons/ s3://$AWS_BUCKET/icons
  - aws s3 sync --delete manifests/ s3://$AWS_BUCKET/manifests
  - aws s3 sync --delete pkgs/ s3://$AWS_BUCKET/pkgs
```

```
  only:
```

```
  - master
```

```
  artifacts:
```

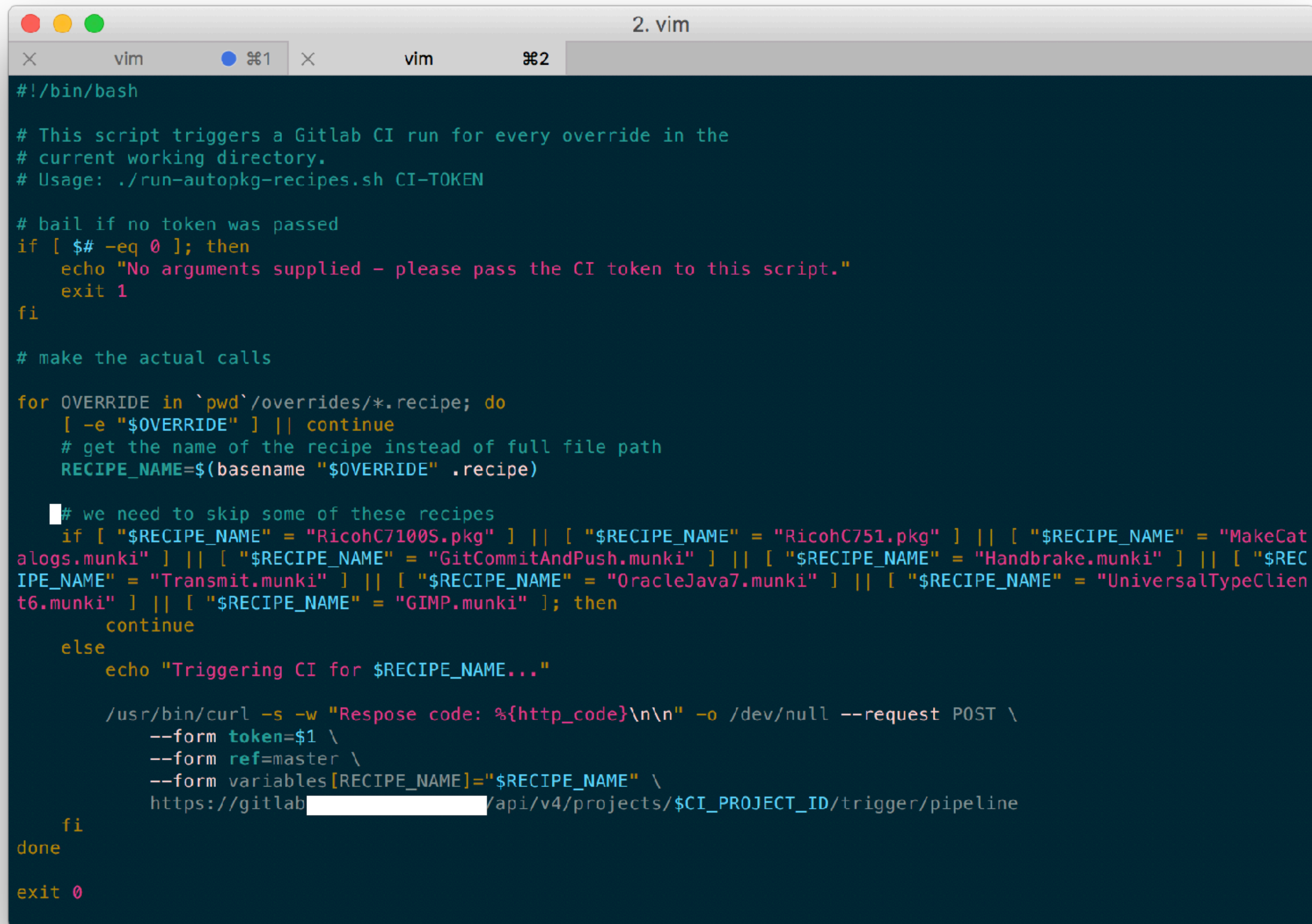
```
    paths:
```

```
      - catalogs/
```

munkilinter

- Compares file hashes to ensure that LFS did everything correctly. We don't want corrupt pkgs going to S3.
- Checks that pretty things are correct in pkginfo files
- Checks plist syntax of all manifests
- Code available on my site for you to browse and modify


```
autopkg:
  stage: build
  tags:
  - OSX
  script:
    - echo $RECIPE_NAME
    - (if [[ ! -z "$RECIPE_NAME" ]]; then python build/autopkg_tools.py -k $TRELLO_KEY -t $TRELLO_TOKEN -c $
TRELLO_LIST -y $VT_API_KEY -r $RECIPE_NAME; else /bin/bash build/run-autopkg-recipes.sh $TRIGGER_TOKEN; fi);
  only:
  - master
~
```

```
#!/bin/bash

# This script triggers a Gitlab CI run for every override in the
# current working directory.
# Usage: ./run-autopkg-recipes.sh CI-TOKEN

# bail if no token was passed
if [ $# -eq 0 ]; then
    echo "No arguments supplied - please pass the CI token to this script."
    exit 1
fi

# make the actual calls

for OVERRIDE in `pwd`/overrides/*.recipe; do
    [ -e "$OVERRIDE" ] || continue
    # get the name of the recipe instead of full file path
    RECIPE_NAME=$(basename "$OVERRIDE" .recipe)

    # we need to skip some of these recipes
    if [ "$RECIPE_NAME" = "RicohC7100S.pkg" ] || [ "$RECIPE_NAME" = "RicohC751.pkg" ] || [ "$RECIPE_NAME" = "MakeCatalogs.munki" ] || [ "$RECIPE_NAME" = "GitCommitAndPush.munki" ] || [ "$RECIPE_NAME" = "Handbrake.munki" ] || [ "$RECIPE_NAME" = "Transmit.munki" ] || [ "$RECIPE_NAME" = "OracleJava7.munki" ] || [ "$RECIPE_NAME" = "UniversalTypeClient6.munki" ] || [ "$RECIPE_NAME" = "GIMP.munki" ]; then
        continue
    else
        echo "Triggering CI for $RECIPE_NAME..."

        /usr/bin/curl -s -w "Respose code: %{http_code}\n\n" -o /dev/null --request POST \
            --form token=$1 \
            --form ref=master \
            --form variables[RECIPE_NAME]="$RECIPE_NAME" \
            https://gitlab[REDACTED]/api/v4/projects/$CI_PROJECT_ID/trigger/pipeline
    fi
done

exit 0
```


autopkg_tools.py

- Released by Facebook CPE team on Github
- Allows for custom “create_task” implementation (in my case, a Trello card)
- Added our branching model and Gitlab elements
- This all took me an afternoon. Thank you Nick and team!

CI Tips

- Use Gitlab's variables function to keep secrets out of the repo
- Use proper error handling
- Only hard-enforce the standards you care deeply about
- Protect production!

**Go to Mac Justice's
session in 207 next if you
want more Gitlab!**

**I don't want to use CI
for munki!**

Okay, use file repo plugins instead!

**What does this effort
get us?**

Questions?

Resources: <https://rickheil./com/munkipsu2017>

Feedback: <https://bit.ly/psumac2017-163>