



A macOS client built from code

MacSysAdmin 2023



Henry Stamerjohann, Éric Falconnier , Johannes Schüller

October 4, 2023



October 4, 2023



A macOS client built from code

A 'configuration-as-code'
session

October 4, 2023



Why?

A macOS client built from code



Why

build a macOS client from code?

- ▶ **Save time** - We want to "set and forget", reduce human error
- ▶ **Reliability** - We want stability
- ▶ **Auditability** - We want authoritativeness for change management

A macOS client built from code



About Zentral

A bit of history in open source device management

- Started as open source project
- Promising concept
- Let's become a MDM vendor!

A macOS client built from code



How did we think this was a good idea?

- A self-fulfilling prophecy
- Challenge accepted



Scope

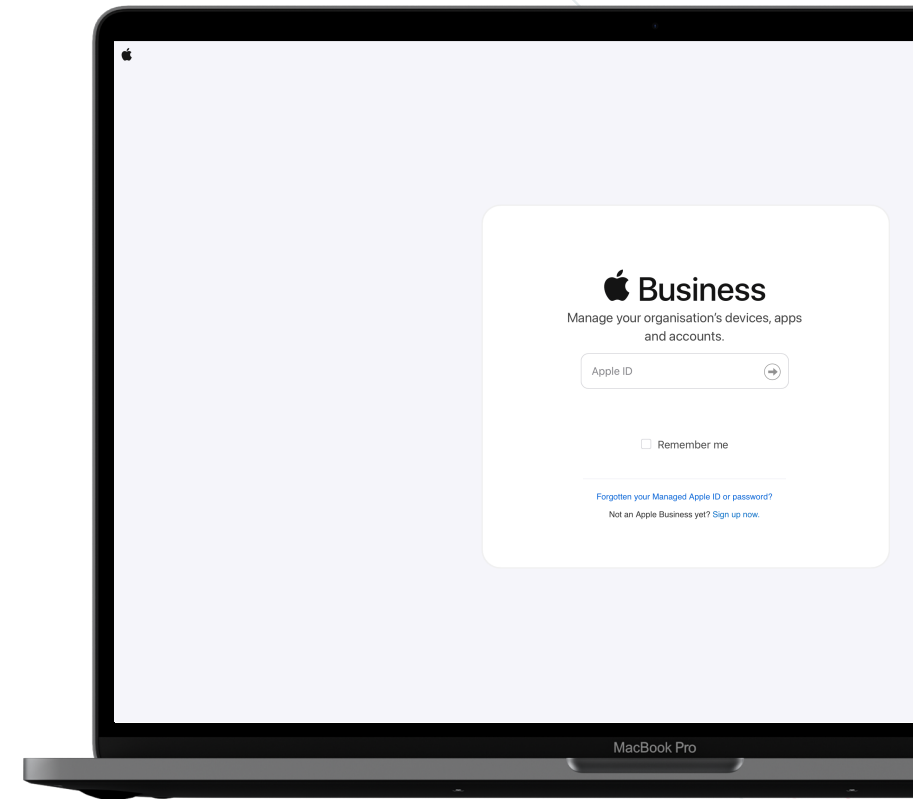
A macOS client built from code



Setting the stage

- **Supervised client**

Organizational device ownership for full control

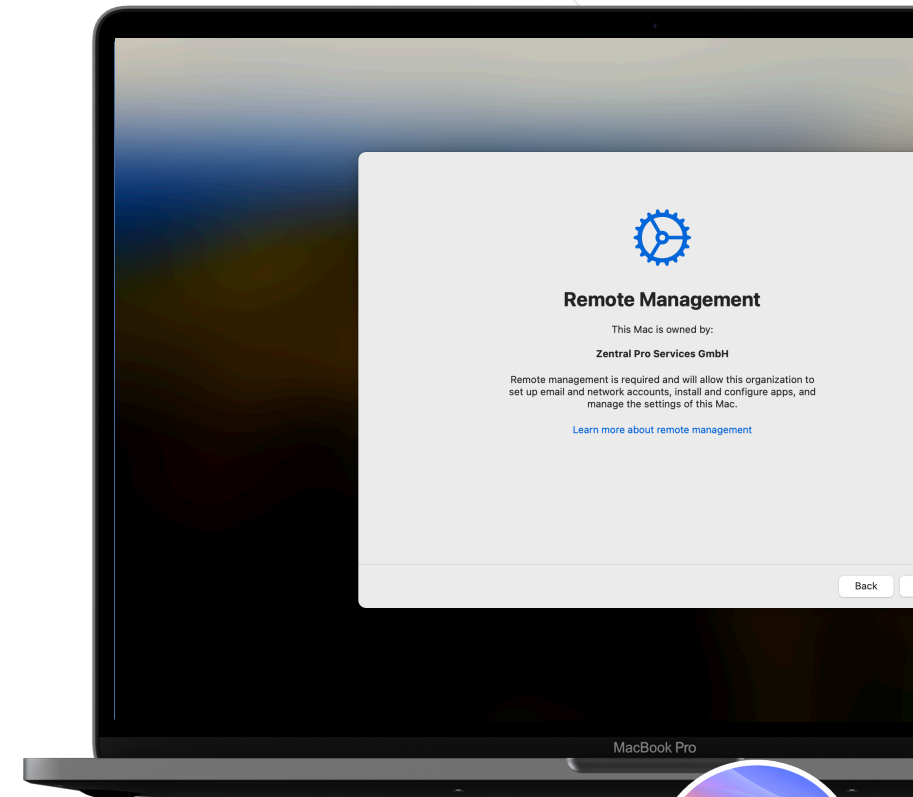


A macOS client built from code

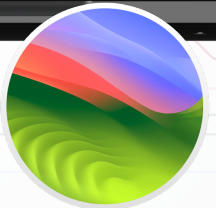


Setting the stage

- ▶ **Supervised client**
Organizational device ownership for full control
- ▶ **Modern devices & OS**
AppleSilicon/T2, Sonoma with auto updates



A macOS client built from code

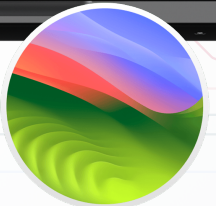
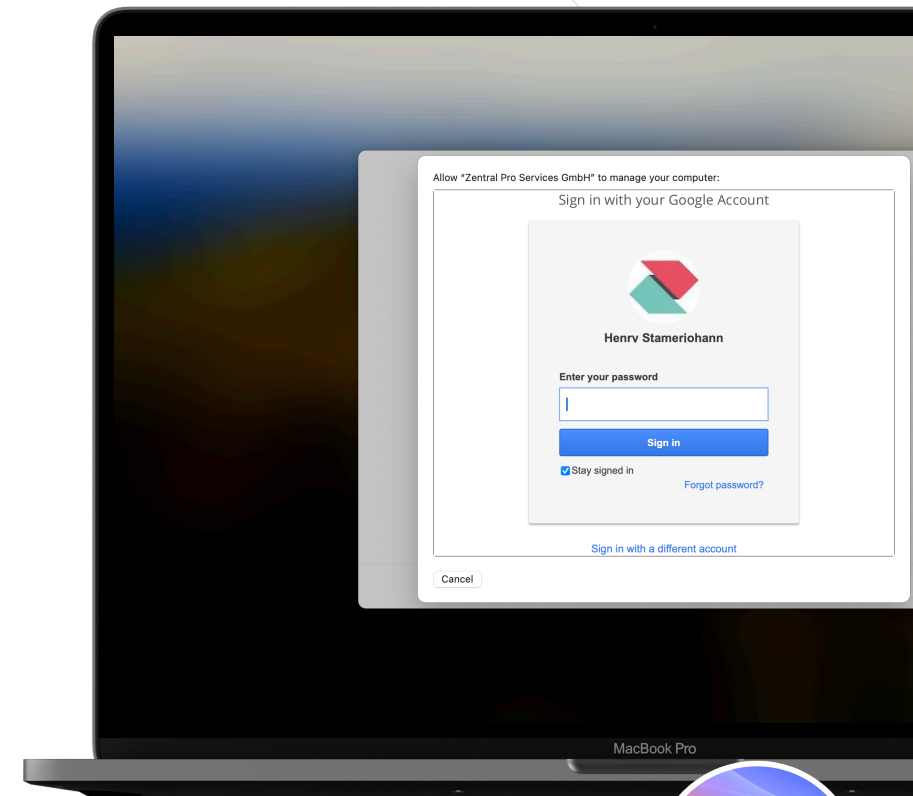




Setting the stage

- ▶ **Supervised client**
Organizational device ownership for full control
- ▶ **Modern devices & OS**
AppleSilicon/T2, Sonoma with auto updates
- ▶ **User / device association**
One-to-one with IdP authentication

A macOS client built from code





Best in class tools

- Patch management
Nothing beats Munki

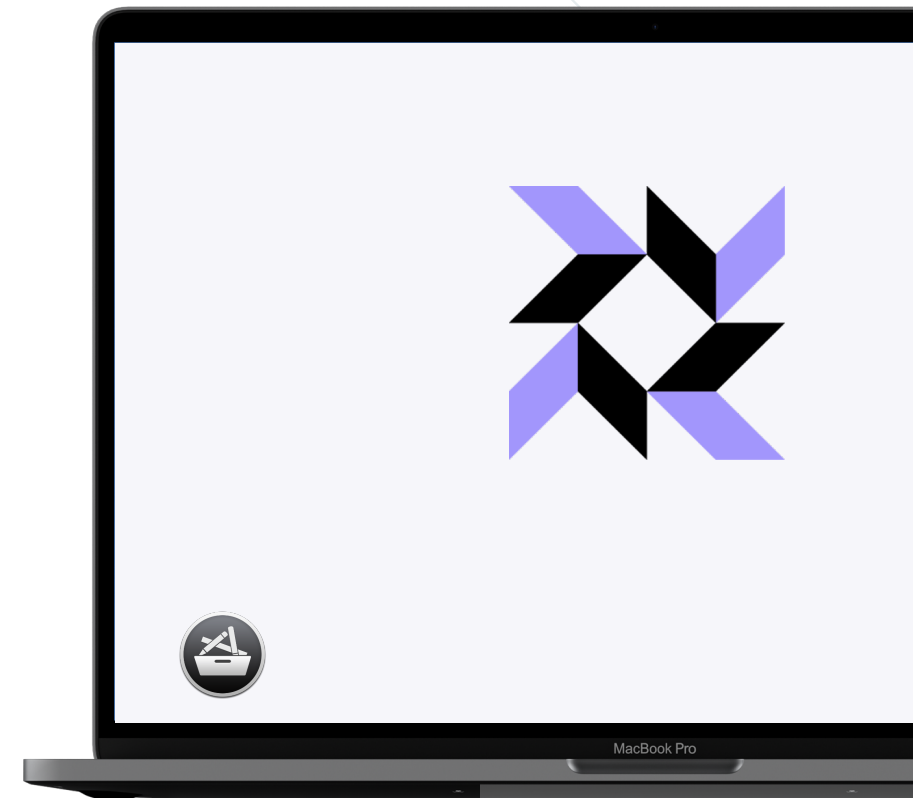
A macOS client built from code





Best in class tools

- ▶ **Patch management**
Nothing beats Munki
- ▶ **Obervability & troubleshooting**
With Osquery

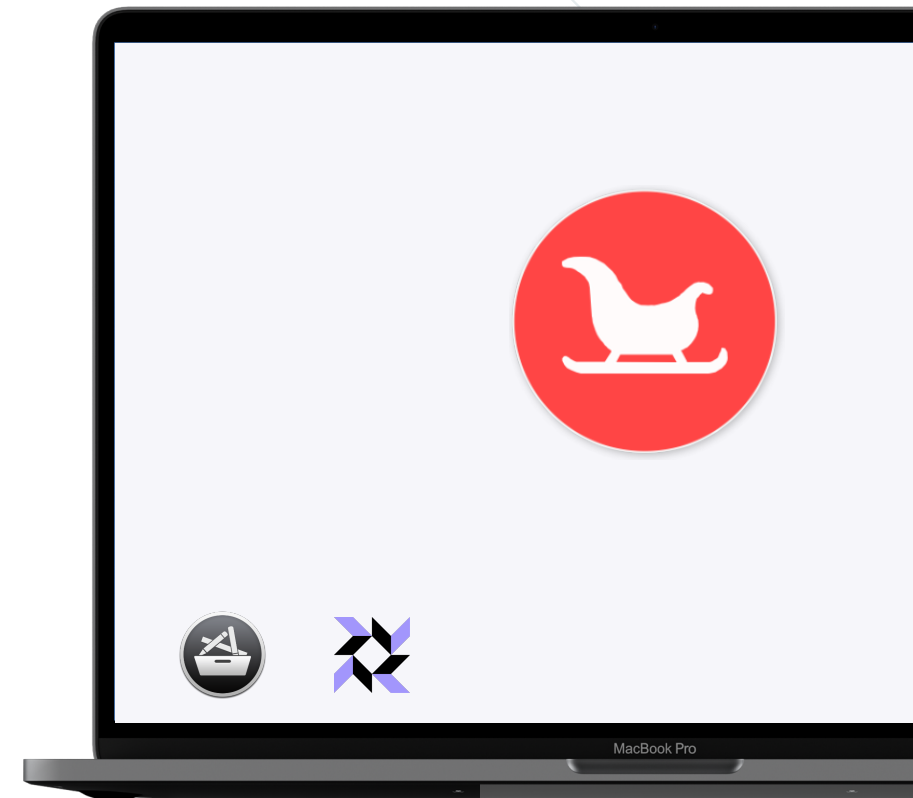


A macOS client built from code



Best in class tools

- ▶ **Patch management**
Nothing beats Munki
- ▶ **Obervability & troubleshooting**
With Osquery
- ▶ **Allow / Blocklisting**
With Google Santa



A macOS client built from code



Best in class tools

- ▶ **Patch management**
Nothing beats Munki
- ▶ **Obervability & troubleshooting**
With Osquery
- ▶ **Allow / Blocklisting**
With Google Santa

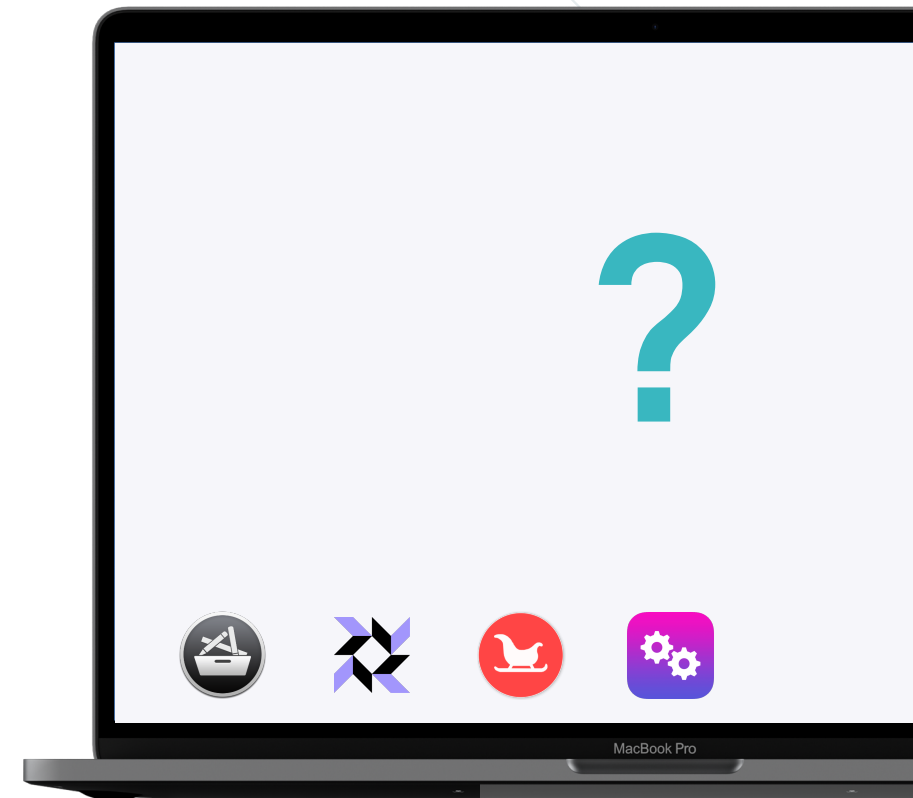


A macOS client built from code



The hardening

- FileVault
- Recovery lock / Firmware password
- Selection of mSCP profiles

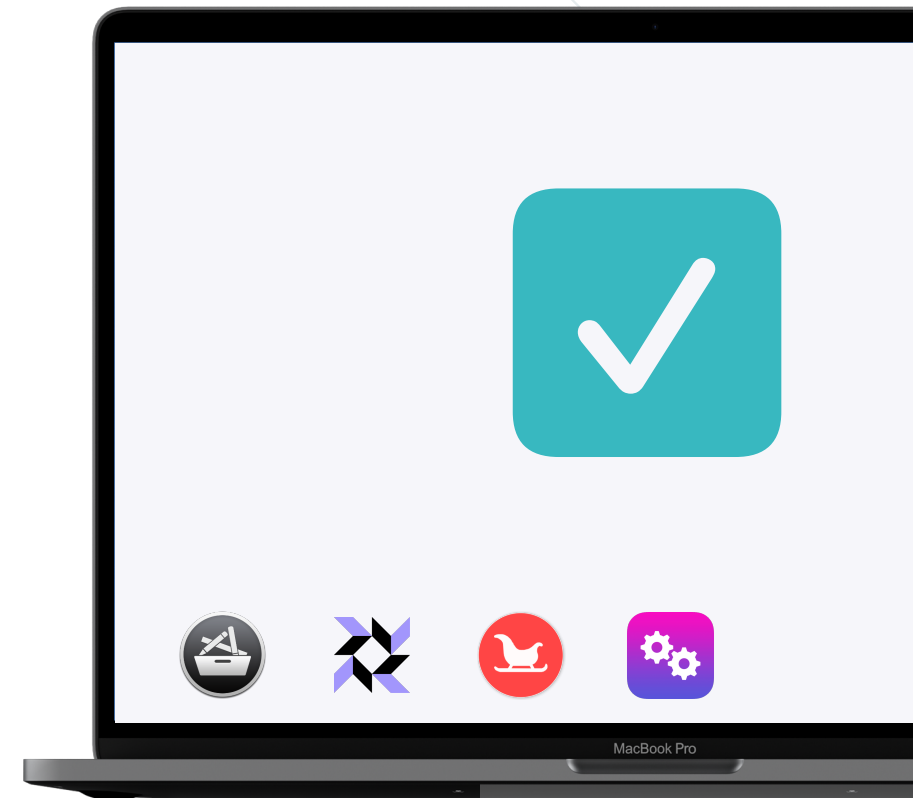


A macOS client built from code



The hardening

- FileVault
- Recovery lock / Firmware password
- Selection of mSCP profiles
- Compliance checks
Inventory or Osquery based



A macOS client built from code



Out of scope

- EDR / 3rd party AntiVirus
- BYOD / User enrollment



A macOS client built from code



Scope summary

Global Decisions

Supervised client

Modern devices and OS

One-to-one
user-device association

Tools

Patch-Management
Munki

Observability / troubleshooting
Osquery

Allow / Blocklisting
Santa

Hardening

FileVault

Recovery lock/
Firmware password

mSCP Profiles

A macOS client built from code



How?

A macOS client built from code



Bootstrap package

A macOS client built from code



Bootstrap package

What it is, why?

- First package distributed via MDM during enrollment
- Install Munki
- Trigger first Munki run using Outset

A macOS client built from code

The screenshot shows the Central console interface. The breadcrumb trail is: Home / MDM / Devices / 1376B77B-EA6D-4B01-9C6D-458D21C9AA47 / Co. The page title is "Device commands". Below this is a table with two columns: "Name" and "Artifact". The table lists various commands and their corresponding artifacts. The "InstallEnterpriseApplication" command is highlighted with an orange box, showing its artifact as "Monolith - Bootstrap package v2".

Name	Artifact
DeclarativeManagement	-
DeviceInformation	-
ProfileList	-
CertificateList	-
InstalledApplicationList	-
SecurityInfo	-
SetupFileVault (InstallProfile)	-
DeclarativeManagement	-
DeviceInformation	-
SetRecoveryLock	-
SetupFileVault (InstallProfile)	-
InstalledApplicationList	Monolith - Bootstrap package v2
InstallEnterpriseApplication	Monolith - Bootstrap package v2
ProfileList	-
CertificateList	-
InstalledApplicationList	-
DeclarativeManagement	-
DeclarativeManagement	-

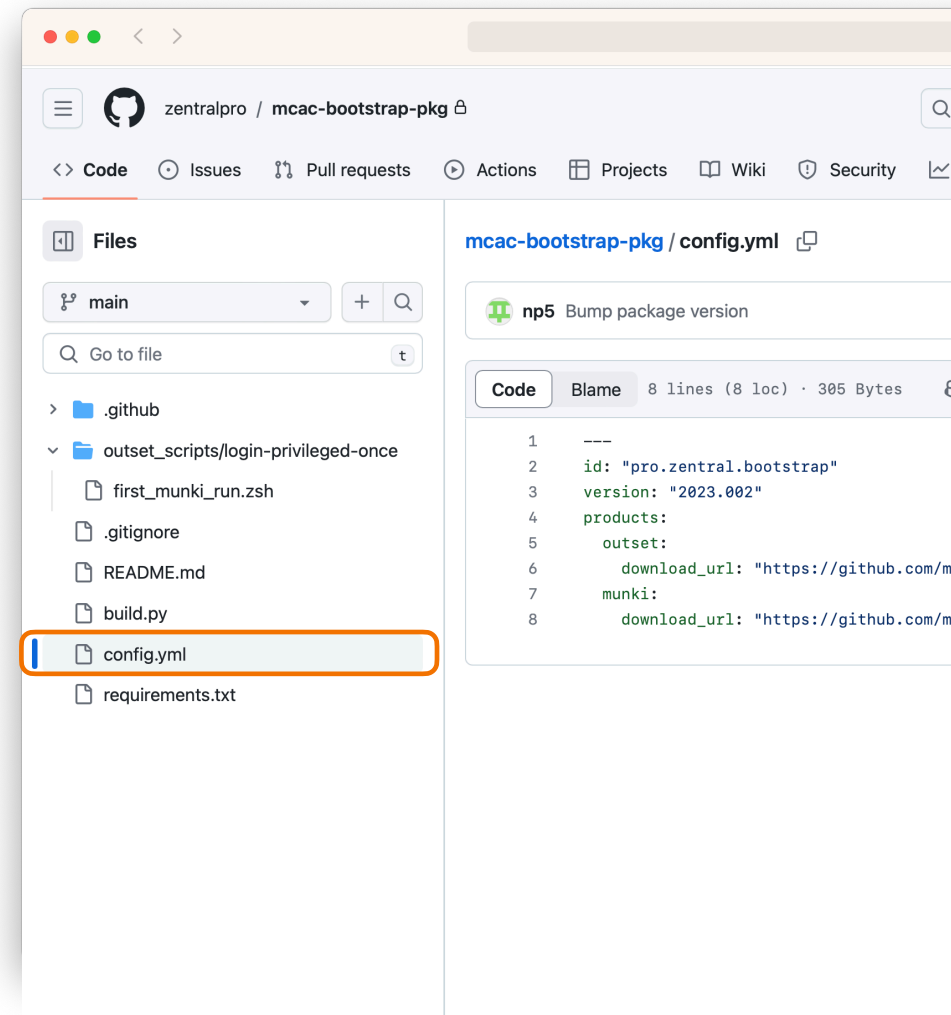


Bootstrap package

The code

- Configuration file in YAML

A macOS client built from code





Bootstrap package

The code

- Configuration file in YAML
- Product ID and version

A macOS client built from code

The screenshot shows the GitHub interface for the repository 'zentralpro / mcac-bootstrap-pkg'. The left sidebar displays the file tree with the following structure:

- main (selected branch)
- .github
- outset_scripts/login-privileged-once
 - first_munki_run.zsh
- .gitignore
- README.md
- build.py
- config.yml (highlighted)
- requirements.txt

The right pane shows the content of 'mcac-bootstrap-pkg / config.yml'. The file is 8 lines (8 loc) and 305 Bytes. The code is as follows:

```
1 ---
2 id: "pro.zentral.bootstrap"
3 version: "2023.002"
4 products:
5   outset:
6     download_url: "https://github.com/m
7   munki:
8     download_url: "https://github.com/m
```

Lines 2 and 3 are highlighted with an orange box, indicating the product ID and version.



Bootstrap package

The code

- Configuration file in YAML
- Product ID and version
- Download URLs for Outset and Munki

A macOS client built from code

The screenshot shows a GitHub repository page for 'zentralpro / mcac-bootstrap-pkg'. The left sidebar displays the file structure, with 'config.yml' selected. The main content area shows the code for 'mcac-bootstrap-pkg / config.yml'. The code is in YAML format and includes a comment 'np5 Bump package version'. The code is as follows:

```
1 ---
2 id: "pro.zentral.bootstrap"
3 version: "2023.002"
4 products:
5   outset:
6     download_url: "https://github.com/m
7   munki:
8     download_url: "https://github.com/m
```

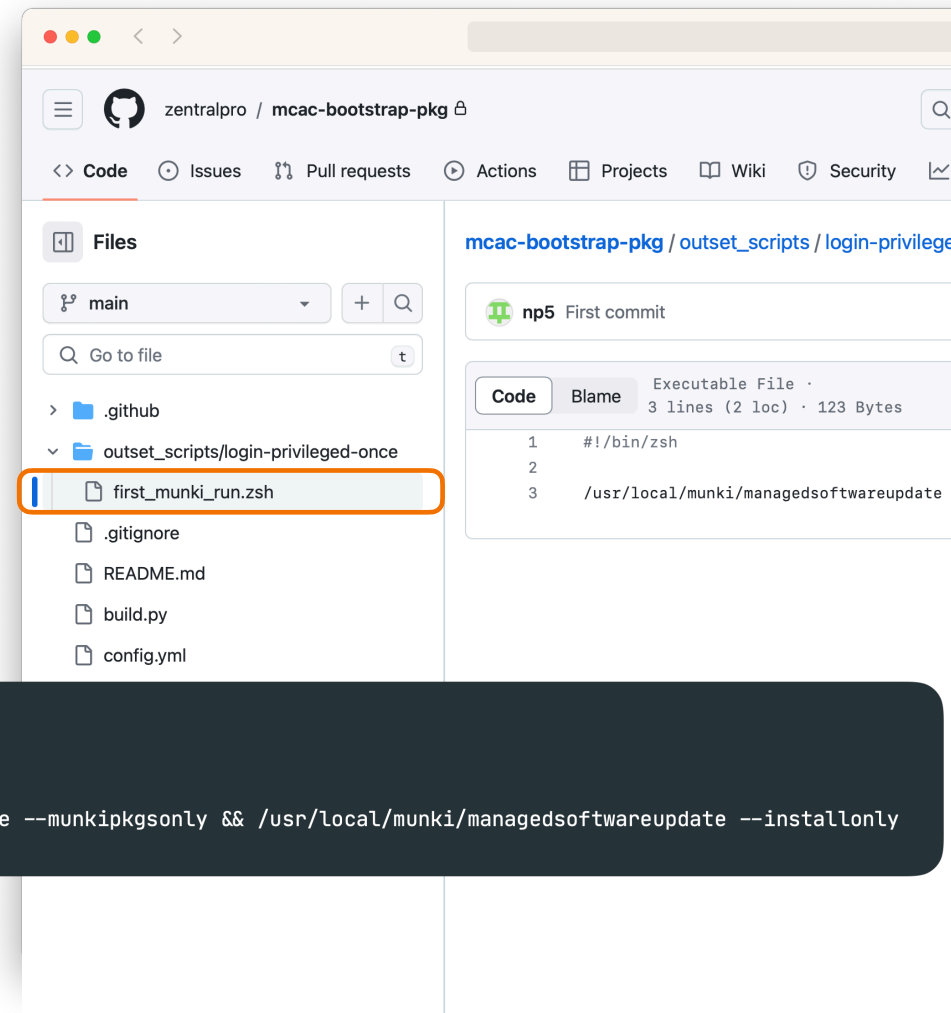
The code is displayed with line numbers 1 through 8. The 'products' section is highlighted with an orange box. The repository page includes navigation links for Code, Issues, Pull requests, Actions, Projects, Wiki, and Security. The file 'config.yml' is 8 lines (8 loc) and 305 Bytes.



Bootstrap package

The code - Outset script

- Login privileged once script
- Included as flat-pkg in the main bootstrap.pkg



```
#!/bin/zsh
```

```
/usr/local/munki/managedsoftwareupdate --munkipkgsonly && /usr/local/munki/managedsoftwareupdate --installonly
```

A macOS client built from code



Bootstrap package

The code - Build script

zentralpro / mcac-bootstrap-pkg

Code Issues Pull requests Actions Projects Wiki Security

Files

main

Go to file

- .github
- outset_scripts/login-privileged-once
 - first_munki_run.zsh
- .gitignore
- README.md
- build.py**
- config.yml
- requirements.txt

mcac-bootstrap-pkg / build.py

np5 Fix S3 upload

Code Blame 182 lines (152 loc) · 5.82 KB

```
1 import boto3
2 from botocore.exceptions import ClientError
3 import os
4 import tempfile
5 import requests
6 import shutil
7 import subprocess
8 import yaml
9
10
11 def get_package_name(config):
12     return f"bootstrap-{config['version']}"
13
14
15 def download(url, dest):
16     r = requests.get(url, allow_redirects=True)
17     with open(dest, "wb") as f:
18         for chunk in r.iter_content(chunk_size=1024):
19             f.write(chunk)
20
21
22 def download_products(build_dir, config):
23     for key, product in config.get("products").items():
```

A macOS client built from code



Bootstrap package

The code - Build script

- Test if the package already exists on S3

A macOS client built from code

```
def build(config_file, outset_scripts_dir):
    config = read_config(config_file)
    package_name = get_package_name(config)
    s3_client = get_s3_client()
    if test_s3_object(s3_client, config, package_name):
        print(f"Package {package_name} already present in the bucket.")
        return
    with tempfile.TemporaryDirectory() as build_dir:
        product_build_dir = os.path.join(build_dir, "_product")
        os.mkdir(product_build_dir)
        packages = []
        for key, pkg in download_products(build_dir, config):
            content_dir = expand_product(build_dir, key, pkg)
            packages.extend(add_packages(product_build_dir, content_dir))
        packages.append(
            add_outset_scripts_package(build_dir, product_build_dir, config, outset_scripts_dir)
        )
        output_file = os.path.join(build_dir, f"bootstrap-{config['version']}.pkg")
        build_product(product_build_dir, config, packages, output_file)
        upload_s3_object(s3_client, config, output_file, package_name)
```



Bootstrap package

The code - Build script

- Test if the package already exists on S3
- Download and unpack Outset and Munki

A macOS client built from code

```
def build(config_file, outset_scripts_dir):
    config = read_config(config_file)
    package_name = get_package_name(config)
    s3_client = get_s3_client()
    if test_s3_object(s3_client, config, package_name):
        print(f"Package {package_name} already present in the bucket.")
        return
    with tempfile.TemporaryDirectory() as build_dir:
        product_build_dir = os.path.join(build_dir, "_product")
        os.mkdir(product_build_dir)
        packages = []
        for key, pkg in download_products(build_dir, config):
            content_dir = expand_product(build_dir, key, pkg)
            packages.extend(add_packages(product_build_dir, content_dir))
        packages.append(
            add_outset_scripts_package(build_dir, product_build_dir, config, outset_scripts_dir)
        )
        output_file = os.path.join(build_dir, f"bootstrap-{config['version']}.pkg")
        build_product(product_build_dir, config, packages, output_file)
        upload_s3_object(s3_client, config, output_file, package_name)
```



Bootstrap package

The code - Build script

- Test if the package already exists on S3
- Download and unpack Outset and Munki
- Build the outset script flat-pkg

A macOS client built from code

```
def build(config_file, outset_scripts_dir):
    config = read_config(config_file)
    package_name = get_package_name(config)
    s3_client = get_s3_client()
    if test_s3_object(s3_client, config, package_name):
        print(f"Package {package_name} already present in the bucket.")
        return
    with tempfile.TemporaryDirectory() as build_dir:
        product_build_dir = os.path.join(build_dir, "_product")
        os.mkdir(product_build_dir)
        packages = []
        for key, pkg in download_products(build_dir, config):
            content_dir = expand_product(build_dir, key, pkg)
            packages.extend(add_packages(product_build_dir, content_dir))
        packages.append(
            add_outset_scripts_package(build_dir, product_build_dir, config, outset_scripts_dir)
        )
        output_file = os.path.join(build_dir, f"bootstrap-{config['version']}.pkg")
        build_product(product_build_dir, config, packages, output_file)
        upload_s3_object(s3_client, config, output_file, package_name)
```




Bootstrap package

The code - Build script

- Test if the package already exists on S3
- Download and unpack Outset and Munki
- Build the outset script flat-pkg
- Build and sign the package with custom distribution file

A macOS client built from code

```
def build(config_file, outset_scripts_dir):
    config = read_config(config_file)
    package_name = get_package_name(config)
    s3_client = get_s3_client()
    if test_s3_object(s3_client, config, package_name):
        print(f"Package {package_name} already present in the bucket.")
        return
    with tempfile.TemporaryDirectory() as build_dir:
        product_build_dir = os.path.join(build_dir, "_product")
        os.mkdir(product_build_dir)
        packages = []
        for key, pkg in download_products(build_dir, config):
            content_dir = expand_product(build_dir, key, pkg)
            packages.extend(add_packages(product_build_dir, content_dir))
        packages.append(
            add_outset_scripts_package(build_dir, product_build_dir, config, outset_scripts_dir)
        )
        output_file = os.path.join(build_dir, f"bootstrap-{config['version']}.pkg")
        build_product(product_build_dir, config, packages, output_file)
        upload_s3_object(s3_client, config, output_file, package_name)
```



Bootstrap package

The code - Build script

- ▶ Test if the package already exists on S3
- ▶ Download and unpack Outset and Munki
- ▶ Build the outset script flat-pkg
- ▶ Build and sign the package with custom distribution file
- ▶ Upload package to S3 bucket

A macOS client built from code

```
def build(config_file, outset_scripts_dir):
    config = read_config(config_file)
    package_name = get_package_name(config)
    s3_client = get_s3_client()
    if test_s3_object(s3_client, config, package_name):
        print(f"Package {package_name} already present in the bucket.")
        return
    with tempfile.TemporaryDirectory() as build_dir:
        product_build_dir = os.path.join(build_dir, "_product")
        os.mkdir(product_build_dir)
        packages = []
        for key, pkg in download_products(build_dir, config):
            content_dir = expand_product(build_dir, key, pkg)
            packages.extend(add_packages(product_build_dir, content_dir))
        packages.append(
            add_outset_scripts_package(build_dir, product_build_dir, config, outset_scripts_dir)
        )
        output_file = os.path.join(build_dir, f"bootstrap-{config['version']}.pkg")
        build_product(product_build_dir, config, packages, output_file)
        upload_s3_object(s3_client, config, output_file, package_name)
```

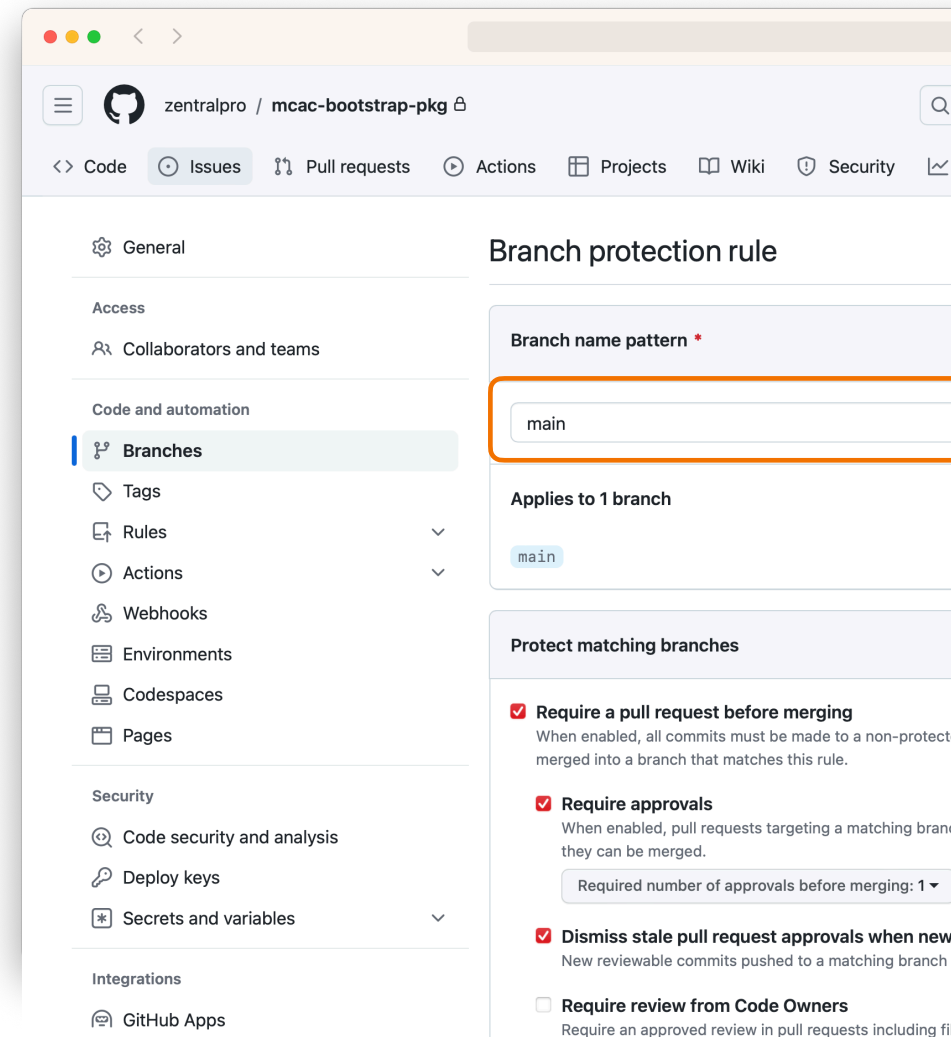


Bootstrap package

Repository configuration

- Main branch is production branch

A macOS client built from code



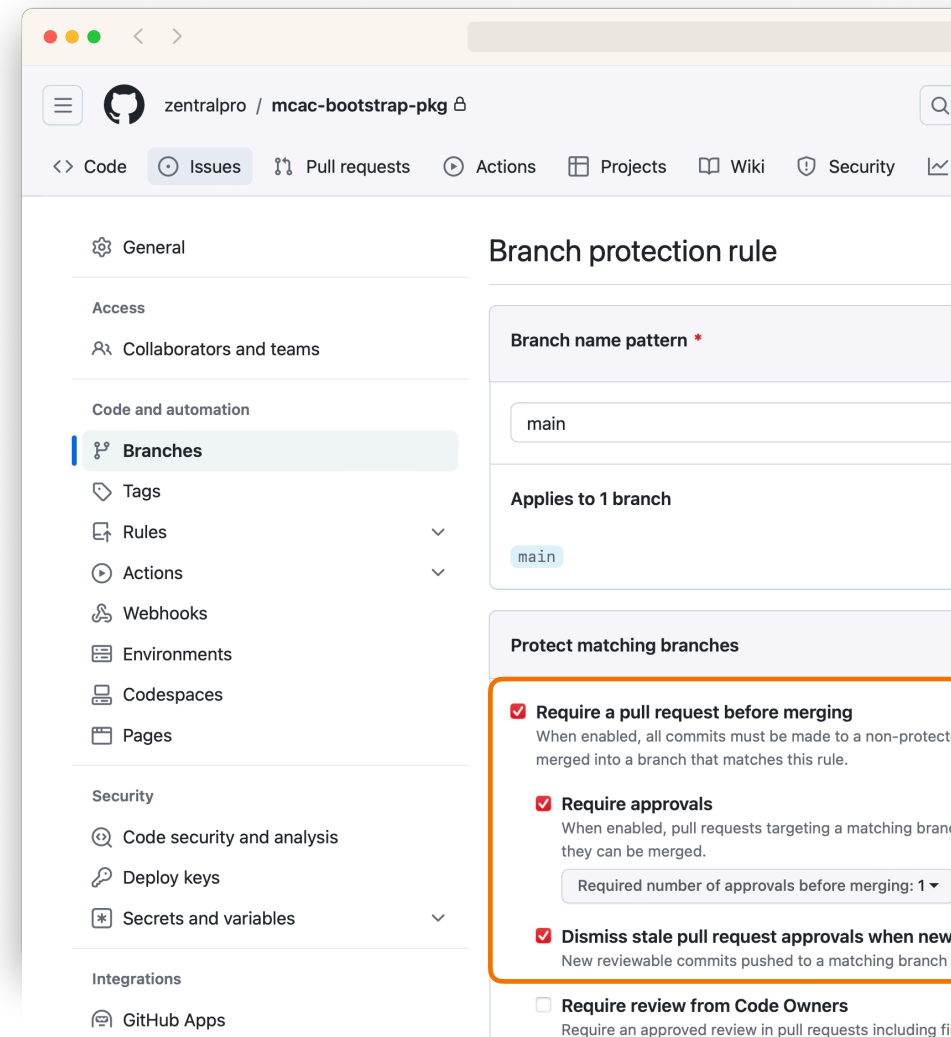


Bootstrap package

Repository configuration

- Main branch is production branch
- Pull requests with code reviews are mandatory to merge into main

A macOS client built from code



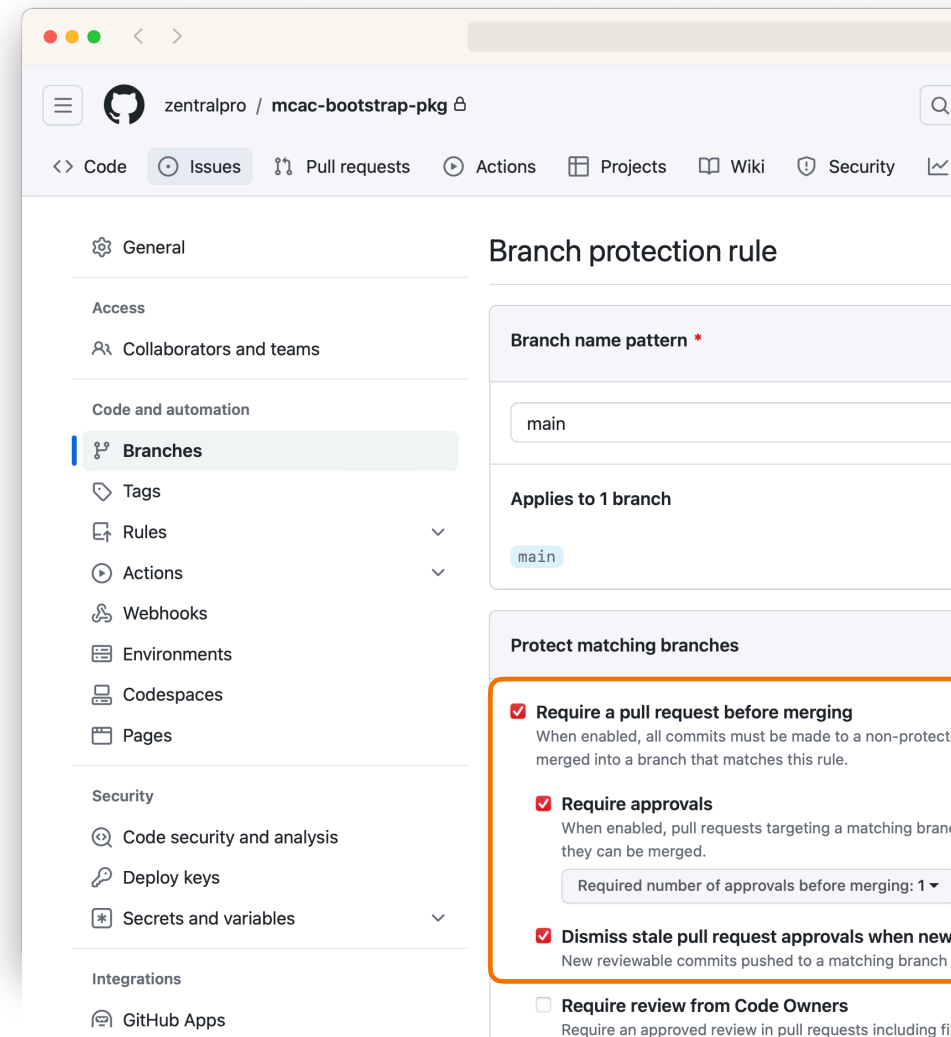


Bootstrap package

Repository configuration

- Main branch is production branch
- Pull requests with code reviews are mandatory to merge into main
- GitHub workflow triggered by merge into main

A macOS client built from code





Bootstrap package

The pipeline

A macOS client built from code

```
---
name: build

on:
  workflow_dispatch:

permissions:
  id-token: write
  contents: read

jobs:
  build:
    runs-on: [macos-latest]
    steps:
      - name: Checkout repository
        uses: actions/checkout@v4
      - name: Install the Apple certificate
        env:
          SIGNING_CERTIFICATE_BASE64: ${ secrets.SIGNING_CERTIFICATE_BASE64 }
          SIGNING_CERTIFICATE_PASSWORD: ${ secrets.SIGNING_CERTIFICATE_PASSWORD }
          KEYCHAIN_PASSWORD: ${ secrets.KEYCHAIN_PASSWORD }
        run: |
          # create variables
          CERTIFICATE_PATH=$RUNNER_TEMP/signing_certificate.cer
          KEYCHAIN_PATH=$RUNNER_TEMP/pkg-signing.keychain-db
          # import certificate from secret
```



Bootstrap package

The pipeline

- Code checkout

A macOS client built from code

```
---
name: build

on:
  workflow_dispatch:

permissions:
  id-token: write
  contents: read

jobs:
  build:
    runs-on: [macos-latest]
    steps:
      - name: Checkout repository
        uses: actions/checkout@v4
      - name: Install the Apple certificate
        env:
          SIGNING_CERTIFICATE_BASE64: ${ secrets.SIGNING_CERTIFICATE_BASE64 }
          SIGNING_CERTIFICATE_PASSWORD: ${ secrets.SIGNING_CERTIFICATE_PASSWORD }
          KEYCHAIN_PASSWORD: ${ secrets.KEYCHAIN_PASSWORD }
        run: |
          # create variables
          CERTIFICATE_PATH=$RUNNER_TEMP/signing_certificate.p12
          KEYCHAIN_PATH=$RUNNER_TEMP/pkg-signing.keychain-db

          # import certificate from secrets
          echo -n "$SIGNING_CERTIFICATE_BASE64" | base64 --decode -o $CERTIFICATE_PATH

          # create temporary keychain
          security create-keychain -p "$KEYCHAIN_PASSWORD" $KEYCHAIN_PATH
          security set-keychain-settings -lut 21600 $KEYCHAIN_PATH
```



Bootstrap package

The pipeline

- Code checkout
- Signing identity setup

A macOS client built from code

```
jobs:
  build:
    runs-on: [macos-latest]
    steps:
      - name: Checkout repository
        uses: actions/checkout@v4

      - name: Install the Apple certificate
        env:
          SIGNING_CERTIFICATE_BASE64: ${ secrets.SIGNING_CERTIFICATE_BASE64 }
          SIGNING_CERTIFICATE_PASSWORD: ${ secrets.SIGNING_CERTIFICATE_PASSWORD }
          KEYCHAIN_PASSWORD: ${ secrets.KEYCHAIN_PASSWORD }
        run: |
          # create variables
          CERTIFICATE_PATH=$RUNNER_TEMP/signing_certificate.p12
          KEYCHAIN_PATH=$RUNNER_TEMP/pkg-signing.keychain-db

          # import certificate from secrets
          echo -n "$SIGNING_CERTIFICATE_BASE64" | base64 --decode -o $CERTIFICATE_PATH

          # create temporary keychain
          security create-keychain -p "$KEYCHAIN_PASSWORD" $KEYCHAIN_PATH
          security set-keychain-settings -lut 21600 $KEYCHAIN_PATH
          security unlock-keychain -p "$KEYCHAIN_PASSWORD" $KEYCHAIN_PATH

          # import certificate to keychain
          security import $CERTIFICATE_PATH -P "$SIGNING_CERTIFICATE_PASSWORD" -A -t cert -f pkcs12
          security list-keychain -d user -s $KEYCHAIN_PATH

      - name: Setup Python
        uses: actions/setup-python@v4
        with:
          python-version: '3.10'

      - name: Install Python dependencies
        run: |
          python -m pip install --upgrade pip wheel
          pip install -U -r requirements.txt

      - name: Configure AWS Credentials
        uses: aws-actions/configure-aws-credentials@v1
        with:
          role-to-assume: ${ vars.AWS_ROLE_ARN }
          role-session-name: github-action
          aws-region: ${ vars.AWS_REGION }

      - name: Build and upload bootstrap package
```




Bootstrap package

The pipeline

- Code checkout
- Signing identity setup
- Python setup

A macOS client built from code

```
jobs:
  build:
    runs-on: [macos-latest]
    steps:
      - name: Checkout repository
        uses: actions/checkout@v4
      - name: Install the Apple certificate
        env:
          SIGNING_CERTIFICATE_BASE64: ${ secrets.SIGNING_CERTIFICATE_BASE64 }
          SIGNING_CERTIFICATE_PASSWORD: ${ secrets.SIGNING_CERTIFICATE_PASSWORD }
          KEYCHAIN_PASSWORD: ${ secrets.KEYCHAIN_PASSWORD }
        run: |
          # create variables
          CERTIFICATE_PATH=$RUNNER_TEMP/signing_certificate.p12
          KEYCHAIN_PATH=$RUNNER_TEMP/pkg-signing.keychain-db

          # import certificate from secrets
          echo -n "$SIGNING_CERTIFICATE_BASE64" | base64 --decode -o $CERTIFICATE_PATH

          # create temporary keychain
          security create-keychain -p "$KEYCHAIN_PASSWORD" $KEYCHAIN_PATH
          security set-keychain-settings -lut 21600 $KEYCHAIN_PATH
          security unlock-keychain -p "$KEYCHAIN_PASSWORD" $KEYCHAIN_PATH

          # import certificate to keychain
          security import $CERTIFICATE_PATH -P "$SIGNING_CERTIFICATE_PASSWORD" -A -t cert -f pkcs12
          security list-keychain -d user -s $KEYCHAIN_PATH

      - name: Setup Python
        uses: actions/setup-python@v4
        with:
          python-version: '3.10'
      - name: Install Python dependencies
        run: |
          python -m pip install --upgrade pip wheel
          pip install -U -r requirements.txt
      - name: Configure AWS Credentials
        uses: aws-actions/configure-aws-credentials@v1
        with:
          role-to-assume: ${ vars.AWS_ROLE_ARN }
          role-session-name: github-action
          aws-region: ${ vars.AWS_REGION }
      - name: Build and upload bootstrap package
```



Bootstrap package

The pipeline

- Code checkout
- Signing identity setup
- Python setup
- AWS credentials setup

A macOS client built from code

```
- name: Install the Apple certificate
  env:
    SIGNING_CERTIFICATE_BASE64: ${ secrets.SIGNING_CERTIFICATE_BASE64 }
    SIGNING_CERTIFICATE_PASSWORD: ${ secrets.SIGNING_CERTIFICATE_PASSWORD }
    KEYCHAIN_PASSWORD: ${ secrets.KEYCHAIN_PASSWORD }
  run: |
    # create variables
    CERTIFICATE_PATH=$RUNNER_TEMP/signing_certificate.p12
    KEYCHAIN_PATH=$RUNNER_TEMP/pkg-signing.keychain-db

    # import certificate from secrets
    echo -n "$SIGNING_CERTIFICATE_BASE64" | base64 --decode -o $CERTIFICATE_PATH

    # create temporary keychain
    security create-keychain -p "$KEYCHAIN_PASSWORD" $KEYCHAIN_PATH
    security set-keychain-settings -lut 21600 $KEYCHAIN_PATH
    security unlock-keychain -p "$KEYCHAIN_PASSWORD" $KEYCHAIN_PATH

    # import certificate to keychain
    security import $CERTIFICATE_PATH -P "$SIGNING_CERTIFICATE_PASSWORD" -A -t cert -f pkcs12
    security list-keychain -d user -s $KEYCHAIN_PATH

- name: Setup Python
  uses: actions/setup-python@v4
  with:
    python-version: '3.10'

- name: Install Python dependencies
  run: |
    python -m pip install --upgrade pip wheel
    pip install -U -r requirements.txt

- name: Configure AWS Credentials
  uses: aws-actions/configure-aws-credentials@v1
  with:
    role-to-assume: ${ vars.AWS_ROLE_ARN }
    role-session-name: github-action
    aws-region: ${ vars.AWS_REGION }

- name: Build and upload bootstrap package
  run: python build.py config.yml outset_scripts/
  env:
    AWS_BUCKET: ${ vars.AWS_BUCKET }
    AWS_BUCKET_PATH: ${ vars.AWS_BUCKET_PATH }
    SIGNING_ID: ${ vars.SIGNING_ID }
```



Bootstrap package

The pipeline

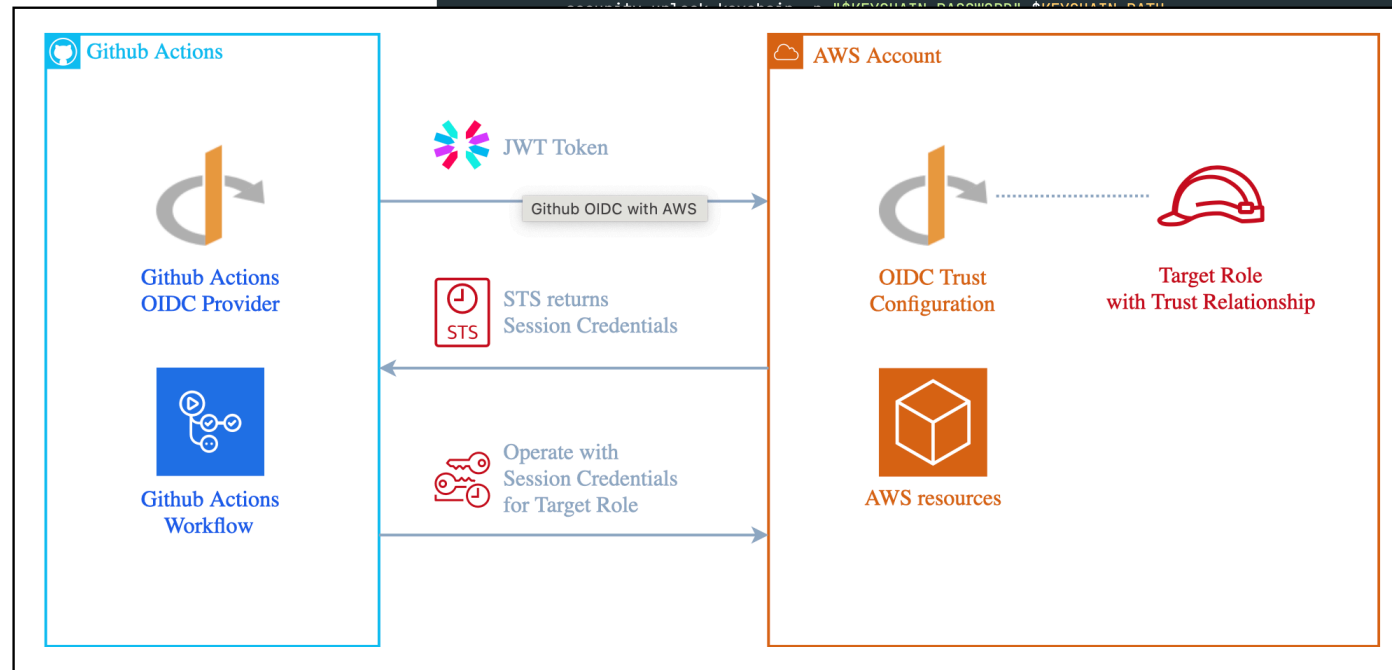
- Code checkout
- Signing identity setup
- Python setup
- AWS credentials setup

A macOS client built from code

```
- name: Install the Apple certificate
  env:
    SIGNING_CERTIFICATE_BASE64: ${ secrets.SIGNING_CERTIFICATE_BASE64 }
    SIGNING_CERTIFICATE_PASSWORD: ${ secrets.SIGNING_CERTIFICATE_PASSWORD }
    KEYCHAIN_PASSWORD: ${ secrets.KEYCHAIN_PASSWORD }
  run: |
    # create variables
    CERTIFICATE_PATH=$RUNNER_TEMP/signing_certificate.p12
    KEYCHAIN_PATH=$RUNNER_TEMP/pkg-signing.keychain-db

    # import certificate from secrets
    echo -n "$SIGNING_CERTIFICATE_BASE64" | base64 --decode -o $CERTIFICATE_PATH

    # create temporary keychain
    security create-keychain -p "$KEYCHAIN_PASSWORD" $KEYCHAIN_PATH
    security set-keychain-settings -lut 21600 $KEYCHAIN_PATH
    security set-keychain-lookup $KEYCHAIN_PATH
```



```
AWS_BUCKET: ${ vars.AWS_BUCKET }
AWS_BUCKET_PATH: ${ vars.AWS_BUCKET_PATH }
SIGNING_ID: ${ vars.SIGNING_ID }
```



Bootstrap package

The pipeline

- Code checkout
- Signing identity setup
- Python setup
- AWS credentials setup
- Build script run

A macOS client built from code

```
CERTIFICATE_PATH=$RUNNER_TEMP/signing_certificate.p12
KEYCHAIN_PATH=$RUNNER_TEMP/pkg-signing.keychain-db

# import certificate from secrets
echo -n "$SIGNING_CERTIFICATE_BASE64" | base64 --decode -o $CERTIFICATE_PATH

# create temporary keychain
security create-keychain -p "$KEYCHAIN_PASSWORD" $KEYCHAIN_PATH
security set-keychain-settings -lut 21600 $KEYCHAIN_PATH
security unlock-keychain -p "$KEYCHAIN_PASSWORD" $KEYCHAIN_PATH

# import certificate to keychain
security import $CERTIFICATE_PATH -P "$SIGNING_CERTIFICATE_PASSWORD" -A -t cert -f pkcs12
security list-keychain -d user -s $KEYCHAIN_PATH

- name: Setup Python
  uses: actions/setup-python@v4
  with:
    python-version: '3.10'

- name: Install Python dependencies
  run: |
    python -m pip install --upgrade pip wheel
    pip install -U -r requirements.txt

- name: Configure AWS Credentials
  uses: aws-actions/configure-aws-credentials@v1
  with:
    role-to-assume: ${vars.AWS_ROLE_ARN}
    role-session-name: github-action
    aws-region: ${vars.AWS_REGION}

- name: Build and upload bootstrap package
  run: python build.py config.yml outset_scripts/
  env:
    AWS_BUCKET: ${vars.AWS_BUCKET}
    AWS_BUCKET_PATH: ${vars.AWS_BUCKET_PATH}
    SIGNING_ID: ${vars.SIGNING_ID}
```



Munki repository

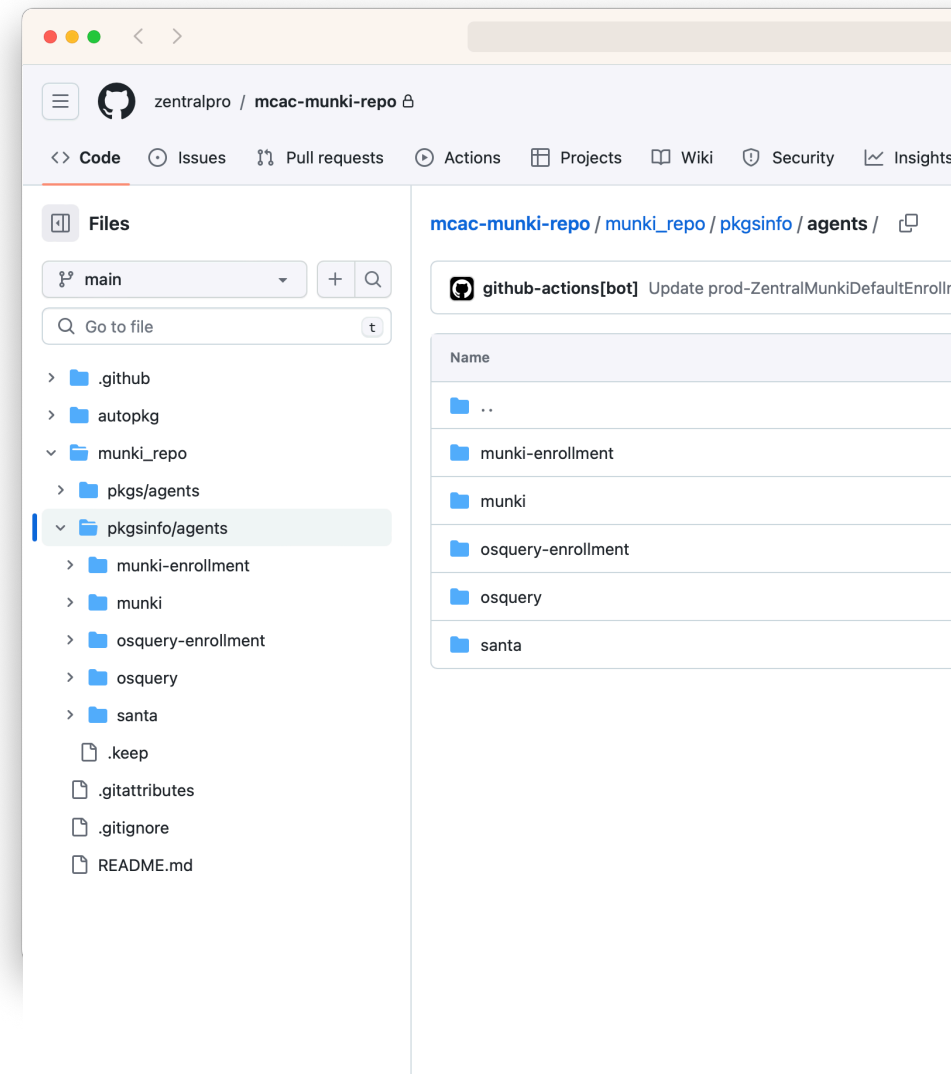


Munki repository

The package selection

- Patch management for user facing apps can be complex
- First: only the required agents and their enrollment packages
- Work in progress

A macOS client built from code





Munki repository

The inspiration

► Facebook: IT-CPE: `autopkg_tools.py`

A macOS client built from code

The screenshot shows the GitHub interface for the 'facebook / IT-CPE' repository. The 'Files' tab is active, displaying a directory tree. The 'legacy' folder is expanded, showing subfolders like 'adobe_tools', 'autodmg_cache_builder', and 'autopkg_tools'. The 'autopkg_tools' folder is selected, and the file 'autopkg_tools.py' is highlighted. On the right, the code editor shows the beginning of the 'autopkg_tools.py' file, which is a Python script for managing tools. The code includes a shebang, copyright notice, docstring, and imports for sys, imp, subprocess, os, json, time, and argparse. A 'try:' block is also visible.

```
1  #!/usr/bin/python
2  # Copyright (c) Facebook, Inc. and its affiliates
3  """Tools to manage Munki repositories"""
4
5  import sys
6  import imp
7  import subprocess
8  import os
9  import json
10 import time
11 import argparse
12
13 try:
14     import yaml
15     YAML_INSTALLED = True
16 except ImportError:
17     YAML_INSTALLED = False
```

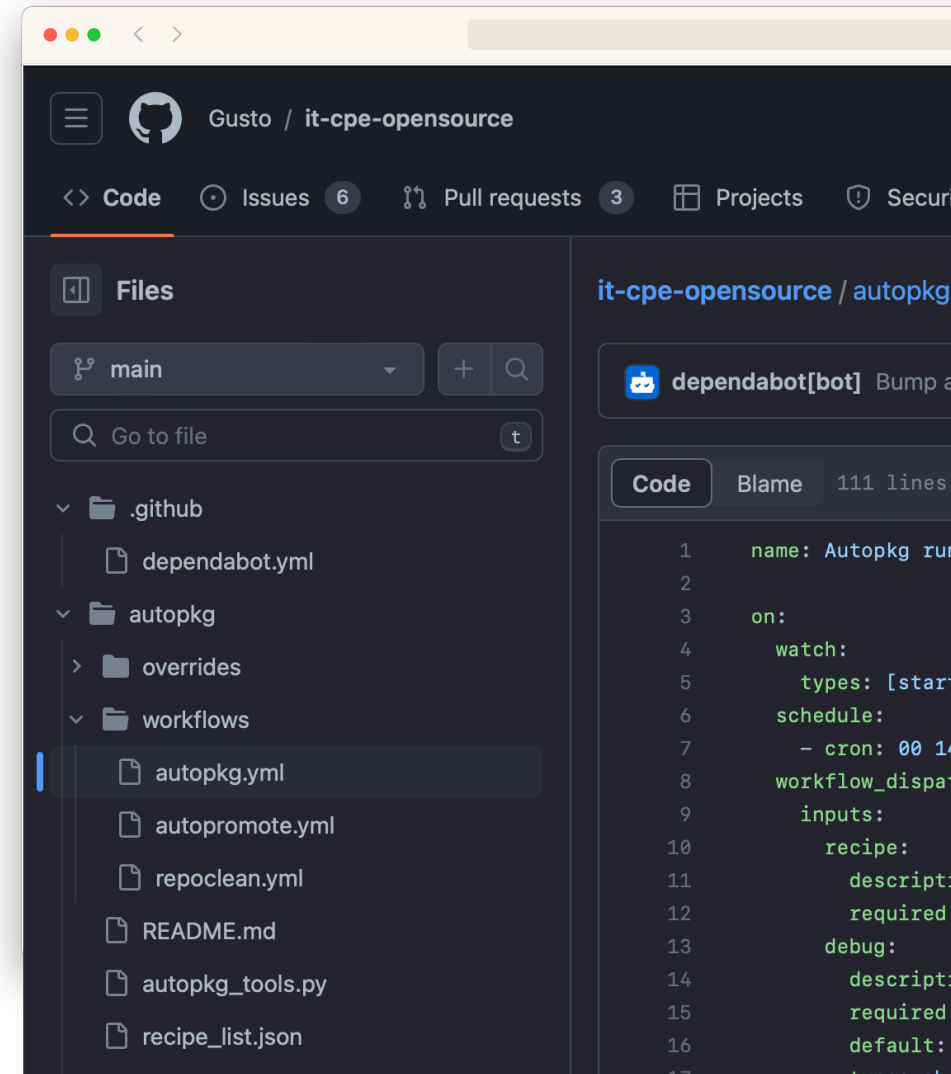


Munki repository

The inspiration

- Facebook: IT-CPE: `autopkg_tools.py`
- Gusto: **GitHub Actions**

A macOS client built from code



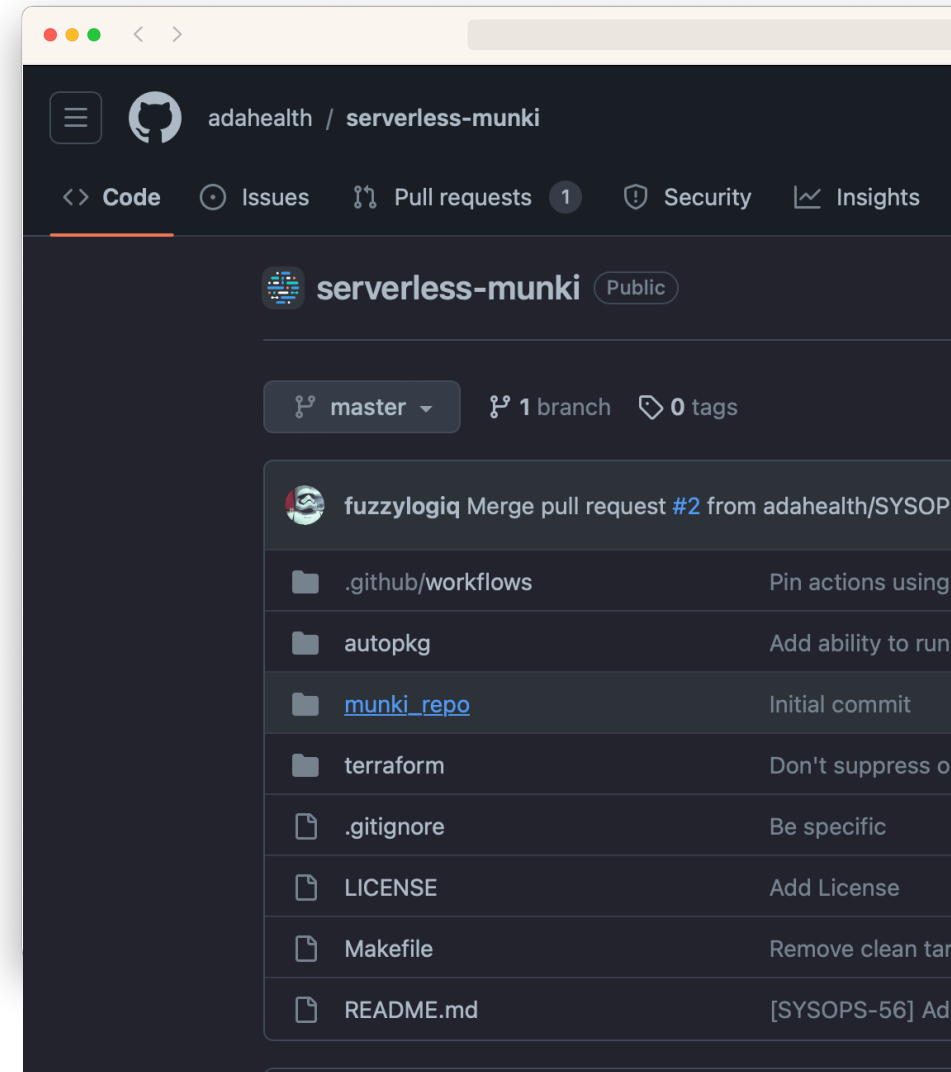


Munki repository

The inspiration

- Facebook: IT-CPE: `autopkg_tools.py`
- Gusto: **GitHub Actions**
- Ben Goodstein/AdaHealth: **serverless-munki-project**

A macOS client built from code



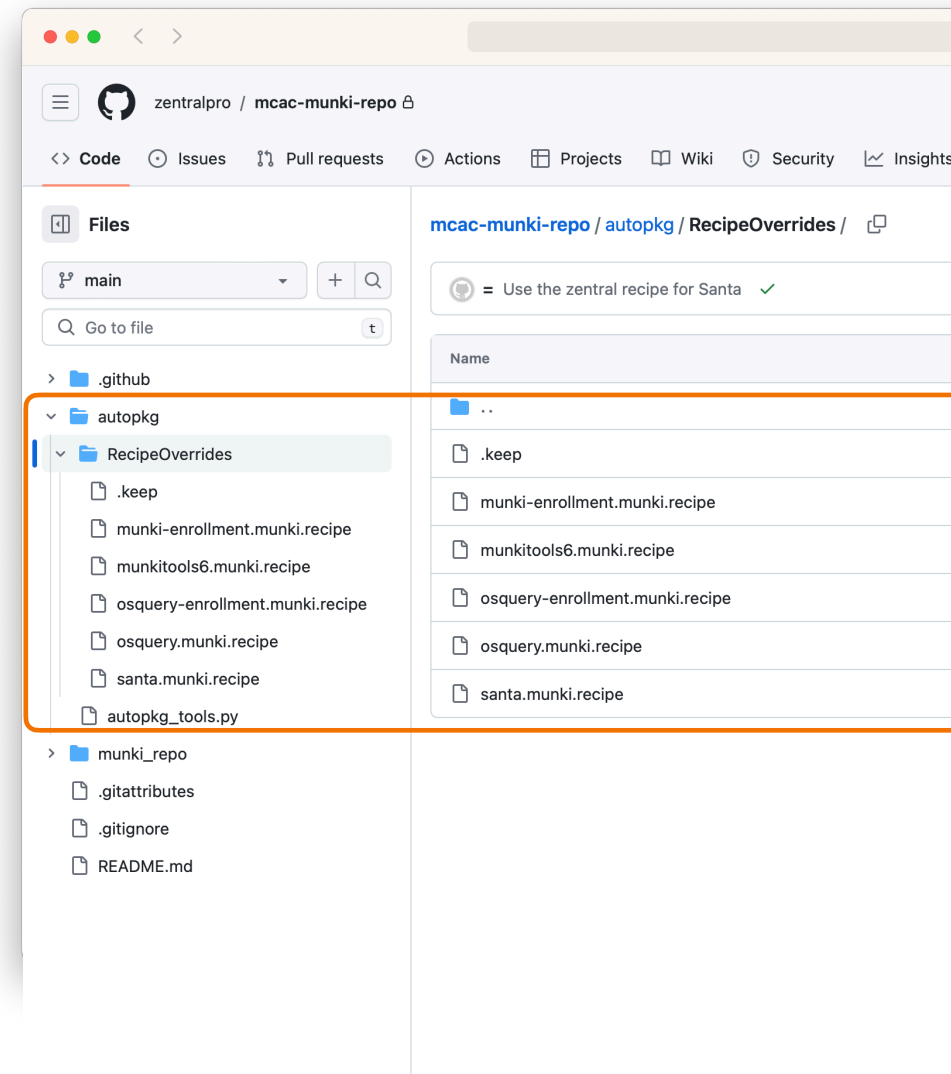


Munki repository

AutoPkg

- ▶ Automatically download process and import packages in a munki repo
- ▶ Community recipes exist for most of the software

A macOS client built from code



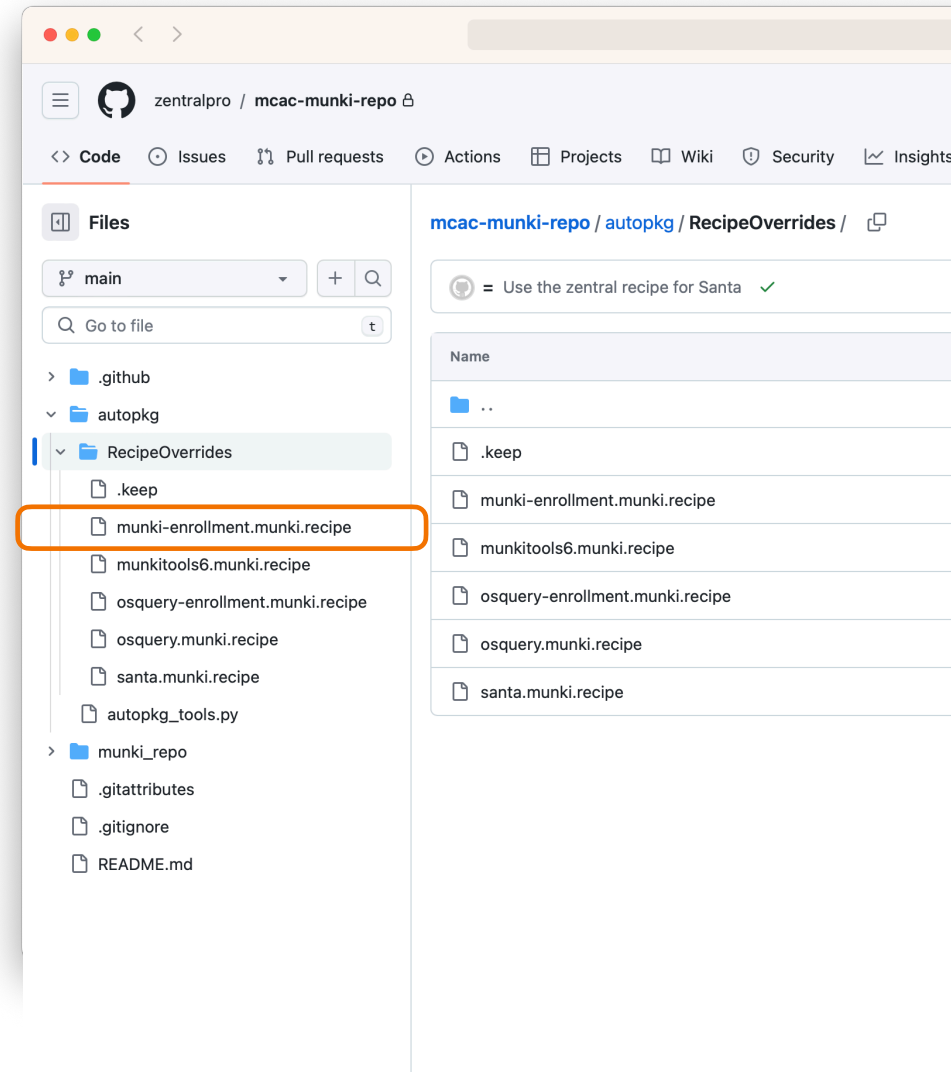


Munki repository

AutoPkg

- ▶ Automatically download process and import packages in a munki repo
- ▶ Community recipes exist for most of the software
- ▶ To add a package to our repository we make an override

A macOS client built from code





Munki repository

AutoPkg

- ▶ Automatically download process and import packages in a munki repo
- ▶ Community recipes exist for most of the software
- ▶ To add a package to our repository we make an override
- ▶ Custom recipes for our enrollment packages

A macOS client built from code

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>Identifier</key>
  <string>local.munki.ZentralEnroll</string>
  <key>Input</key>
  <dict>
    <key>MUNKI_CATEGORY</key>
    <string>IT Required</string>
    <key>MUNKI_DEVELOPER</key>
    <string>Zentral Pro Services GmbH</string>
    <key>MUNKI_REPO_SUBDIR</key>
    <string>agents/munki-enrollment</string>
    <key>PREFIX</key>
    <string>prod</string>
    <key>enrollment</key>
    <string>Munki</string>
    <key>enrollment_id</key>
    <string>1</string>
    <key>pkginfo</key>
  </dict>
  <key>catalogs</key>

```



Munki repository

autopkg_tools.py

A macOS client built from code

The screenshot shows a GitHub repository page for 'mcac-munki-repo' by 'zentralpro'. The file browser on the left shows the directory structure, with 'autopkg_tools.py' highlighted. The code editor on the right shows the content of 'autopkg_tools.py', which is a Python script for handling AutoPKG operations. The script includes imports for 'os', 'json', 'requests', 'subprocess', and 'plistlib'. It defines environment variables for 'WEBHOOK_URL', 'GIT', 'GITHUB_CLI', 'REPO_DIR', 'GITHUB_TOKEN', and 'INPUT_RECIPES'. It also defines two exception classes: 'Error' and 'GitError'.

```
1  #!/usr/bin/python3
2  # Copyright (c) Facebook, Inc. and its affiliates
3  # Modifications copyright (C) 2021 Ada Health Gr
4
5  """Wrapper script for handling AutoPKG operation
6
7  import os
8  import json
9  import requests
10 import subprocess
11 import plistlib
12
13
14 WEBHOOK_URL = os.environ['SLACK_WEBHOOK']
15 GIT = "/usr/bin/git"
16 GITHUB_CLI = "gh"
17 REPO_DIR = os.environ['GITHUB_WORKSPACE'] + "/m
18 GITHUB_TOKEN = os.environ['GITHUB_TOKEN']
19 INPUT_RECIPES = os.environ['INPUT_RECIPES'].spl
20
21
22 class Error(Exception):
23     """Base class for domain-specific exceptions
24
25
26 class GitError(Error):
27     """Git exceptions."""
28
29
```



Munki repository

autopkg_tools.py

- Run each recipe individually

A macOS client built from code

```
def handle_recipes():
    imported = []
    failed = []
    git_errors = []
    if INPUT_RECIPES:
        recipes = INPUT_RECIPES
    else:
        recipes = get_recipes()
    for recipe in recipes:
        # Parse the recipe name for basic item name
        branchname = parse_recipe_name(recipe)
        # Create new branch for item
        create_feature_branch(branchname)
        # Run Autopkg
        autopkg_run(recipe)
        # Parse the results from report plist
        run_results = parse_report_plist("report.plist")
        if not run_results['imported'] and not run_results['failed']:
            # Nothing happened
            continue
        if run_results['failed']:
            # Add to list of failed items
            failed.append(run_results['failed'][0])
        if run_results['imported']:
```



Munki repository

autopkg_tools.py

- Run each recipe individually
- Create a feature branch

A macOS client built from code

```
def handle_recipes():
    imported = []
    failed = []
    git_errors = []
    if INPUT_RECIPES:
        recipes = INPUT_RECIPES
    else:
        recipes = get_recipes()
    for recipe in recipes:
        # Parse the recipe name for basic item name
        branchname = parse_recipe_name(recipe)
        # Create new branch for item
        create_feature_branch(branchname)
        # Run Autopkg
        autopkg_run(recipe)
        # Parse the results from report plist
        run_results = parse_report_plist("report.plist")
        if not run_results['imported'] and not run_results['failed']:
            # Nothing happened
            continue
        if run_results['failed']:
            # Add to list of failed items
            failed.append(run_results['failed'][0])
        if run_results['imported']:
```



Munki repository

autopkg_tools.py

- Run each recipe individually
- Create a feature branch
- Run AutoPkg for the recipe

A macOS client built from code

```
def handle_recipes():
    imported = []
    failed = []
    git_errors = []
    if INPUT_RECIPES:
        recipes = INPUT_RECIPES
    else:
        recipes = get_recipes()
    for recipe in recipes:
        # Parse the recipe name for basic item name
        branchname = parse_recipe_name(recipe)
        # Create new branch for item
        create_feature_branch(branchname)
        # Run Autopkg
        autopkg_run(recipe)
        # Parse the results from report plist
        run_results = parse_report_plist("report.plist")
        if not run_results['imported'] and not run_results['failed']:
            # Nothing happened
            continue
        if run_results['failed']:
            # Add to list of failed items
            failed.append(run_results['failed'][0])
        if run_results['imported']:
```




Munki repository

autopkg_tools.py

- Run each recipe individually
- Create a feature branch
- Run AutoPkg for the recipe
- If a package was imported, commit the branch and make a PR

A macOS client built from code

```
if not run_results['imported'] and not run_results['failed']:
    # Nothing happened
    continue
if run_results['failed']:
    # Add to list of failed items
    failed.append(run_results['failed'][0])
if run_results['imported']:
    # Commit changes
    create_commit(run_results['imported'][0])
    branch_version = rename_branch_version(
        branchname,
        str(run_results['imported'][0]['version'])
    )
    # Push changes to github
    push_result = git_push(branch_version)
    if not push_result['success']:
        # This means there was a problem pushing changes to github
        # Add to list of git errors
        git_errors.append(push_result)
    else:
        # If push was successful then create a PR
        pull_request(branch_version)
        # Add basic item name to imported results so we can tell
        # the difference between arm and intel items
        run_results['imported'][0]['branchname'] = branchname
        # Add to list of imported items
        imported.append(run_results['imported'][0])

if not WEBHOOK_URL:
    print("Slack Webhook not set.. No notification sent.")
    return
```



Munki repository

autopkg_tools.py

- ▶ Run each recipe individually
- ▶ Create a feature branch
- ▶ Run AutoPkg for the recipe
- ▶ If a package was imported, commit the branch and make a PR
- ▶ Switch back to main branch before next iteration

A macOS client built from code

```
if not run_results['imported'] and not run_results['failed']:
    # Nothing happened
    continue
if run_results['failed']:
    # Add to list of failed items
    failed.append(run_results['failed'][0])
if run_results['imported']:
    # Commit changes
    create_commit(run_results['imported'][0])
    branch_version = rename_branch_version(
        branchname,
        str(run_results['imported'][0]['version'])
    )
    # Push changes to github
    push_result = git_push(branch_version)
    if not push_result['success']:
        # This means there was a problem pushing changes to github
        # Add to list of git errors
        git_errors.append(push_result)
    else:
        # If push was successful then create a PR
        pull_request(branch_version)
        # Add basic item name to imported results so we can tell
        # the difference between arm and intel items
        run_results['imported'][0]['branchname'] = branchname
        # Add to list of imported items
        imported.append(run_results['imported'][0])

if not WEBHOOK_URL:
    print("Slack Webhook not set.. No notification sent.")
    return
```



Munki repository

The CI/CD pipeline - AutoPkg run

A macOS client built from code

```
1  name: AutoPkg Run
2
3  on:
4    schedule:
5      - cron: '0 6 * * MON-FRI'
6    workflow_dispatch:
7      inputs:
8        recipes:
9          description: List of recipes to run separately
10         required: False
11
12  jobs:
13    autopkg-run:
14      runs-on: macos-latest
15      steps:
16        - uses: actions/checkout@v4
17        - name: Install python dependencies
18          run: |
19            python3 -m pip install --upgrade pip
20            pip3 install requests
21        - name: Install Munki
22          run: |
23            curl -OL https://github.com/macadmins/munki/releases/download/v2.12.0/munkitools-6.3.3.4593.pkg
24            sudo installer -pkg munkitools-6.3.3.4593.pkg
25            sudo rm munkitools-6.3.3.4593.pkg
26        - name: Install AutoPkg
27          run: |
28            curl -OL https://github.com/autopkg/autopkg/releases/download/v2.7.2/autopkg-2.7.2.pkg
29            sudo installer -pkg autopkg-2.7.2.pkg
```



Munki repository

The CI/CD pipeline - AutoPkg run

- Scheduled to run daily

A macOS client built from code

```
name: AutoPkg Run

on:
  schedule:
    - cron: '0 6 * * MON-FRI'
  workflow_dispatch:
    inputs:
      recipes:
        description: List of recipes to run separated by spaces
        required: False

jobs:
  autopkg-run:
    runs-on: macos-latest
    steps:
      - uses: actions/checkout@v4
      - name: Install python dependencies
        run: |
          python3 -m pip install --upgrade pip
          pip3 install requests
      - name: Install Munki
        run: |
          curl -OL https://github.com/macadmins/munki-builds/releases/download/v6.3.3.
          sudo installer -pkg munkitools-6.3.3.4593.pkg -target /
          sudo rm munkitools-6.3.3.4593.pkg
      - name: Install AutoPkg
        run: |
          curl -OL https://github.com/autopkg/autopkg/releases/download/v2.7.2/autopkg
          sudo installer -pkg autopkg-2.7.2.pkg -target /
          sudo rm autopkg-2.7.2.pkg
      - name: Configure AutoPkg
        env:
          TOKEN: ${ secrets.GITHUB_TOKEN }
        run: |
```



Munki repository

The CI/CD pipeline - AutoPkg run

- Scheduled to run daily
- Munki and AutoPkg are added and configured

A macOS client built from code

```
name: AutoPkg Run

on:
  schedule:
    - cron: '0 6 * * MON-FRI'
  workflow_dispatch:
  inputs:
    recipes:
      description: List of recipes to run separated by spaces
      required: False

jobs:
  autopkg-run:
    runs-on: macos-latest
    steps:
      - uses: actions/checkout@v4
      - name: Install python dependencies
        run: |
          python3 -m pip install --upgrade pip
          pip3 install requests
      - name: Install Munki
        run: |
          curl -OL https://github.com/macadmins/munki-builds/releases/download/v6.3.3.
          sudo installer -pkg munkitools-6.3.3.4593.pkg -target /
          sudo rm munkitools-6.3.3.4593.pkg
      - name: Install AutoPkg
        run: |
          curl -OL https://github.com/autopkg/autopkg/releases/download/v2.7.2/autopkg
          sudo installer -pkg autopkg-2.7.2.pkg -target /
          sudo rm autopkg-2.7.2.pkg
      - name: Configure AutoPkg
        env:
          TOKEN: ${ secrets.GITHUB_TOKEN }
        run: |
```



Munki repository

The CI/CD pipeline - AutoPkg run

- Scheduled to run daily
- Munki and AutoPkg are added and configured
- The Munki catalogs are reconstructed

A macOS client built from code

```
sudo installer -pkg munkitools-6.3.3.4593.pkg -target /
sudo rm munkitools-6.3.3.4593.pkg

- name: Install AutoPkg
  run: |
    curl -OL https://github.com/autopkg/autopkg/releases/download/v2.7.2/autopkg-2.7.2.pkg
    sudo installer -pkg autopkg-2.7.2.pkg -target /
    sudo rm autopkg-2.7.2.pkg

- name: Configure AutoPkg
  env:
    TOKEN: ${ secrets.GITHUB_TOKEN }
  run: |
    defaults write com.github.autopkg RECIPE_OVERRIDE_DIRS "$GITHUB_WORKSPACE"/a
    defaults write com.github.autopkg RECIPE_SEARCH_DIRS "$GITHUB_WORKSPACE"/a
    defaults write com.github.autopkg FAIL_RECIPES_WITHOUT_TRUST_INFO -bool YES
    defaults write com.github.autopkg MUNKI_REPO "$GITHUB_WORKSPACE"/munki_repo
    defaults write com.github.autopkg GITHUB_TOKEN "$TOKEN"

# Add any required parent repos for your recipes here.

- name: Add AutoPkg repos
  run: |
    autopkg repo-add recipes
    autopkg repo-add jbakker10-recipes
    autopkg repo-add zentral-recipes

- name: Run makecatalogs
  run: /usr/local/munki/makecatalogs munki_repo

- name: Run AutoPkg
  run: python3 autopkg/autopkg_tools.py
  env:
    SLACK_WEBHOOK: ${ secrets.SLACK_WEBHOOK }
    GITHUB_TOKEN: ${ secrets.GITHUB_TOKEN }
    INPUT_RECIPES: ${ github.event.inputs.recipes }
    ZTL_API_TOKEN: ${ secrets.ZTL_API_TOKEN }
    ZTL_FQDN: ${ vars.ZTL_FQDN }
```

26 - name: Install AutoPkg
27 run: |
28 curl -OL https://github.com/autopkg/a
29 sudo installer -pkg autopkg-2.7.2.pkg



Munki repository

The CI/CD pipeline - AutoPkg run

- Scheduled to run daily
- Munki and AutoPkg are added and configured
- The Munki catalogs are reconstructed
- autopkg_tools.py is run

A macOS client built from code

```
curl -OL https://github.com/munki/munki/releases/download/v2.7.2/autopkg-2.7.2.pkg
sudo installer -pkg munkitools-6.3.3.4593.pkg -target /
sudo rm munkitools-6.3.3.4593.pkg

- name: Install AutoPkg
  run: |
    curl -OL https://github.com/autopkg/autopkg/releases/download/v2.7.2/autopkg-2.7.2.pkg
    sudo installer -pkg autopkg-2.7.2.pkg -target /
    sudo rm autopkg-2.7.2.pkg

- name: Configure AutoPkg
  env:
    TOKEN: ${ secrets.GITHUB_TOKEN }
  run: |
    defaults write com.github.autopkg RECIPE_OVERRIDE_DIRS "$GITHUB_WORKSPACE"/a
    defaults write com.github.autopkg RECIPE_SEARCH_DIRS "$GITHUB_WORKSPACE"/a
    defaults write com.github.autopkg FAIL_RECIPES_WITHOUT_TRUST_INFO -bool YES
    defaults write com.github.autopkg MUNKI_REPO "$GITHUB_WORKSPACE"/munki_repo
    defaults write com.github.autopkg GITHUB_TOKEN "$TOKEN"

# Add any required parent repos for your recipes here.

- name: Add AutoPkg repos
  run: |
    autopkg repo-add recipes
    autopkg repo-add jbakker10-recipes
    autopkg repo-add zentral-recipes

- name: Run makecatalogs
  run: /usr/local/munki/makecatalogs munki_repo

- name: Run AutoPkg
  run: python3 autopkg/autopkg_tools.py
  env:
    SLACK_WEBHOOK: ${ secrets.SLACK_WEBHOOK }
    GITHUB_TOKEN: ${ secrets.GITHUB_TOKEN }
    INPUT_RECIPES: ${ github.event.inputs.recipes }
    ZTL_API_TOKEN: ${ secrets.ZTL_API_TOKEN }
    ZTL_FQDN: ${ vars.ZTL_FQDN }

26 - name: Install AutoPkg
27   run: |
28     curl -OL https://github.com/autopkg/autopkg/releases/download/v2.7.2/autopkg-2.7.2.pkg
29     sudo installer -pkg autopkg-2.7.2.pkg
```



Munki repository

The CI/CD pipeline - S3 sync

zentralpro / mcac-munki-repo

Code Issues Pull requests Actions Projects Wiki Security Insights

Files

main

Go to file

- .github/workflows
 - autopkg-run.yml
 - clean-repo.yml
 - sync-repo.yml**
 - update-trust-info.yml
- autopkg
 - RecipeOverrides
 - autopkg_tools.py
- munki_repo
 - pkgs/agents
 - pkgsinfo/agents
 - .keep
 - .gitattributes
 - .gitignore
 - README.md

mcac-munki-repo / .github / workflows / sync-repo.yml

Add poke Zentral server step

Code Blame 63 lines (60 loc) · 2 KB Code 55% f

```
1  name: Sync Munki Repo
2
3  on:
4    push:
5      paths:
6        - 'munki_repo/**'
7      branches:
8        - main
9    workflow_dispatch:
10
11  permissions:
12    id-token: write
13    contents: read
14
15  jobs:
16    sync-repo:
17      runs-on: macos-latest
18      steps:
19        - uses: actions/checkout@v4
20          with:
21            lfs: 'true'
22            fetch-depth: 1
23        - name: Install AWS CLI
24          run: |
25            curl "https://awscli.amazonaws.com/AWSCLI-v2-*.exe" -o awscli.exe
26            sudo installer -pkg ./AWSCLIV2.pkg -t /
27            sudo rm AWSCLIV2.pkg
28        - name: Install Munki
29          run: |
```

A macOS client built from code



Munki repository

The CI/CD pipeline - S3 sync

- Scheduled on each merge in the main branch

A macOS client built from code

```
name: Sync Munki Repo

on:
  push:
    paths:
      - 'munki_repo/**'
    branches:
      - main
  workflow_dispatch:

permissions:
  id-token: write
  contents: read

jobs:
  sync-repo:
    runs-on: macos-latest
    steps:
      - uses: actions/checkout@v4
        with:
          lfs: 'true'
          fetch-depth: 1
      - name: Install AWS CLI
        run: |
          curl "https://awscli.amazonaws.com/AWSCLIV2.pkg" -o "AWSCLIV2.pkg"
          sudo installer -pkg ./AWSCLIV2.pkg -target /
          sudo rm AWSCLIV2.pkg
      - name: Install Munki
        run: |
          curl -OL https://github.com/macadmins/munki-builds/releases/download/v6.3.3.4593/munki-6.3.3.4593.pkg
          sudo installer -pkg munkitools-6.3.3.4593.pkg -target /
          sudo rm munkitools-6.3.3.4593.pkg
      - name: Run makecatalogs
        run: |
          /usr/local/munki/makecatalogs "$GITHUB_WORKSPACE"/munki_repo
```



Munki repository

The CI/CD pipeline - S3 sync

- ▶ Scheduled on each merge in the main branch
- ▶ Same steps install and configure the required tools

A macOS client built from code

```
jobs:
  sync-repo:
    runs-on: macos-latest
    steps:
      - uses: actions/checkout@v4
        with:
          lfs: 'true'
          fetch-depth: 1
      - name: Install AWS CLI
        run: |
          curl "https://awscli.amazonaws.com/AWSCLIV2.pkg" -o "AWSCLIV2.pkg"
          sudo installer -pkg ./AWSCLIV2.pkg -target /
          sudo rm AWSCLIV2.pkg
      - name: Install Munki
        run: |
          curl -OL https://github.com/macadmins/munki-builds/releases/download/v6.3.3.4593/munki-tools-6.3.3.4593.pkg
          sudo installer -pkg munkitools-6.3.3.4593.pkg -target /
          sudo rm munkitools-6.3.3.4593.pkg
      - name: Run makecatalogs
        run: |
          /usr/local/munki/makecatalogs "$GITHUB_WORKSPACE"/munki_repo
      - name: Configure AWS Credentials
        uses: aws-actions/configure-aws-credentials@v1
        with:
          role-to-assume: ${ vars.AWS_ROLE_ARN }
          role-session-name: github-action
          aws-region: ${ vars.AWS_REGION }
      - name: Run AWS S3 sync
        run: |
          aws s3 sync "$GITHUB_WORKSPACE"/munki_repo ${ vars.MUNKI_REPO_URI } --exclude '.DS_Store'
      - name: Poke Zentral server
        run: |
          curl -X POST -H "Authorization: Token $ZTL_API_TOKEN" https://$ZTL_FQDN/api/monolith
        env:
          ZTL_FQDN: ${ vars.ZTL_FQDN }
          ZTL_API_TOKEN: ${ secrets.ZTL_API_TOKEN }
  slack-notify:
    runs-on: ubuntu-latest
    needs: sync-repo
```



Munki repository

The CI/CD pipeline - S3 sync

- ▶ Scheduled on each merge in the main branch
- ▶ Same steps install and configure the required tools
- ▶ AWS CLI, Munki are installed

A macOS client built from code

```
jobs:
  sync-repo:
    runs-on: macos-latest
    steps:
      - uses: actions/checkout@v4
        with:
          lfs: 'true'
          fetch-depth: 1
      - name: Install AWS CLI
        run: |
          curl "https://awscli.amazonaws.com/AWSCLIV2.pkg" -o "AWSCLIV2.pkg"
          sudo installer -pkg ./AWSCLIV2.pkg -target /
          sudo rm AWSCLIV2.pkg
      - name: Install Munki
        run: |
          curl -OL https://github.com/macadmins/munki-builds/releases/download/v6.3.3.4593/munki-tools-6.3.3.4593.pkg
          sudo installer -pkg munkitools-6.3.3.4593.pkg -target /
          sudo rm munkitools-6.3.3.4593.pkg
      - name: Run makecatalogs
        run: |
          /usr/local/munki/makecatalogs "$GITHUB_WORKSPACE"/munki_repo
      - name: Configure AWS Credentials
        uses: aws-actions/configure-aws-credentials@v1
        with:
          role-to-assume: ${ vars.AWS_ROLE_ARN }
          role-session-name: github-action
          aws-region: ${ vars.AWS_REGION }
      - name: Run AWS S3 sync
        run: |
          aws s3 sync "$GITHUB_WORKSPACE"/munki_repo ${ vars.MUNKI_REPO_URI } --exclude '.DS_Store'
      - name: Poke Zentral server
        run: |
          curl -X POST -H "Authorization: Token $ZTL_API_TOKEN" https://$ZTL_FQDN/api/monolith
        env:
          ZTL_FQDN: ${ vars.ZTL_FQDN }
          ZTL_API_TOKEN: ${ secrets.ZTL_API_TOKEN }
  slack-notify:
    runs-on: ubuntu-latest
    needs: sync-repo
```



Munki repository

The CI/CD pipeline - S3 sync

- ▶ Scheduled on each merge in the main branch
- ▶ Same steps install and configure the required tools
- ▶ AWS CLI, Munki are installed
- ▶ Catalogs are reconstructed

A macOS client built from code

```
jobs:
  sync-repo:
    runs-on: macos-latest
    steps:
      - uses: actions/checkout@v4
        with:
          lfs: 'true'
          fetch-depth: 1
      - name: Install AWS CLI
        run: |
          curl "https://awscli.amazonaws.com/AWSCLIV2.pkg" -o "AWSCLIV2.pkg"
          sudo installer -pkg ./AWSCLIV2.pkg -target /
          sudo rm AWSCLIV2.pkg
      - name: Install Munki
        run: |
          curl -OL https://github.com/macadmins/munki-builds/releases/download/v6.3.3.4593/munki-builds-6.3.3.4593.pkg
          sudo installer -pkg munkitools-6.3.3.4593.pkg -target /
          sudo rm munkitools-6.3.3.4593.pkg
      - name: Run makecatalogs
        run: |
          /usr/local/munki/makecatalogs "$GITHUB_WORKSPACE"/munki_repo
      - name: Configure AWS Credentials
        uses: aws-actions/configure-aws-credentials@v1
        with:
          role-to-assume: ${ vars.AWS_ROLE_ARN }
          role-session-name: github-action
          aws-region: ${ vars.AWS_REGION }
      - name: Run AWS S3 sync
        run: |
          aws s3 sync "$GITHUB_WORKSPACE"/munki_repo ${ vars.MUNKI_REPO_URI } --exclude '.DS_Store'
      - name: Poke Zentral server
        run: |
          curl -X POST -H "Authorization: Token $ZTL_API_TOKEN" https://$ZTL_FQDN/api/monolith
        env:
          ZTL_FQDN: ${ vars.ZTL_FQDN }
          ZTL_API_TOKEN: ${ secrets.ZTL_API_TOKEN }
  slack-notify:
    runs-on: ubuntu-latest
    needs: sync-repo
```



Munki repository

The CI/CD pipeline - S3 sync

- ▶ Scheduled on each merge in the main branch
- ▶ Same steps install and configure the required tools
- ▶ AWS CLI, Munki are installed
- ▶ Catalogs are reconstructed
- ▶ AWS credentials are configured using OIDC

A macOS client built from code

```
jobs:
  sync-repo:
    runs-on: macos-latest
    steps:
      - uses: actions/checkout@v4
        with:
          lfs: 'true'
          fetch-depth: 1
      - name: Install AWS CLI
        run: |
          curl "https://awscli.amazonaws.com/AWSCLIV2.pkg" -o "AWSCLIV2.pkg"
          sudo installer -pkg ./AWSCLIV2.pkg -target /
          sudo rm AWSCLIV2.pkg
      - name: Install Munki
        run: |
          curl -OL https://github.com/macadmins/munki-builds/releases/download/v6.3.3.4593/munki-6.3.3.4593.pkg
          sudo installer -pkg munkitools-6.3.3.4593.pkg -target /
          sudo rm munkitools-6.3.3.4593.pkg
      - name: Run makecatalogs
        run: |
          /usr/local/munki/makecatalogs "$GITHUB_WORKSPACE"/munki_repo
      - name: Configure AWS Credentials
        uses: aws-actions/configure-aws-credentials@v1
        with:
          role-to-assume: ${vars.AWS_ROLE_ARN}
          role-session-name: github-action
          aws-region: ${vars.AWS_REGION}
      - name: Run AWS S3 sync
        run: |
          aws s3 sync "$GITHUB_WORKSPACE"/munki_repo ${vars.MUNKI_REPO_URI} --exclude '.DS_Store'
      - name: Poke Zentral server
        run: |
          curl -X POST -H "Authorization: Token $ZTL_API_TOKEN" https://$ZTL_FQDN/api/monolith
        env:
          ZTL_FQDN: ${vars.ZTL_FQDN}
          ZTL_API_TOKEN: ${secrets.ZTL_API_TOKEN}
  slack-notify:
    runs-on: ubuntu-latest
    needs: sync-repo
```



Munki repository

The CI/CD pipeline - S3 sync

- ▶ Scheduled on each merge in the main branch
- ▶ Same steps install and configure the required tools
- ▶ AWS CLI, Munki are installed
- ▶ Catalogs are reconstructed
- ▶ AWS credentials are configured using OIDC
- ▶ S3 sync is triggered

A macOS client built from code

```
jobs:
  sync-repo:
    runs-on: macos-latest
    steps:
      - uses: actions/checkout@v4
        with:
          lfs: 'true'
          fetch-depth: 1
      - name: Install AWS CLI
        run: |
          curl "https://awscli.amazonaws.com/AWSCLIV2.pkg" -o "AWSCLIV2.pkg"
          sudo installer -pkg ./AWSCLIV2.pkg -target /
          sudo rm AWSCLIV2.pkg
      - name: Install Munki
        run: |
          curl -OL https://github.com/macadmins/munki-builds/releases/download/v6.3.3.4593/munki-6.3.3.4593.pkg
          sudo installer -pkg munkitools-6.3.3.4593.pkg -target /
          sudo rm munkitools-6.3.3.4593.pkg
      - name: Run makecatalogs
        run: |
          /usr/local/munki/makecatalogs "$GITHUB_WORKSPACE"/munki_repo
      - name: Configure AWS Credentials
        uses: aws-actions/configure-aws-credentials@v1
        with:
          role-to-assume: ${vars.AWS_ROLE_ARN}
          role-session-name: github-action
          aws-region: ${vars.AWS_REGION}
      - name: Run AWS S3 sync
        run: |
          aws s3 sync "$GITHUB_WORKSPACE"/munki_repo ${vars.MUNKI_REPO_URI} --exclude '*.DS_Store'
      - name: Poke Zentral server
        run: |
          curl -X POST -H "Authorization: Token $ZTL_API_TOKEN" https://$ZTL_FQDN/api/monolith
        env:
          ZTL_FQDN: ${vars.ZTL_FQDN}
          ZTL_API_TOKEN: ${secrets.ZTL_API_TOKEN}
  slack-notify:
    runs-on: ubuntu-latest
    needs: sync-repo
```



Munki repository

The CI/CD pipeline - S3 sync

- ▶ Scheduled on each merge in the main branch
- ▶ Same steps install and configure the required tools
- ▶ AWS CLI, Munki are installed
- ▶ Catalogs are reconstructed
- ▶ AWS credentials are configured using OIDC
- ▶ S3 sync is triggered
- ▶ Zentral is notified

A macOS client built from code

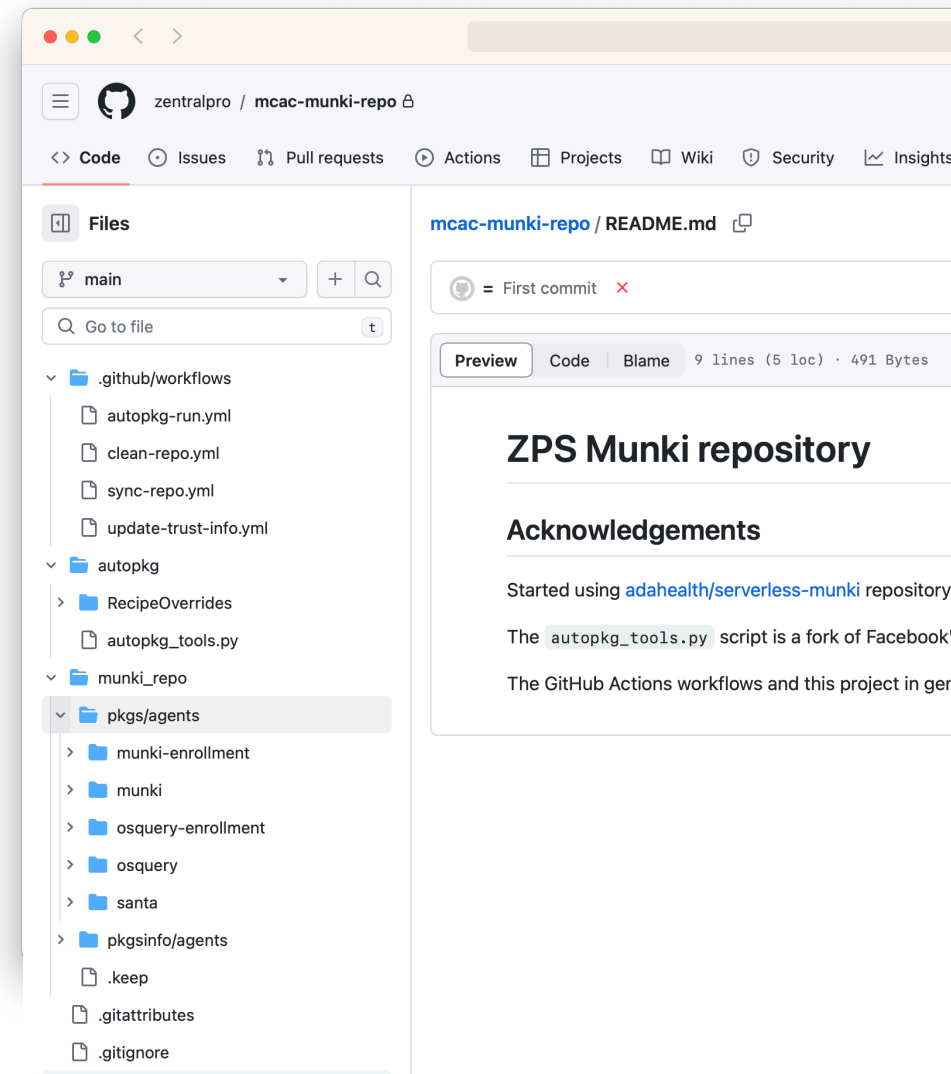
```
jobs:
  sync-repo:
    runs-on: macos-latest
    steps:
      - uses: actions/checkout@v4
        with:
          lfs: 'true'
          fetch-depth: 1
      - name: Install AWS CLI
        run: |
          curl "https://awscli.amazonaws.com/AWSCLIV2.pkg" -o "AWSCLIV2.pkg"
          sudo installer -pkg ./AWSCLIV2.pkg -target /
          sudo rm AWSCLIV2.pkg
      - name: Install Munki
        run: |
          curl -OL https://github.com/macadmins/munki-builds/releases/download/v6.3.3.4593/munki-6.3.3.4593.pkg
          sudo installer -pkg munkitools-6.3.3.4593.pkg -target /
          sudo rm munkitools-6.3.3.4593.pkg
      - name: Run makecatalogs
        run: |
          /usr/local/munki/makecatalogs "$GITHUB_WORKSPACE"/munki_repo
      - name: Configure AWS Credentials
        uses: aws-actions/configure-aws-credentials@v1
        with:
          role-to-assume: ${vars.AWS_ROLE_ARN}
          role-session-name: github-action
          aws-region: ${vars.AWS_REGION}
      - name: Run AWS S3 sync
        run: |
          aws s3 sync "$GITHUB_WORKSPACE"/munki_repo ${vars.MUNKI_REPO_URI} --exclude '*.DS_Store'
      - name: Poke Zentral server
        run: |
          curl -X POST -H "Authorization: Token $ZTL_API_TOKEN" https://$ZTL_FQDN/api/monolith
        env:
          ZTL_FQDN: ${vars.ZTL_FQDN}
          ZTL_API_TOKEN: ${secrets.ZTL_API_TOKEN}
  slack-notify:
    runs-on: ubuntu-latest
    needs: sync-repo
```



Munki repository

Are we happy with this setup?

- It does the job
- But it is highly inefficient
 - The AutoPkg cache is not used
 - Needs both Git LFS and Amazon S3



A macOS client built from code



The MDM configuration



The MDM configuration

The code - overview

- Official Zentral Terraform provider

```
1 terraform {
2   required_providers {
3     zentral = {
4       source = "zentralopensource/zentral"
5     }
6   }
7   cloud {
8     organization = "zentral-pro-services"
9     workspaces {
10      name = "macos-client-as-code"
11    }
12  }
13 }
14
15 // configure the provider
16 provider "zentral" {
17   // URL where the API endpoints are mounted in the Zentral deployment.
18   // The ZTL_API_BASE_URL environment variable can be used instead.
19   base_url = "https://zentral.example.com/api/"
20
21   // Zentral service account (better) or user API token.
22   // This is a secret, it must be managed using a variable.
23   // The ZTL_API_TOKEN environment variable can be used instead.
24   token = var.api_token
25 }
26
```

A macOS client built from code



The MDM configuration

The code - overview

- Official Zentral Terraform provider
- Resources described as Terraform HCL files

```
1 terraform {
2   required_providers {
3     zentral = {
4       source = "zentralopen-source/zentral"
5     }
6   }
7   cloud {
8     organization = "zentral-pro-services"
9     workspaces {
10      name = "macos-client-as-code"
11    }
12  }
13 }
14
15 // configure the provider
16 provider "zentral" {
17   // URL where the API endpoints are mounted in the Zentral deployment.
18   // The ZTL_API_BASE_URL environment variable can be used instead.
19   base_url = "https://zentral.example.com/api/"
20
21   // Zentral service account (better) or user API token.
22   // This is a secret, it must be managed using a variable.
23   // The ZTL_API_TOKEN environment variable can be used instead.
24   token = var.api_token
25 }
```

A macOS client built from code



The MDM configuration

The code - overview

- ▶ Official Zentral Terraform provider
- ▶ Resources described as Terraform HCL files
- ▶ Standard configuration profiles in separate sub folder

The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORER' sidebar displays a file tree. The 'MCAC-ZENTRAL-CONFIG' folder is expanded, showing a list of files including 'mobileconfigs', 'all.notificationsettings.v1.mobile...', 'all.servicemanagement.v1.mobile...', 'all.servicemanagement.v2.mobile...', 'hardening.firewall.v1.mobileconfig', 'hardening.loginwindow.v1.mobile...', 'hardening.passwordpolicy.mobile...', 'hardening.screensaver.sonomas.m...', 'hardening.screensaver.ventura.m...', 'monolith.default-enrollment.v1.m...', 'osquery.tcc.v1.mobileconfig', 'santa.default-configuration.v4.m...', 'santa.default-configuration.v5.m...', 'santa.notificationsettings.v1.mobi...', 'santa.system-extension-policy.v1...', and 'santa.tcc.v1.mobileconfig'. The 'provider.tf' file is selected and highlighted. The main editor area shows the contents of 'provider.tf', which is a Terraform configuration file. It defines a provider 'zentral' and a cloud configuration. The code is as follows:

```
1 terraform {
2   required_providers {
3     zentral = {
4       source = "zentralopensource/zentral"
5     }
6   }
7 }
8
9 cloud {
10   organization = "zentral-pro-services"
11   workspaces {
12     name = "macos-client-as-code"
13   }
14 }
15
16 // configure the provider
17 provider "zentral" {
18   // URL where the API endpoints are mounted in the Zentral deployment.
19   // The ZTL_API_BASE_URL environment variable can be used instead.
20   base_url = "https://zentral.example.com/api/"
21
22   // Zentral service account (better) or user API token.
23   // This is a secret, it must be managed using a variable.
24   // The ZTL_API_TOKEN environment variable can be used instead.
25   token = var.api_token
26 }
```

A macOS client built from code



The MDM configuration

The code - Santa / Zentral configuration

- One TF resource for Zentral Santa configuration

```
santa_configurations.tf x
1 resource "zentral_santa_configuration" "default" {
2   name           = "Default"
3   client_mode    = "MONITOR"
4   enable_bundles = true
5   enable_transitive_rules = true
6   sync_incident_severity = 200
7 }
8
9 resource "zentral_santa_enrollment" "default" {
10  configuration_id = zentral_santa_configuration.default.id
11  meta_business_unit_id = zentral_meta_business_unit.default.id
12 }
13
```

A macOS client built from code



The MDM configuration

The code - Santa / Zentral configuration

- One TF resource for Zentral Santa configuration
- One TF resource for the Santa enrollment

```
santa_configurations.tf X
1 resource "zentral_santa_configuration" "default" {
2   name           = "Default"
3   client_mode    = "MONITOR"
4   enable_bundles = true
5   enable_transitive_rules = true
6   sync_incident_severity = 200
7 }
8
9 resource "zentral_santa_enrollment" "default" {
10  configuration_id = zentral_santa_configuration.default.id
11  meta_business_unit_id = zentral_meta_business_unit.default.id
12 }
13
```

A macOS client built from code



The MDM configuration

The code - Santa / Zentral configuration

- One TF resource for Zentral Santa configuration
- One TF resource for the Santa enrollment
- One to one mapping with the Zentral server resources

```
santa_configurations.tf  
1 resource "zentral_santa_configuration" "default"  
2   name = "Default"  
3   client_mode = "Monitor"  
4   enable_bundles = true  
5   enable_transitive_rules = true  
6   sync_incident_severity = "Major"  
7 }  
8  
9 resource "zentral_santa_enrollment" "default"  
10  configuration_id = zentral_santa_configuration.default.id  
11  meta_business_unit_id = "Default"  
12 }  
13
```

The screenshot shows the Zentral web interface. The top navigation bar includes 'Inventory', 'Probes', 'Incidents', and 'Monolith'. The breadcrumb trail is 'Home / Santa / Configurations / Default'. The main heading is 'Santa configuration Default'. Below this is a table with 'Attribute' and 'Value' columns. The 'Mode' attribute is set to 'Monitor' (highlighted with a green button). Other attributes include 'Client certificate auth' (no), 'Batch size' (50), 'Full sync interval' (600s), 'Enable bundles' (yes), 'Enable transitive rules' (yes), 'Block USB mass storage' (no), 'Remount USB mode' (-), 'Allow Unknown shard' (100%), 'Enable all event upload shard' (0%), and 'Sync incident severity' (Major). At the bottom, there are buttons for 'Update', 'Events', and 'elasticsearch'. Below the table, the 'Created at' and 'Updated at' timestamps are shown as 'Fri, 08 Sep 2023 07:41:55 +0000'. The section '1 Enrollment' has a 'Create' button. Below this is a table with columns 'Business unit', 'Tags', 'Created at', and 'Request co'. The 'Default' business unit is listed with a tag of '-' and a creation time of 'Sept. 8, 2023, 7:42 a.m.'.

Attribute	Value
Name	Default
Mode	Monitor
Client certificate auth	no
Batch size	50
Full sync interval	600s
Enable bundles	yes
Enable transitive rules	yes
Block USB mass storage	no
Remount USB mode	-
Allow Unknown shard	100%
Enable all event upload shard	0%
Sync incident severity	Major

Created at: Fri, 08 Sep 2023 07:41:55 +0000
Updated at: Fri, 08 Sep 2023 07:41:55 +0000

Buttons: Update, Events, elasticsearch

1 Enrollment

Create

Business unit	Tags	Created at	Request co
Default	-	Sept. 8, 2023, 7:42 a.m.	

A macOS client built from code



The MDM configuration

The code - Santa / Agent configuration

- One TF resource for the MDM artifact

```
mdm_artifacts.tf x
200 resource "zentral_mdm_artifact" "santa-default-configuration" {
201   name      = "Santa - Default configuration"
202   type      = "Profile"
203   channel   = "Device"
204   platforms = ["macOS"]
205 }
206
207 resource "zentral_mdm_profile" "santa-default-configuration-4" {
208   artifact_id = zentral_mdm_artifact.santa-default-configuration.id
209   source = base64encode(
210     templatefile(
211       "${path.module}/mobileconfigs/santa.default-configuration.v4.mobileconfig",
212       { secret = zentral_santa_enrollment.default.secret }
213     )
214   )
215   macos = true
216   version = 4
217 }
218
```

A macOS client built from code



The MDM configuration

The code - Santa / Agent configuration

- One TF resource for the MDM artifact
- One TF resource for the .mobileconfig version

```
mdm_artifacts.tf x
200 resource "zentral_mdm_artifact" "santa-default-configuration" {
201   name      = "Santa - Default configuration"
202   type      = "Profile"
203   channel   = "Device"
204   platforms = ["macOS"]
205 }
206
207 resource "zentral_mdm_profile" "santa-default-configuration-4" {
208   artifact_id = zentral_mdm_artifact.santa-default-configuration.id
209   source = base64encode(
210     templatefile(
211       "${path.module}/mobileconfigs/santa.default-configuration.v4.mobileconfig",
212       { secret = zentral_santa_enrollment.default.secret }
213     )
214   )
215   macos    = true
216   version  = 4
217 }
218
```

A macOS client built from code



The MDM configuration

The code - Santa / Agent configuration

- One TF resource for the MDM artifact
- One TF resource for the .mobileconfig version
- Enrollment secret passed as template variable

A macOS client built from code

```
mdm_artifacts.tf
200 resource "zentral_mdm_artifact" "santa-default-configuration" {
201   name      = "Santa - Default configuration"
202   type      = "Profile"
203   channel   = "Device"
204   platforms = ["macOS"]
205 }
206
207 resource "zentral_mdm_profile" "santa-default-configuration-4" {
208   artifact_id = zentral_mdm_artifact.santa-default-configuration.id
209   source = base64encode(
210     templatefile(
211       "${path.module}/mobileconfigs/santa-default-configuration.v4.mobileconfig",
212       { secret = zentral_santa_enrollment.default.secret }
213     )
214   )
215   macos = true
216   version = 4
217 }

santa.default-configuration.v4.mobileconfig
21 <key>mcx_preference_settings</key>
22 <dict>
23   <key>ClientMode</key>
24   <integer>1</integer>
25   <key>MachineOwner</key>
26   <string>$REALM_USER_EMAIL</string>
27   <key>SyncBaseUrl</key>
28   <string>https://zentral.example.com/public/santa/sync/${secret}</string>
29 </dict>
```



The MDM configuration

The code - Bootstrap package

- One TF resource for the MDM artifact
- Installed during setup assistant with required .mobileconfig

```
mdm_artifacts.tf x
22 # Bootstrap package
23
24 resource "zentral_mdm_artifact" "bootstrap-pkg" {
25   name           = "Monolith - Bootstrap package"
26   type           = "Enterprise App"
27   channel        = "Device"
28   platforms      = ["macOS"]
29   auto_update    = false
30   install_during_setup_assistant = true
31   requires       = [zentral_mdm_artifact.monolith-default-enrollme
32 }
33
34 resource "zentral_mdm_enterprise_app" "bootstrap-pkg-2023-001" {
35   artifact_id    = zentral_mdm_artifact.bootstrap-pkg.id
36   package_uri    = "s3://zentral-artifacts-2011/bootstrap_pk
37   package_sha256 = "c577e35b373874fe8319d508be2455d3b9629301ca74cee1705481bdb4585c
38   macos         = true
39   version        = 1
40 }
41
```

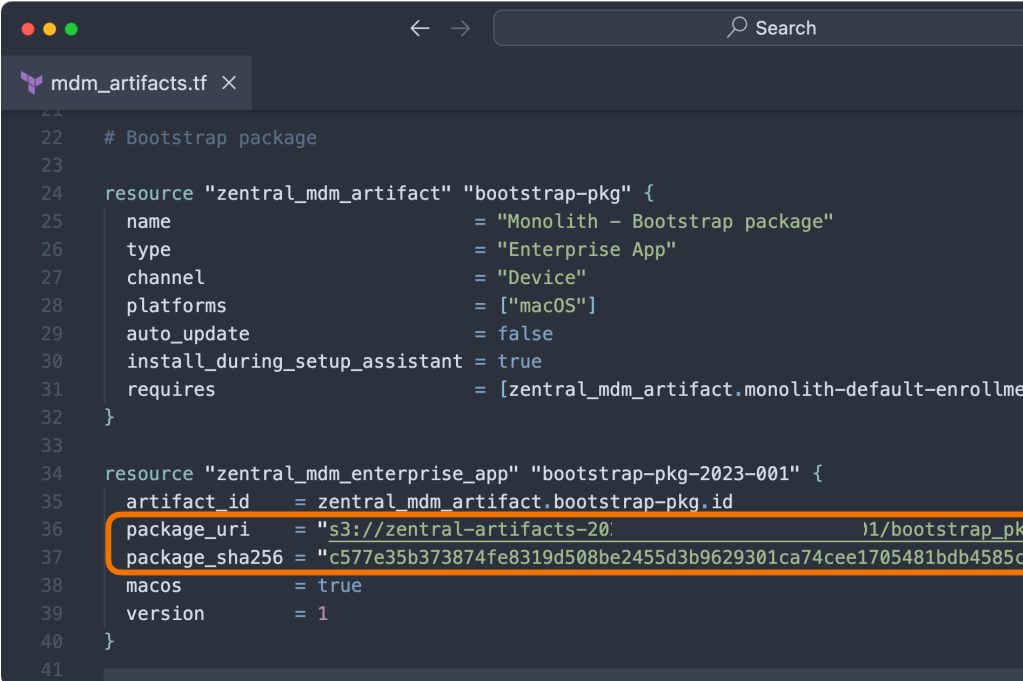
A macOS client built from code



The MDM configuration

The code - Bootstrap package

- One TF resource for the MDM artifact
 - Installed during setup assistant with required .mobileconfig
- One TF resource for the Enterprise Application version
 - Package from S3 with checksum



```
22 # Bootstrap package
23
24 resource "zentral_mdm_artifact" "bootstrap-pkg" {
25   name           = "Monolith - Bootstrap package"
26   type           = "Enterprise App"
27   channel        = "Device"
28   platforms      = ["macOS"]
29   auto_update    = false
30   install_during_setup_assistant = true
31   requires       = [zentral_mdm_artifact.monolith-default-enrollme
32 }
33
34 resource "zentral_mdm_enterprise_app" "bootstrap-pkg-2023-001" {
35   artifact_id    = zentral_mdm_artifact.bootstrap-pkg.id
36   package_uri    = "s3://zentral-artifacts-2011/bootstrap_pk
37   package_sha256 = "c577e35b373874fe8319d508be2455d3b9629301ca74cee1705481bdb4585c
38   macos         = true
39   version        = 1
40 }
41
```

A macOS client built from code



The MDM configuration

The code - MDM Blueprint

- One TF resource for the Blueprint

A macOS client built from code

```
mdm_blueprints.tf x
1 resource "zentral_mdm_blueprint" "default" {
2   name                = "Default"
3   collect_apps        = "ALL"
4   collect_certificates = "ALL"
5   collect_profiles    = "ALL"
6   filevault_config_id = zentral_mdm_filevault_config.default.id
7   recovery_password_config_id = zentral_mdm_recovery_password_config.default.id
8 }
9
10 # FileVault
11
12 resource "zentral_mdm_filevault_config" "default" {
13   name                = "Default"
14   escrow_location_display_name = "Zentral Pro Services GmbH"
15   at_login_only       = true
16   bypass_attempts     = 0
17   destroy_key_on_standby = true
18   show_recovery_key   = false
19 }
20
21 # Recovery password
22
23 resource "zentral_mdm_recovery_password_config" "default" {
24   name = "Default"
25 }
26
27 # Bootstrap package
28
29 resource "zentral_mdm_blueprint_artifact" "bootstrap-pkg" {
30   blueprint_id = zentral_mdm_blueprint.default.id
31   artifact_id  = zentral_mdm_artifact.bootstrap-pkg.id
32   macos        = true
33 }
34
```



The MDM configuration

The code - MDM Blueprint

- One TF resource for the Blueprint
 - With one FileVault configuration

A macOS client built from code

```
mdm_blueprints.tf x
1 resource "zentral_mdm_blueprint" "default" {
2   name                = "Default"
3   collect_apps        = "ALL"
4   collect_certificates = "ALL"
5   collect_profiles    = "ALL"
6   filevault_config_id = zentral_mdm_filevault_config.default.id
7   recovery_password_config_id = zentral_mdm_recovery_password_config.default.id
8 }
9
10 # FileVault
11
12 resource "zentral_mdm_filevault_config" "default" {
13   name                = "Default"
14   escrow_location_display_name = "Zentral Pro Services GmbH"
15   at_login_only       = true
16   bypass_attempts     = 0
17   destroy_key_on_standby = true
18   show_recovery_key   = false
19 }
20
21 # Recovery password
22
23 resource "zentral_mdm_recovery_password_config" "default" {
24   name = "Default"
25 }
26
27 # Bootstrap package
28
29 resource "zentral_mdm_blueprint_artifact" "bootstrap-pkg" {
30   blueprint_id = zentral_mdm_blueprint.default.id
31   artifact_id  = zentral_mdm_artifact.bootstrap-pkg.id
32   macos        = true
33 }
34
```



The MDM configuration

The code - MDM Blueprint

- One TF resource for the Blueprint
 - With one FileVault configuration
 - With one recovery password configuration

A macOS client built from code

```
mdm_blueprints.tf x
1 resource "zentral_mdm_blueprint" "default" {
2   name                = "Default"
3   collect_apps        = "ALL"
4   collect_certificates = "ALL"
5   collect_profiles    = "ALL"
6   filevault_config_id = zentral_mdm_filevault_config.default.id
7   recovery_password_config_id = zentral_mdm_recovery_password_config.default.id
8 }
9
10 # FileVault
11
12 resource "zentral_mdm_filevault_config" "default" {
13   name                = "Default"
14   escrow_location_display_name = "Zentral Pro Services GmbH"
15   at_login_only       = true
16   bypass_attempts     = 0
17   destroy_key_on_standby = true
18   show_recovery_key   = false
19 }
20
21 # Recovery password
22
23 resource "zentral_mdm_recovery_password_config" "default" {
24   name = "Default"
25 }
26
27 # Bootstrap package
28
29 resource "zentral_mdm_blueprint_artifact" "bootstrap-pkg" {
30   blueprint_id = zentral_mdm_blueprint.default.id
31   artifact_id  = zentral_mdm_artifact.bootstrap-pkg.id
32   macos        = true
33 }
34
```



The MDM configuration

The code - MDM Blueprint

- One TF resource for the Blueprint
 - With one FileVault configuration
 - With one recovery password configuration
- One TF resource to attach the Bootstrap package artifact

A macOS client built from code

```
mdm_blueprints.tf x
1 resource "zentral_mdm_blueprint" "default" {
2   name                = "Default"
3   collect_apps         = "ALL"
4   collect_certificates = "ALL"
5   collect_profiles    = "ALL"
6   filevault_config_id  = zentral_mdm_filevault_config.default.id
7   recovery_password_config_id = zentral_mdm_recovery_password_config.default.id
8 }
9
10 # FileVault
11
12 resource "zentral_mdm_filevault_config" "default" {
13   name                = "Default"
14   escrow_location_display_name = "Zentral Pro Services GmbH"
15   at_login_only       = true
16   bypass_attempts     = 0
17   destroy_key_on_standby = true
18   show_recovery_key   = false
19 }
20
21 # Recovery password
22
23 resource "zentral_mdm_recovery_password_config" "default" {
24   name = "Default"
25 }
26
27 # Bootstrap package
28
29 resource "zentral_mdm_blueprint_artifact" "bootstrap-pkg" {
30   blueprint_id = zentral_mdm_blueprint.default.id
31   artifact_id  = zentral_mdm_artifact.bootstrap-pkg.id
32   macos       = true
33 }
34
```




The MDM configuration

The code - Osquery compliance checks

- One TF resource for the Osquery query
- Compliance check for the screensaver hardening settings

A macOS client built from code

```
osquery_compliance_checks.tf x
39 resource "zentral_osquery_query" "loginwindow-cc" {
40   name      = "Loginwindow settings check"
41   description = "Check if the loginwindow settings are the expected ones"
42   sql = trimspace(<<--EOT
43     WITH expected_policies(domain, name, expected_value) AS (
44       VALUES
45         ('com.apple.screensaver', 'askForPassword', '1'),
46         ('com.apple.screensaver', 'askForPasswordDelay', '5'),
47         ('com.apple.screensaver', 'idleTime', '900')
48     ) SELECT expected_policies.*, managed_policies.value,
49     CASE
50       WHEN managed_policies.value = expected_policies.expected_value THEN 'OK'
51       ELSE 'FAILED'
52     END ztl_status
53   FROM expected_policies
54   LEFT JOIN managed_policies ON (
55     managed_policies.domain = expected_policies.domain
56     AND managed_policies.name = expected_policies.name
57   )
58   WHERE managed_policies.username = '' AND managed_policies.manual = 0
59   ORDER BY domain, name;
60   EOT
61 )
62   platforms      = ["darwin"]
63   compliance_check_enabled = true
64   scheduling = {
65     pack_id      = zentral_osquery_pack.compliance-checks.id,
66     interval     = var.osquery_default_compliance_check_interval,
67     log_removed_actions = false,
68     snapshot_mode = true
69   }
70 }
71
```



The MDM configuration

Pipeline - Overview

- ▶ No need for custom deployment scripts
- ▶ Terraform plan / Terraform apply
- ▶ Standard GitHub / Terraform Cloud integration

A macOS client built from code

The screenshot displays the Terraform Cloud web interface. On the left is a dark sidebar with navigation options: Workspaces, mcac-zentral-config, Overview, Runs (selected), States, Variables, and Settings. The main panel shows the 'mcac-zentral-config' workspace details, including its description, status (Unlocked), resources (70), Terraform version (v1.5.7), and update time. Below this is the 'Current Run' section, which shows a run titled 'add BTM notification settings (#3)' with a 'CURRENT' status. The 'Run List' section follows, featuring a filter bar with counts for All (56), Needs Attention (0), Errored (12), Running (0), and On Hold (0). A search bar and filter tabs for Status, Operation, and Source are also present. The run list contains three entries: 1) 'add BTM notification settings (#3)' (CURRENT), 2) 'add BTM notification settings' (plan-only run), and 3) 'Add Firewall settings to default blueprint' (plan-only run).

zentral-pro-services / Projects & workspaces / mcac-zentral-config / Runs

mcac-zentral-config

The Zentral configuration for the macOS standard Zentral Pro Services client.

Unlocked Resources 70 Terraform v1.5.7 Updated a few seconds ago

Current Run

add BTM notification settings (#3) CURRENT
#run-pn3mhMzJNdPqEv69 | headmin triggered via GitHub | Branch main | d5bb2

Run List

All 56 Needs Attention 0 Errored 12 Running 0 On Hold 0

Search Runs

add BTM notification settings (#3) CURRENT
#run-pn3mhMzJNdPqEv69 | headmin triggered via GitHub | Branch main | d5bb2

add BTM notification settings
#run-bm6hbh5VoUj4mkgU | plan-only run | headmin triggered via GitHub | Branch 230920_BT_notifications | 2645378

Add Firewall settings to default blueprint
#run-BxXBwSTQ3Pv1aBmq | np5 triggered via GitHub | Branch main | 12df15b

Add Firewall settings to default blueprint
#run-2g74V8Dkf7he5bKp | plan-only run | np5 triggered via GitHub | Branch 20230919_add_firewall_settings_to_blueprint | 17e8bc3



The MDM configuration

Pipeline example: Silence the background task notifications

- ▶ Add the Terraform resources:
 - ▶ MDM artifact, MDM profile, MDM Blueprint artifact

A macOS client built from code

The screenshot shows a GitHub commit titled "add BTM notification settings (#3)" by user "headmin" committed 1 minute ago. The commit shows 3 changed files with 68 additions and 0 deletions. The files are:

- mdm_artifacts.tf
- mdm_blueprints.tf
- mobileconfigs/all.notificationsettings.v1...

The code snippets shown are:

```
@@ -70,6 +70,20 @@ resource "zentral_mdm_profile" "servicemanagement-2" {
  70     version      = 2
  71 }
  72
  73 + resource "zentral_mdm_artifact" "all-notificationsettings" {
  74 +   name          = "All - Notification settings"
  75 +   type          = "Profile"
  76 +   channel       = "Device"
  77 +   platforms     = ["macOS"]
  78 + }
  79 +
  80 + resource "zentral_mdm_profile" "all-notificationsettings-1" {
  81 +   artifact_id   = zentral_mdm_artifact.all-notificationsettings.id
  82 +   source        = filebase64("${path.module}/mobileconfigs/all.notificationsettings.v1...")
  83 +   macos        = true
  84 +   version       = 1
  85 + }
  86 +

  73     # Hardening
  74
  75     resource "zentral_mdm_artifact" "hardening-loginwindow" {
```

```
@@ -40,6 +40,12 @@ resource "zentral_mdm_blueprint_artifact" "servicemanagement" {
  40     macos        = true
  41 }
  42
  43 + resource "zentral_mdm_blueprint_artifact" "all-notificationsettings" {
  44 +   blueprint_id  = zentral_mdm_blueprint.default.id
  45 +   artifact_id   = zentral_mdm_artifact.all-notificationsettings.id
  46 +   macos        = true
  47 + }
  48 +

  43     # Hardening
  44
  45     resource "zentral_mdm_blueprint_artifact" "hardening-loginwindow" {
```

```
... @@ -0,0 +1,48 @@
  1 + <?xml version="1.0" encoding="UTF-8"?>
  2 + <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
  3 + <plist version="1.0">
```

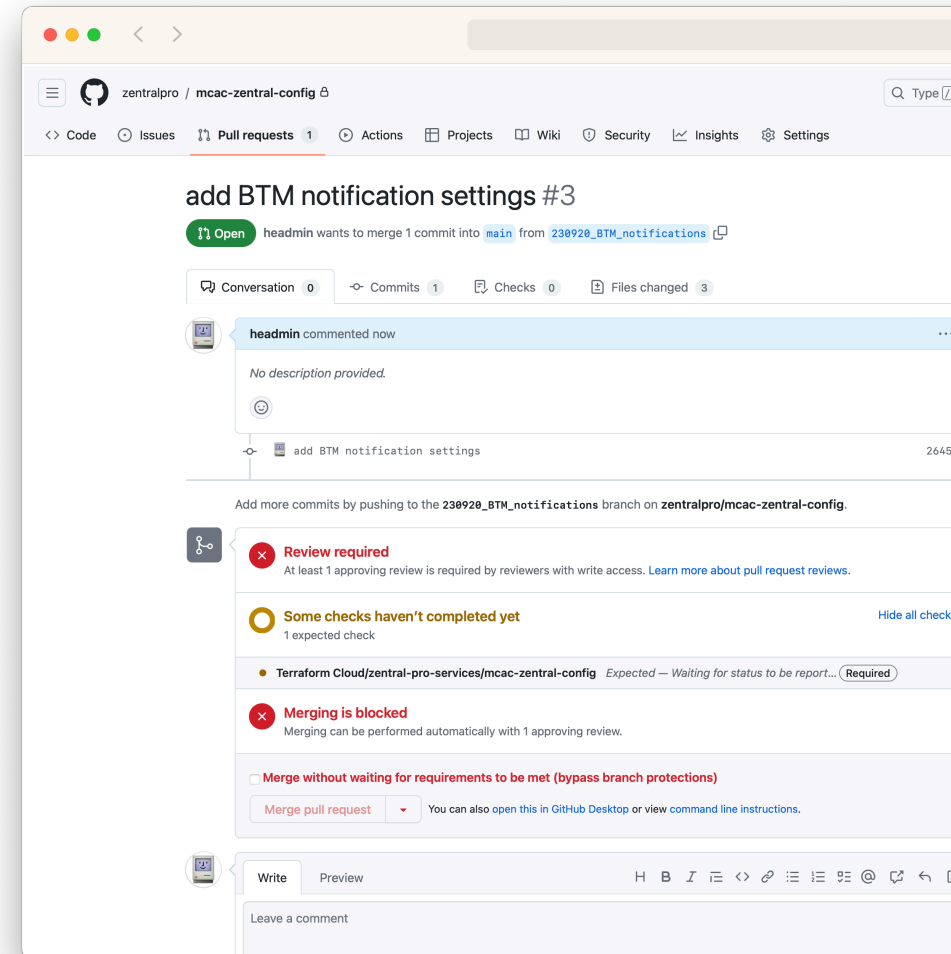


The MDM configuration

Pipeline example: Silence the background task notifications

- ▶ Add the Terraform resources:
 - ▶ MDM artifact, MDM profile, MDM Blueprint artifact
- ▶ Commit and make a PR
- ▶ **Terraform plan** run triggered in Terraform Cloud

A macOS client built from code



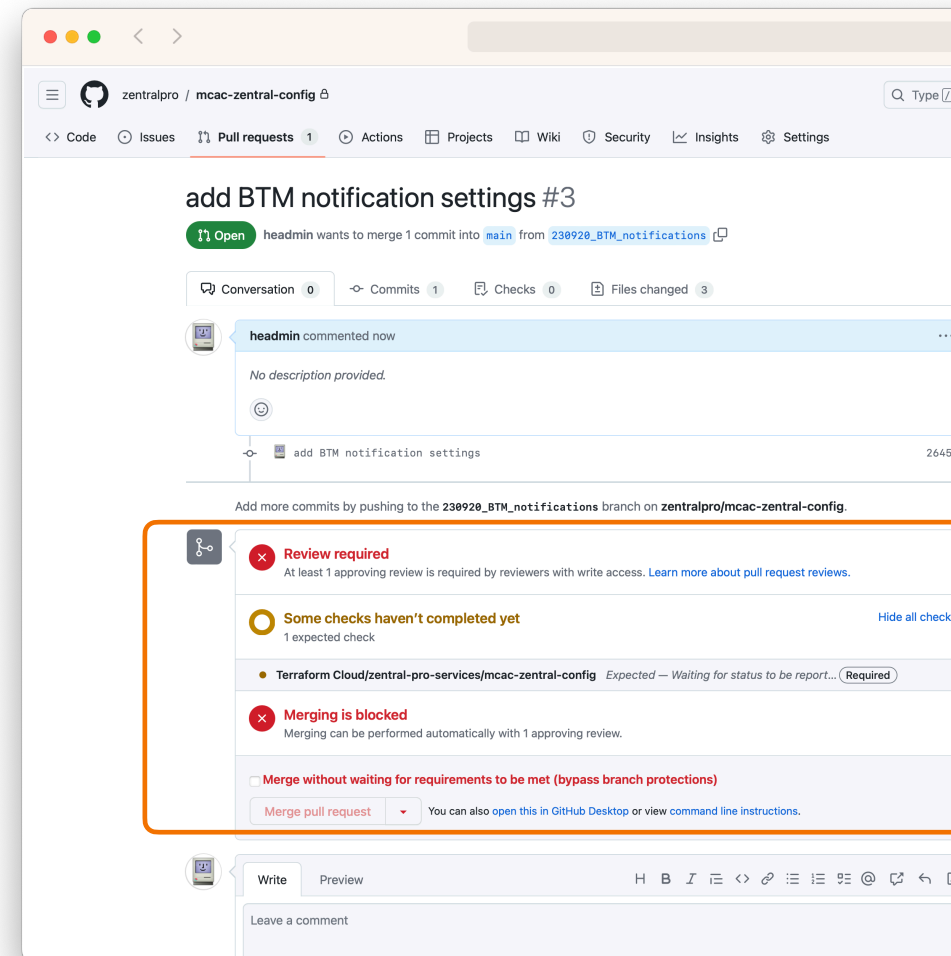


The MDM configuration

Pipeline example: Silence the background task notifications

- ▶ Add the Terraform resources:
 - ▶ MDM artifact, MDM profile, MDM Blueprint artifact
- ▶ Commit and make a PR
- ▶ **Terraform plan** run triggered in Terraform Cloud
- ▶ Mandatory code review

A macOS client built from code



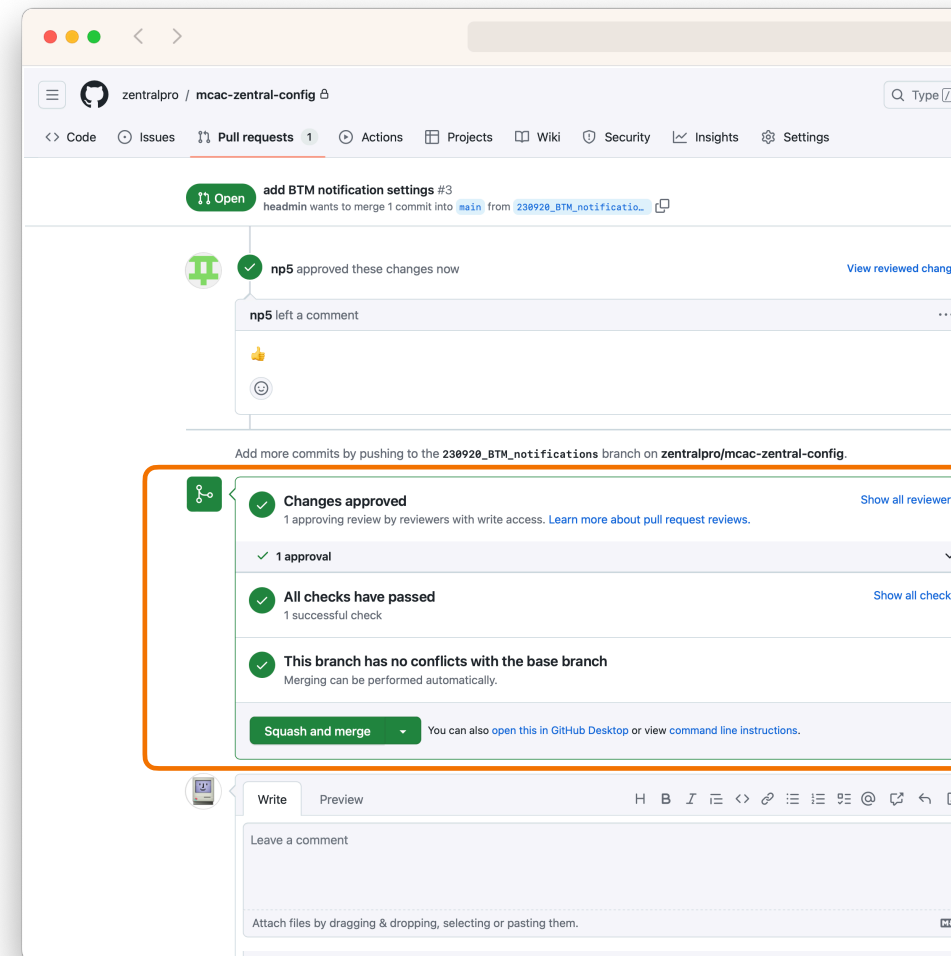


The MDM configuration

Pipeline example: Silence the background task notifications

- ▶ Add the Terraform resources:
 - ▶ MDM artifact, MDM profile, MDM Blueprint artifact
- ▶ Commit and make a PR
- ▶ **Terraform plan** run triggered in Terraform Cloud
- ▶ Mandatory code review
- ▶ Merge the PR

A macOS client built from code





The MDM configuration

Pipeline example: Silence the background task notifications

- ▶ Add the Terraform resources:
 - ▶ MDM artifact, MDM profile, MDM Blueprint artifact
- ▶ Commit and make a PR
- ▶ **Terraform plan** run triggered in Terraform Cloud
- ▶ Mandatory code review
- ▶ Merge the PR
- ▶ **Terraform apply** run triggered in Terraform Cloud

A macOS client built from code

zentral-pro-services / Projects & workspaces / mcac-zentral-config / Runs / run-pn3mhMzJNdPqEv69

mcac-zentral-config

ID: ws-27oQj2wRTh1XDPA [🔗](#)

The Zentral configuration for the macOS standard Zentral Pro Services client.

🔓 Unlocked 📦 Resources 70 🐛 Terraform v1.5.7 ⌚ Updated a few seconds ago

add BTM notification settings (#3)

Estimated cost change: None | Plan & apply duration: Less than a minute | Resources changed: +3 -0 -0

👤 headmin triggered a run from GitHub a minute ago

✅ Plan finished a minute ago

✅ Cost estimation finished a minute ago

✅ Apply finished a minute ago

Started a minute ago → Finished a minute ago

+ 3 created

Filter resources by address... Filter by action

> + 🐛	zentral_mdm_artifact.all-notificationsettings	✓ Created	id=c4677aaa-10f1-4a24-a...
> + 🐛	zentral_mdm_blueprint_artifact.all-notificationsettings	✓ Created	
> + 🐛	zentral_mdm_profile.all-notificationsettings-1	✓ Created	id=4adb67f1-64ba-4be7-93...

State versions created:



The MDM configuration

Pipeline example: Silence the background task notifications

- ▶ Add the Terraform resources:
 - ▶ MDM artifact, MDM profile, MDM Blueprint artifact
- ▶ Commit and make a PR
- ▶ **Terraform plan** run triggered in Terraform Cloud
- ▶ Mandatory code review
- ▶ Merge the PR
- ▶ **Terraform apply** run triggered in Terraform Cloud
- ▶ Changes applied in the GUI

The screenshot displays the Zentra web interface for configuring MDM settings. The breadcrumb trail is: Home / MDM / Artifacts / All - Notification settings. The page title is "All - Notification settings" with an "Update" button. The configuration is organized into several sections:

- Channel:** Device
- Platform:** macOS
- Depends on:** -
- Install during setup assistant:** no
- Auto update:** yes
- Reinstall interval:** Never
- Reinstall on OS update:** No
- Profile:**
 - Payload identifier:** pro.zentral.all.notificationsettings
 - Payload description:** Disables Background Task Management notifications
 - Content:**

Type	Device
com.apple.notificationsettings	No
- Installed payload identifier:** pro.zentral.dmd.artifact.c4677aaa-10f1-4a2
- Version:**

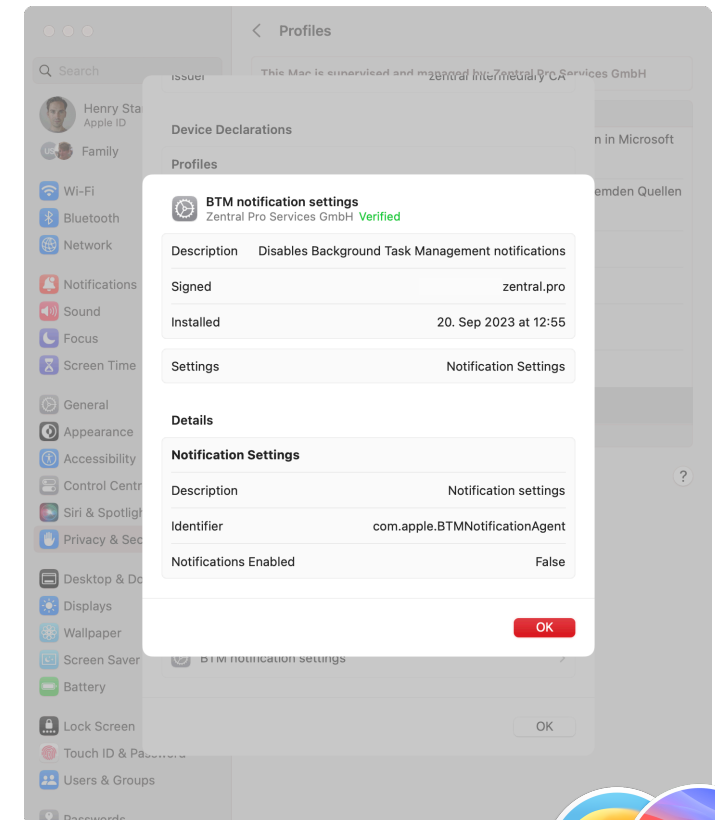
Version	Platforms	Excl. tags	Def. shard	Tag shards
1	macOS	-	100/100	-
- Blueprint:**
 - Add** button
 - | Name | Platforms | Excl. tags | Def. shard | Tag shards |
|---------|-----------|------------|------------|------------|
| Default | macOS | - | 100/100 | - |



The MDM configuration

Pipeline example: Silence the background task notifications

- ▶ Add the Terraform resources:
 - ▶ MDM artifact, MDM profile, MDM Blueprint artifact
- ▶ Commit and make a PR
- ▶ **Terraform plan** run triggered in Terraform Cloud
- ▶ Mandatory code review
- ▶ Merge the PR
- ▶ **Terraform apply** run triggered in Terraform Cloud
- ▶ Changes applied in the GUI





Result


A macOS client built from code



The Journey

- Zentral MDM MVP
- New era for Apple device mangement
- GitOps & DDM “buzz”

MENU ▼



In October 3-6 2023 we proudly present

MacSysAdmin Conference

**Come hell or high water.
We'll do it live!**

01 : 23 : 52 : 1

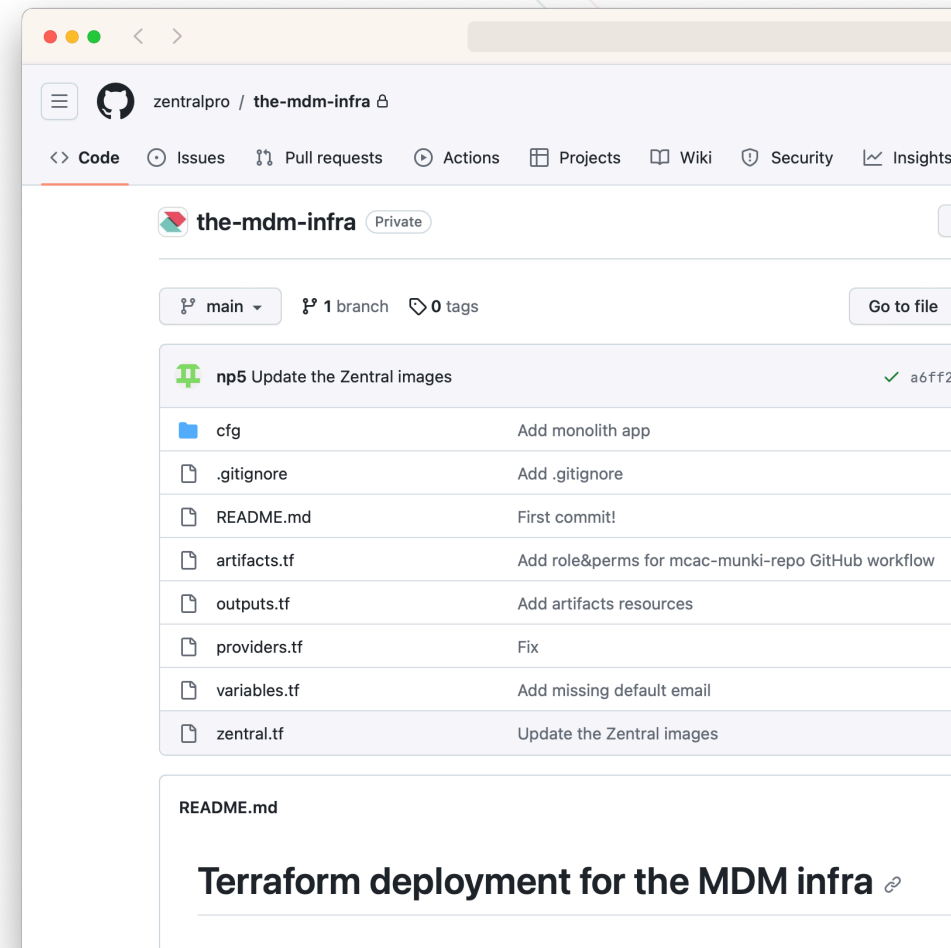
Days Hours Minutes Seconds

Kommentare sind deaktiviert. [Weitere Informationen](#)



The Journey

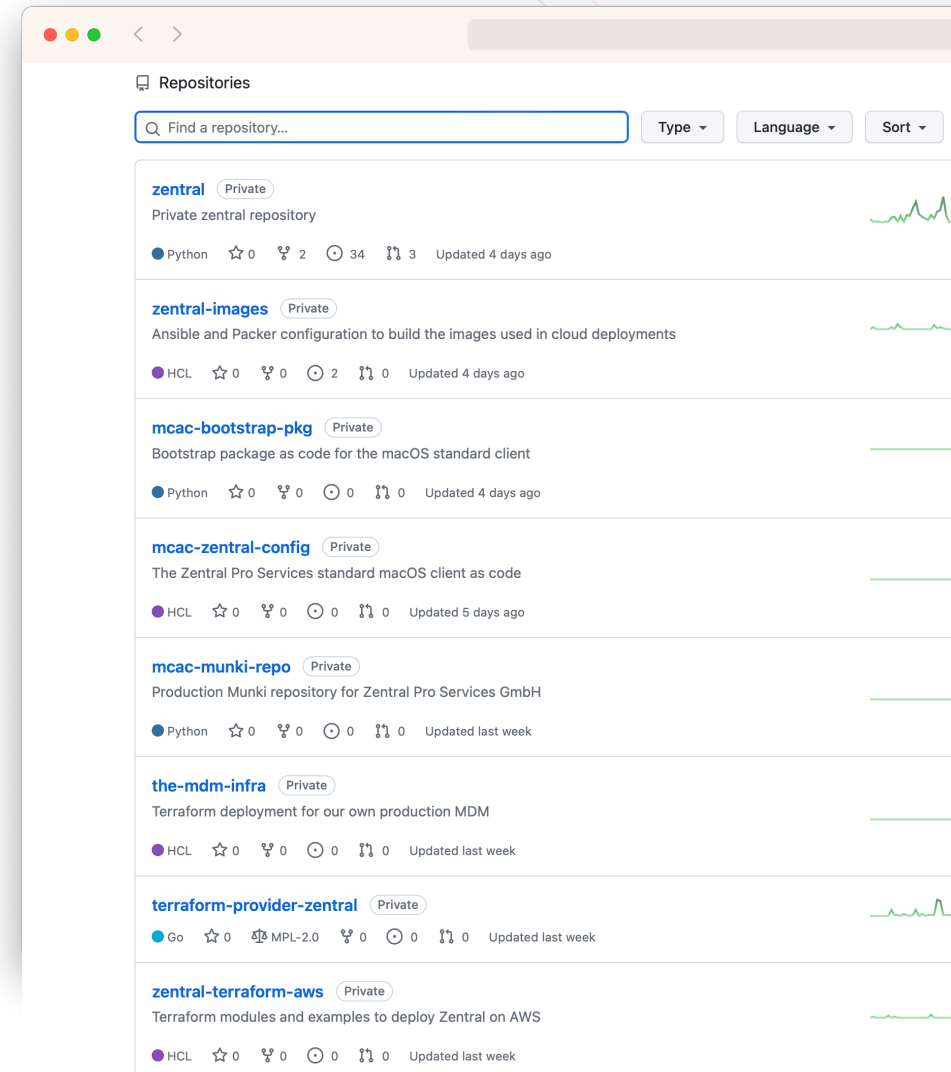
- ▶ Unforeseen challenges
- ▶ We did it!
- ▶ But it was a nail-biter





The Journey

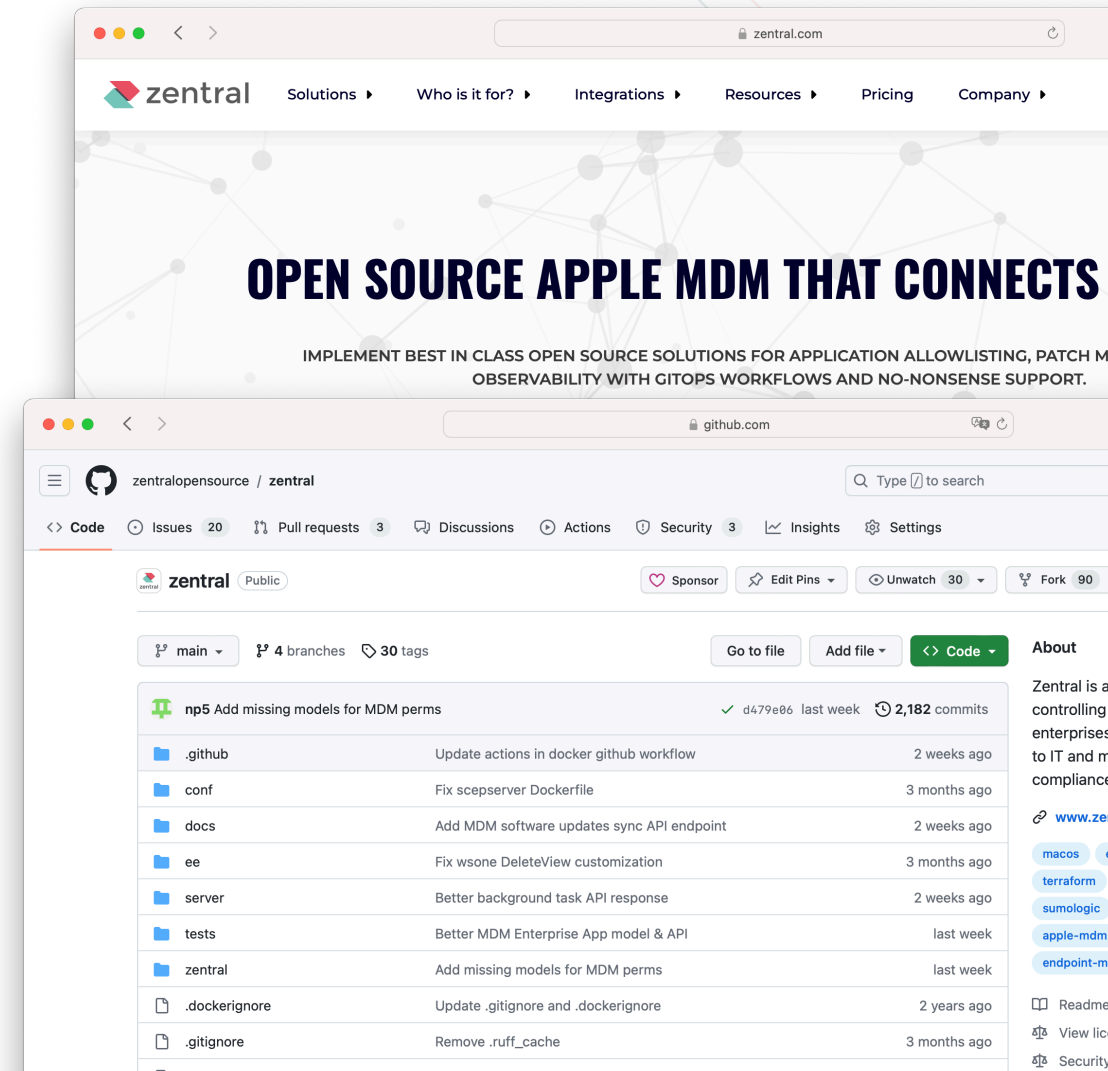
- ▶ Everything connects
- ▶ Zentral has evolved
- ▶ What do you think ?





Preview

- ▶ Santa voting system
- ▶ SCIM server
- ▶ Leverage DDM as much as possible





Thank you!

www.zentral.com