

Getting started with Contains Portal

Build a Mac-hosted content studio, pair a Portal, and publish a complete native dashboard update over home Wi-Fi. This guide starts from the public source repositories. The companion [design story](#) shows the ImageGen alternatives, the selected directions, and the changes made after testing on hardware.

This is a source-first personal project, tested on a first-generation Portal+ running Android 9. There is no prebuilt Contains Portal APK in these repositories. The Android fork's inherited provisioning kit and release manifest still refer to upstream releases; downloading those does not install this custom renderer.

1. Understand the two parts

Part	Runs where	Responsibility
contains-portal	Mac	Editor, approved content library, signing, publication, scheduling, device acknowledgments, backups
immortal-glass, contains-portal branch	Portal	Native family dashboard and Discover, verified downloads, cached generations, local date selection and alarms
Weather, transit, Sonos, Muse	Separate integrations	The Portal fetches weather/transit and controls Sonos directly; Muse is a separate application with its own setup

flowchart LR

```
B["Browser on the Mac"] -->|"Authenticated loopback :9444"| M["Mac studio and publisher"]
M --> D["Private SQLite, originals, keys and backups"]
P["Native Portal renderer"] -->|"Pull signed releases over pinned HTTPS :9443"| M
P --> C["Complete local content cache"]
P --> A["Local clock and alarm"]
P --> N["Weather, transit and LAN Sonos"]
```

The Mac sends data and assets to a fixed vocabulary of native widgets. It does not remotely execute arbitrary UI code. Content, supported theme colors, tile order, images, dates and supported settings can change without rebuilding Android. A new widget type, different layout algorithm, or new native behavior needs an APK change.

2. Have these ready

- A Mac on the same private network as the Portal, with Git and Python 3.9 or newer. Python 3.12 is also exercised by the publisher's GitHub workflow.

- A powered Portal, a USB data cable, permission to enable ADB, and its current data backed up. This guide assumes ADB can be enabled on its existing Android installation. For an unprepared stock device, first read the Android fork's [inherited provisioning instructions](#); their firmware and installer steps are separate from this content publisher.
- For Android builds: JDK 21, the Android command-line tools or Android Studio, Platform Tools, SDK Platform 36.1 and Build Tools 36.0.0. The checked-in Gradle wrapper supplies Gradle.
- For a release installation: the existing app's private signing key, or a genuinely new installation using a key you create and keep. Android requires signed APKs, and an update normally needs the existing signing identity. [Android signing documentation](#).
- For optional checks: Node.js; the publisher workflow uses Node 22.

No Google Play Services or cloud push service is required for content OTA. Image generation and Muse account/services are separate from local publishing.

Existing installation or new installation?

Situation	Path
This Contains Portal setup already runs	Start with section 9; do not initialize another data root or signing identity
The release package exists and you have its signing key	Back up first, then build and update with the same key
The release package exists but you do not have its signing key	Use the debug package alongside it to experiment; do not uninstall or clear the existing app to force an update
No app with the release package is installed	Create a private signing key for the release build, or try the debug package first

The release package is `dev.wobsoriano.immortalglass`. The debug package is `dev.wobsoriano.immortalglass.debug`. The Kotlin namespace and activity class still begin with `com.immortal.launcher`; that is not the installed package ID.

3. Clone both repositories

```
mkdir -p "$HOME/projects"
cd "$HOME/projects"
git clone https://github.com/mgalpert/contains-portal.git
git clone --branch contains-portal https://github.com/mgalpert/immortal-glass.git

export PORTAL_PROJECT="$HOME/projects/contains-portal"
export PORTAL_ANDROID="$HOME/projects/immortal-glass"
```

The publisher records the tested Android companion commit in [android-version.json](#). For this documented baseline, check out:

```
cd "$PORTAL_ANDROID"
git checkout 6d2539c2e1eb7ff999445968279e3b33c38109b2
```

This selects the tested snapshot and leaves a detached HEAD. Create your own branch before changing Android code. Later versions should use the commit recorded by the publisher version you choose.

For the tested manual-only Muse change described in the post, use [the tap-only revision](#) instead of that original baseline:

```
git checkout bbba965a58183d4c9014478899d2413bd3217279
```

This revision was built and checked on the Portal. It is on the codex/muse-tap-only branch while the change is reviewed; it is not a prebuilt APK release.

4. Prepare Python and the Mac's local identity

Keep the virtual environment outside the repository:

```
python3 -m venv "$HOME/.venvs/contains-portal"
source "$HOME/.venvs/contains-portal/bin/activate"
cd "$PORTAL_PROJECT"
python -m pip install -r requirements.txt
```

Identify the actual Wi-Fi interface and the Mac's local hostname:

```
networksetup -listallhardwareports
scutil --get LocalHostName
```

Find the Device line under Hardware Port: Wi-Fi. It may be en0 or en1. A second Ethernet interface can have a different address; use the private Wi-Fi interface.

Create the ignored local configuration file:

```
mkdir -p private
chmod 700 private
touch private/runtime-env.sh
chmod 600 private/runtime-env.sh
```

Edit that file with values from your Mac:

```
export PORTAL_PYTHON="/absolute/path/to/.venvs/contains-portal/bin/python"
export PORTAL_WIFI_INTERFACE="en0"
export PORTAL_SERVER_NAME="Your-Mac-LocalHostName.local"
```

Use the full hostname ending in .local, for example the LocalHostName output followed by .local. Configure this before the first initialization: it becomes a DNS identity in the device TLS certificate.

Both launchers load this private file. It is excluded from Git.

Using the Mac's actual .local hostname for the pairing endpoint avoids relying on the source's example IP address, 10.0.0.2. The certificate includes that example IP as well; changing a pairing URL does not add another IP to a certificate. Bonjour discovery can recover a changed address using the enrolled DNS identity and unchanged certificate pin.

Existing installations keep their existing certificate. Changing an environment variable does not replace it or silently re-enroll devices.

5. Initialize and publish the starter library

From the publisher checkout:

```
cd "$PORTAL_PROJECT"
./portalctl init
./portalctl seed --publish
./portalctl health
```

Initialization creates the private database, signing key, TLS key/certificate and studio access key. The seed publishes 30 consecutive approved Pacific dates from the existing reusable learning bank. Personal activity schedules start empty; the original family photos and deployment data are not in GitHub.

Review the starter content before using it with your family. New content you write in the editor needs its own approval.

Start the server in this terminal:

```
./portalctl serve --host auto
```

Keep that terminal running the server. Use a second setup terminal for the remaining steps. Environment exports do not carry between terminal sessions, so set the paths again, then open [the studio](#):

```
export PORTAL_PROJECT="$HOME/projects/contains-portal"
export PORTAL_ANDROIDID="$HOME/projects/immortal-glass"
cd "$PORTAL_PROJECT"
./portalctl access
```

Paste the key into the studio's access field. The browser editor is available only on this Mac. Allow the publisher through the Mac firewall for private-LAN device connections if prompted; no router port forwarding is needed. Guest-network client isolation can block discovery even when both devices have Wi-Fi. The Portal's download service listens on private Wi-Fi port 9443 and advertises a certificate-specific Bonjour service.

Expected host health is status ok, a positive revision, and at least 30 approved days. Before pairing, the device list can be empty. A fresh installation need not have the same revision number as the

original hardware demonstration.

6. Prepare and build the Android companion

Tools

Install JDK 21 and the SDK components through Android Studio's SDK Manager. Set your actual SDK and JDK locations:

```
export JAVA_HOME=$(/usr/libexec/java_home -v 21)
export ANDROID_HOME="$HOME/Library/Android/sdk"
export PATH="$JAVA_HOME/bin:$ANDROID_HOME/platform-tools:$ANDROID_HOME/cmdline-tools/latest/bin:$PATH"
java -version
adb version
```

Add ANDROID_HOME and JAVA_HOME to private/runtime-env.sh as well if you will use the publisher's APK registration tools later.

If using an existing command-line-tools installation, its sdkmanager can install the packages and present the license prompts:

```
sdkmanager "platform-tools" "platforms;android-36.1" "build-tools;36.0.0"
sdkmanager --licenses
```

SDK tool installation interfaces can change; Android Studio's SDK Manager is also supported. [Official SDK tool documentation](#).

Choose a build

For a side-by-side experiment that does not replace the release app:

```
cd "$PORTAL_ANDROID"
./gradlew :app:testDebugUnitTest :app:assembleDebug --no-daemon
export PORTAL_PACKAGE="dev.wobsoriano.immortalglass.debug"
export PORTAL_APK="$PORTAL_ANDROID/app/build/outputs/apk/debug/app-debug.apk"
```

For a release on your own installation, keep a stable signing key outside Git. Existing installations must use their existing key. Only for a new signing identity:

```
mkdir -p "$HOME/.immortal-signing"
chmod 700 "$HOME/.immortal-signing"
keytool -genkeypair -storetype PKCS12 \
  -keystore "$HOME/.immortal-signing/portal-release.p12" \
  -alias contains-portal -keyalg RSA -keysize 3072 -validity 10000
```

The command prompts for private signing details. Create `~/immortal-signing/keystore.properties` locally, using the passwords you chose:

```
storeFile=portal-release.p12
storePassword=YOUR_PRIVATE_STORE_PASSWORD
keyAlias=contains-portal
keyPassword=YOUR_PRIVATE_KEY_PASSWORD
```

```
chmod 600 "$HOME/.immortal-signing/keystore.properties"
cd "$PORTAL_ANDROID"
./gradlew :app:testDebugUnitTest :app:assembleRelease --no-daemon
export PORTAL_PACKAGE="dev.wobsoriano.immortalglass"
export PORTAL_APK="$PORTAL_ANDROID/app/build/outputs/apk/release/app-release.apk"
```

Keep the APK key and its passwords in a private backup. They are separate from the publisher's content-signing key. Release builds deliberately fail if signing configuration is absent. The optional release lint audit currently retains nine pre-existing errors; a successful release build is not a claim that lint is clean.

Connect and back up before installing

On the Portal, enable Settings → Debug → ADB Enabled, connect the USB data cable, and accept the authorization prompt. [Official ADB documentation](#).

```
adb devices -l
export PORTAL_SERIAL="REPLACE_WITH_YOUR_DEVICE_SERIAL"
adb -s "$PORTAL_SERIAL" shell pm path "$PORTAL_PACKAGE"
```

If that package already exists, create a private before-change backup. In the command below, replace the remote path with the base.apk path printed by pm path, omitting its package: prefix. If the output lists split APKs, pull each split too.

```
export PORTAL_DEVICE_BACKUP="$PORTAL_PROJECT/private/device-before-$(date +%Y%m%d-%H%M%S)"
mkdir -p "$PORTAL_DEVICE_BACKUP"
chmod 700 "$PORTAL_DEVICE_BACKUP"
adb -s "$PORTAL_SERIAL" pull "/REPLACE_WITH_INSTALLED_BASE_APK_PATH/base.apk" \
"$PORTAL_DEVICE_BACKUP/installed-base.apk"
adb -s "$PORTAL_SERIAL" pull "/sdcard/Android/data/$PORTAL_PACKAGE/files" \
"$PORTAL_DEVICE_BACKUP/external-files"
"$ANDROID_HOME/build-tools/36.0.0/apksigner" verify --print-certs \
"$PORTAL_DEVICE_BACKUP/installed-base.apk"
"$ANDROID_HOME/build-tools/36.0.0/apksigner" verify --print-certs "$PORTAL_APK"
```

Compare the signer certificate SHA-256 digest in the last two outputs. For an in-place update in this project, they must match. If there is no existing package, skip these installed-package backup and comparison commands. A missing external directory means there are no files there to copy; it does not establish that all app data has been backed up.

Existing Fleet users can also back up readable app preferences using their trusted management tools. Protected data in other applications may not be readable; an accessible-files backup is not a full device image. A signer mismatch or version downgrade error is a reason to stop this installation path, not to uninstall or add a downgrade flag.

For the chosen compatible build:

```
adb -s "$PORTAL_SERIAL" install -r "$PORTAL_APK"
```

7. Pair the Portal to the Mac

Keep the publisher server running. In the Mac's publisher checkout, create a new device bootstrap. Use the same .local name that was configured before initialization:

```
cd "$PORTAL_PROJECT"
./portalctl pair --name "Family Portal" \
  --endpoint "https://Your-Mac-LocalHostName.local:9443" \
  --output private/portal-bootstrap.json
```

This file contains a device token, certificate pin, signing public keys and discovery information. It does not contain the studio admin key. The output path must not already exist. For an existing device's credential rotation, reuse its device ID with `--device-id`; do not casually create a second identity.

Transfer it to the package you actually installed:

```
adb -s "$PORTAL_SERIAL" shell mkdir -p "/sdcard/Android/data/$PORTAL_PACKAGE/files"
adb -s "$PORTAL_SERIAL" push "$PORTAL_PROJECT/private/portal-bootstrap.json" \
  "/sdcard/Android/data/$PORTAL_PACKAGE/files/portal-bootstrap.json"
adb -s "$PORTAL_SERIAL" shell am start -n "$PORTAL_PACKAGE/com.immortal.launcher.HomeActivity"
```

The app validates enrollment, stores it privately, removes the external bootstrap, and pauses its upstream self-updater to protect the custom renderer. Pairing should not require disabling certificate verification or a browser certificate-warning exception.

Glass is the default theme for a new app profile. Swipe horizontally to Your day and then Discover. If an older profile uses the classic home, enable the Glass theme in Immortal settings. The dots indicate the face; horizontal swipes reliably move between faces.

Optional: if you want Immortal to open when Home is pressed, explicitly select it as the device's Home app. On supported firmware the equivalent ADB command is:

```
adb -s "$PORTAL_SERIAL" shell cmd package set-home-activity
"$PORTAL_PACKAGE/com.immortal.launcher.HomeActivity"
```

That changes the default Home app. It does not enroll device-owner kiosk mode or pin a parent-message screen.

Muse requires its own installed companion application, SDK/account setup and phone pairing. The Android widget opens that app; publishing a widget does not provision a Muse account or fix speech recognition. The [tap-only launcher revision](#) opens Muse with its Hey Muse setting off, rather than sending a wake-word broadcast. Tap the character inside Muse to begin speaking. If you use the older pinned baseline, open Muse → Settings → Hey Muse and turn it off. Open Muse from the app grid and tap its character; that baseline's dashboard shortcut still uses the old wake receiver and does not start a conversation with the listener off. The post describes why this household stopped using the wake word.

8. Prove the first sync is over Wi-Fi

Allow about a minute for the initial check, or resume the home app for catch-up. An open lesson or a ringing alarm can defer adoption.

```
cd "$PORTAL_PROJECT"
./portalctl health
adb -s "$PORTAL_SERIAL" reverse --list
adb -s "$PORTAL_SERIAL" shell ls "/sdcard/Android/data/$PORTAL_PACKAGE/files/portal-bootstrap.json"
```

Success means:

- Host device health acknowledges the current revision as applied.
- The family dashboard shows the content revision.
- The external bootstrap is absent after consumption.
- Your test is not using an ADB reverse route for the publisher connection.

If old publisher reverse routes exist, remove only those routes. Do not remove unrelated routes used by another project. Disconnect USB and confirm a subsequent published revision is acknowledged over Wi-Fi. Content OTA does not require Fleet to be enabled. After confirming successful enrollment, remove the host staging copy of private/portal-bootstrap.json; the durable device enrollment is already in the private data root.

For optional detailed device diagnostics, enable the authenticated Fleet agent through its settings and use its existing trusted token for GET /portal/health or POST /portal/sync. Fleet is a separate LAN management surface; its token is not the Mac studio key.

9. Publish everyday changes

1. Open the studio and choose the intended draft or published content.
2. In Daily learning, select a date and edit its complete fact, linked English/Spanish lesson, puzzle, science piece, kindness prompt and family conversation. Changing learning clears that day's approval.
3. Approve the edited date. Publication requires 30 consecutive approved America/Los_Angeles dates from activation; retain a complete future window.

4. In Quotes & activities, keep quote IDs stable and add activities to specific dates. Dates do not imply weekly recurrence; the native layout supports four activities per date.
5. In Photos & artwork, import an original JPEG, PNG or WebP with provenance. Choose its role before importing or including it. Portrait photo assets are selected for the family photo tile; landscape originals can remain in the private library.
6. Use Layout & theme for supported colors, tile order and provider cadences. The native layouts are family-04 and pocket-playground, not an arbitrary drag-and-drop canvas.
7. Review landscape and portrait content previews. Actual typography and long text still need a device check.
8. Save draft, then Review & publish. Choose immediate publication or a Pacific date/time. The editor rejects DST gaps and ambiguous repeated-hour scheduling.
9. Watch the device's applied revision. A staged/downloaded revision is not yet the visible revision. Close an open lesson or stop a ringing alarm normally before expecting adoption.

Images must decode successfully, fit within 8192 pixels per dimension and 16 million pixels total, and be at most 32 MiB each. Whole release limits also apply. The bundled generated artwork already works without importing replacement art.

Discover is available any time you swipe to it. Its daily pieces change by local date; there is no daily Next workflow. Puzzle and science answers can be revealed, while deeper lessons remain standalone pieces. Ordinary content releases preserve locally owned alarm and interaction state.

Leave Include Portal sync settings unchecked for ordinary content edits. If you intentionally manage those settings, use the acknowledged local settings version. A mismatch reports `settings_conflict` and retains the device's newer value.

The weather cities and J train route/stop are currently source defaults in `DailyDashboardData.kt`, not studio fields. Changing those provider locations or adding a new native widget requires an Android code change and another verified build.

10. Run the Mac service after login

Stop the foreground server with Control-C before starting another server on the same ports. Confirm that `private/runtime-env.sh` points to the Python environment that has the dependencies.

Generate a local `LaunchAgent` from the checked-in template:

```
cd "$PORTAL_PROJECT"
python3 - <<'PY'
from pathlib import Path
import plistlib
root = Path.cwd().resolve()
with (root / "support/dev.local.contains-portal.plist").open("rb") as source:
    job = plistlib.load(source)
job["ProgramArguments"] = [str(root / "support/run-service")]
job["WorkingDirectory"] = str(root)
job["StandardOutPath"] = str(root / "private/service.log")
```

```

job["StandardErrorPath"] = str(root / "private/service-error.log")
destination = Path.home() / "Library/LaunchAgents/dev.local.contains-portal.plist"
destination.parent.mkdir(parents=True, exist_ok=True)
with destination.open("wb") as output:
    plistlib.dump(job, output)
print(destination)
PY
launchctl bootstrap "gui/$(id -u)" "$HOME/Library/LaunchAgents/dev.local.contains-portal.plist"
launchctl print "gui/$(id -u)/dev.local.contains-portal"

```

For an already-loaded job, use `launchctl kickstart -k` instead of bootstrapping a duplicate. If you changed the installed plist, unload that job and bootstrap the updated file deliberately.

The wrapper uses service-scoped `caffeinate` to prevent system sleep on AC while permitting display sleep. The service starts after login and restarts after a process failure. It does not promise pre-login/FileVault access or unattended recovery after physical power loss. Cached Portal content and local alarms continue when the Mac is unavailable.

11. Back up and recover

Automatic private data snapshots are created daily. Keep another copy on a separate private disk. Publisher data backup does not include every file elsewhere on the Mac.

For a complete local configuration set, create a new backup folder and keep configuration beside the data snapshot:

```

cd "$PORTAL_PROJECT"
export PORTAL_BACKUP="$HOME/PortalBackups/portal-$(date +%Y%m%d-%H%M%S)"
mkdir -p "$PORTAL_BACKUP"
chmod 700 "$PORTAL_BACKUP"
./portalctl backup "$PORTAL_BACKUP/data"
cp private/runtime-env.sh "$PORTAL_BACKUP/runtime-env.sh"
cp "$HOME/Library/LaunchAgents/dev.local.contains-portal.plist" "$PORTAL_BACKUP/launchagent.plist"

```

The data snapshot contains a consistent SQLite database, original assets, signed release history, device configuration, TLS material and content-signing keys. Keep the separate Android APK signing key and properties in a private backup too. Do not add extra files inside the inventory-verified data snapshot.

Verify restoration in a nonexistent, separate destination:

```

./portalctl restore "$PORTAL_BACKUP/data" "$PORTAL_PROJECT/private/restore-check"
./portalctl --data "$PORTAL_PROJECT/private/restore-check" health

```

Restore checks the inventory, database integrity, signatures and asset hashes. It does not switch the running service to that destination. Follow [operations and recovery](#) before changing a live data root.

For content rollback, choose an earlier compatible release in Releases & recovery. Rollback publishes a new, higher revision. Very old content without 30 current approved dates is rejected. This is independent of APK installation and is not an Android downgrade.

12. Know the offline and APK limits

Situation	Expected behavior
Mac offline, Portal powered	Cached content works within its validity window; bundled content remains as fallback, and local clock/alarms continue; new publications wait
Internet outage	Weather can show a labeled cached observation; stale transit predictions are suppressed
Home Wi-Fi off	Content remains cached; LAN Sonos and network providers are unavailable
Open lesson or ringing alarm	A complete content download can wait for safe adoption
Corrupt file, bad signature or unsupported schema	Download is rejected; the prior complete content remains
New native feature	Requires a compatible APK and the correct existing signing key

The separate app-release channel checks package, version, signing certificate, verification receipt, release signature, APK hash and live installer capability. The tested Gen1 setup reports `installer_unavailable_after_reboot` because it has no working daemon or proven fixed/auto-confirm dialog. Daily content OTA works; unattended APK updates are not promised.

The inherited upstream provisioning documentation describes overlay/installer fixes. Those are separate device changes, not prerequisites to displaying cached content, and were not silently enabled by this project. See [verification](#) for the measured scope.

13. Troubleshooting

Symptom	Check
Studio does not open	Run the server or inspect <code>launchctl</code> state and private <code>service-error.log</code> ; use port 9444 on this Mac
Studio key is rejected	Run access from the correct checkout/data root; a device/Fleet token cannot log in
Device never appears	Check actual package ID, bootstrap destination, consumed-file status, Wi-Fi interface, firewall and Bonjour/multicast reachability
TLS failure after copying setup	Preserve the original TLS files and enrollment; replacing a certificate needs explicit re-pairing
Host serves the wrong interface	Verify the Wi-Fi Device entry and <code>PORTAL_WIFI_INTERFACE</code> ; do not bind management to a public interface

Symptom	Check
Release is staged	Check for an open lesson, ringing alarm, or scheduled activation time
Publication fails	Check approval window, text/asset bounds, date, supported actions/components and draft conflict messages
settings_conflict	Refresh the acknowledged local version and intentionally resolve the setting; do not keep resending a stale base
Install reports signature mismatch	Keep the installed app and its data; obtain the original key or use the side-by-side debug package
Alarm or clock is wrong	Check device clock/timezone; daily content uses America/Los_Angeles, transport timestamps use UTC
J train or weather is for the wrong place	These are current source defaults; edit and rebuild the native provider implementation
APK command cannot find tools	Set ANDROID_HOME and JAVA_HOME in private/runtime-env.sh

For development checks:

```
source "$HOME/.venvs/contains-portal/bin/activate"
cd "$PORTAL_PROJECT"
PYTHONPATH=. python3 -m unittest discover -s tests -v
node tests/editor-save.test.cjs
cd "$PORTAL_ANDROID"
./gradlew :app:testDebugUnitTest --no-daemon
```

The publication baseline passed 23 publisher tests, the editor Save regression, and 465 Android JVM tests. GitHub test workflows also passed. The [design story](#) explains how the visual concepts were turned into native components; the [protocol](#) documents the data they accept.