TRIGLAV OS · QUINZEE · ADMINISTRATOR'S GUIDE

Volume 1: Quinzee Administrator's Guide

TOMAZ STIH

FIRST EDITION



Development edition · 16 November 2026. Commands using repository paths, Make targets or sample projects require access to the Quinzee source checkout. No public installer is available. Examples use disposable local environments.

Your path through Quinzee

Begin with a safe local installation, learn the module boundaries, then configure, operate and recover every service.

1. Start with a small, understood installation
2. Connect QHQ and make changes deliberately
3. Understand modules and their dependencies
4. Configure the cloud and its launch environment
5. Make names, certificates and permissions agree
6. Operate the shared Entra and LDAP directory
7. Control resources and infrastructure deployments
8. Administer storage, secrets and configuration
9. Keep events and messages moving
10. Run PostgreSQL, MySQL and Valkey safely
11. Operate Functions and App Service workloads
12. Observe the platform and the application separately
13. Back up, recover and rehearse failure
14. Diagnose failures from the outside inward
15. Find every control in QHQ

Commands marked **host** run from the repository root. Each exercise identifies its prerequisites, the place to run commands and the expected result. Use a disposable environment while learning.

1. Start with a small, understood installation

An administrator's first responsibility is to know which process owns which state. Quinzee Cloud serves the local Azure-compatible APIs. QHQ is a client of those APIs, not the server. The DNS process answers names. Separate helpers run SQL/cache engines and hosted applications. A running QHQ window does not prove that all of those processes are healthy.

This book starts with one local development installation. It then explains how to operate each part separately. Quinzee is a single-tenant compatibility platform with bounded service implementations; it is not an Azure control plane, a Windows domain controller or a highly available production directory. Treat its credentials, application data and backups as private even in a local lab.

Step 1: prepare the operator workstation

Use the verified Linux x86-64 platform. Install the repository's supported C++ build toolchain, Docker/Compose, Python 3 and OpenSSL. The host or development image needs CMake, a C++20 compiler, Make, Git, curl, Perl, bzip2, Bison and Flex. The real hosted .NET examples additionally need .NET SDK 10. QHQ's graphical viewer needs a Linux graphical session; offscreen screenshots do not.

Run these from the repository root:

```
make deps
make
make test
make certs
```

Dependencies are verified and cached in `.packages/`; binaries go into `bin/debug/`. The first run can need downloads and time to compile. A later cached build should reuse them. A missing compiler or failed check is an installation problem to fix before provisioning application data.

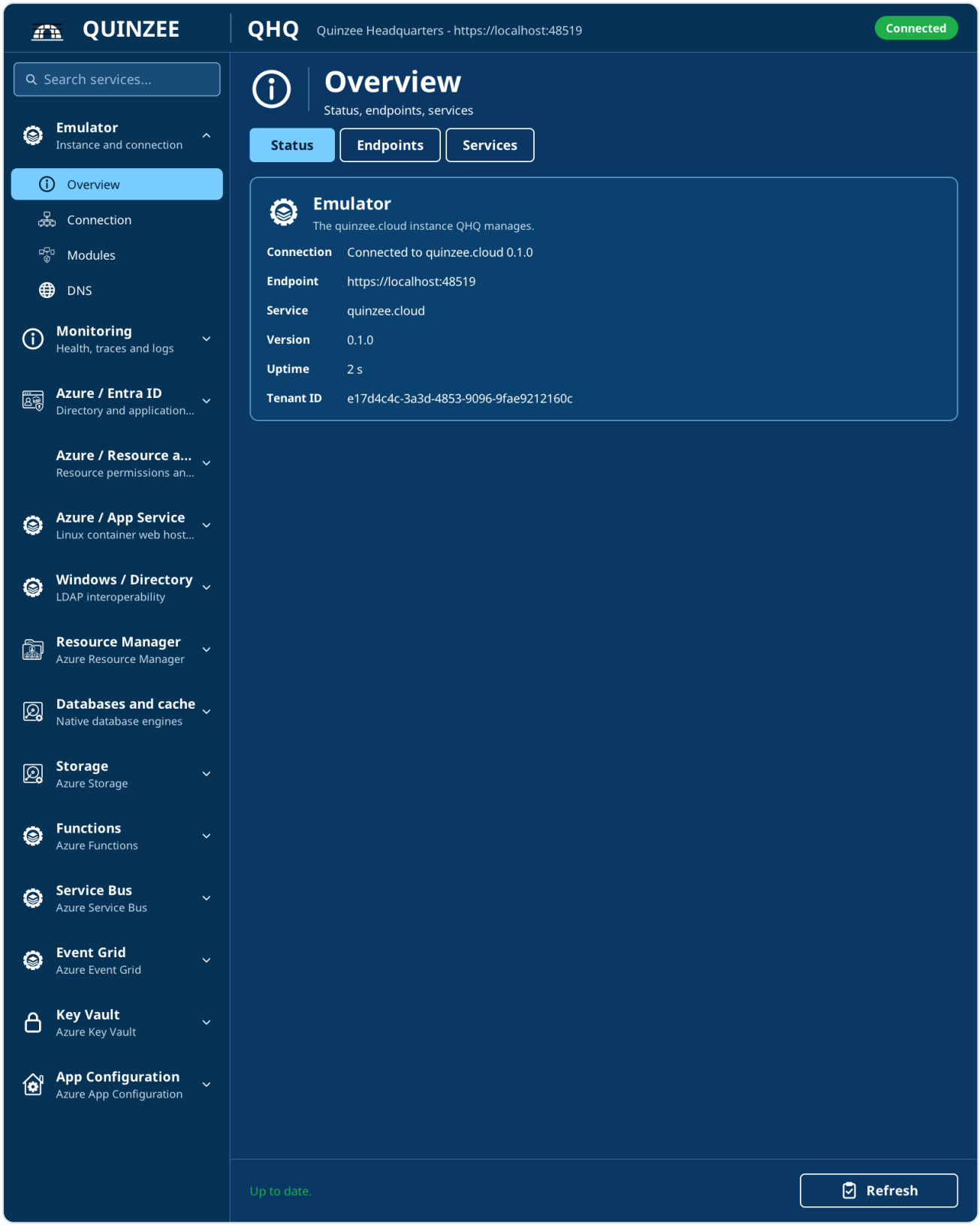
Step 2: start a default local environment

```
make qhq-run
```

This is the convenient operator lab: it starts the HTTPS cloud and DNS containers and opens QHQ through the Rasta viewer. By default, the host connects to `https://localhost:11443`, and state is in `.local/quinzee`. Certificate material is in `.local/certificates`. Bootstrap credentials

are generated at first startup in the private `credentials.json` inside the cloud data directory. There is no default human administrator password.

Wait for QHQ's Connected indicator. Open Emulator → Overview and verify the endpoint and tenant before changing anything. Then open Modules. All default modules should be loaded; LDAP's optional external listener can still be unconfigured. SQL or hosted apps need their helpers before their workloads run.



The screenshot uses synthetic resources and a temporary port. Your persistent installation's values differ. Screenshots throughout this guide are real QHQ captures; the UI atlas includes every page and its actual tabs.

Step 3: prove HTTPS independently

In another host terminal:

```
curl --fail --silent --show-error \
  --cacert .local/certificates/ca/quinzee-dev-root-ca.crt \
  https://localhost:11443/health
```

Expect a healthy `quinzee.cloud` response. A refused connection means listener, process or port mapping trouble. A certificate error means trust, hostname or certificate trouble. Fix the actual layer; adding `-k` would hide the problem that your SDK clients will encounter.

Step 4: stop without deleting the installation

Close QHQ and run `make qhq-env-down`. Containers stop; the data directory remains. Starting again reuses the tenant and resources. For cloud-only use, `make cloud-up` and `make cloud-down` control the Compose services. Stopping a helper or unloading a module is not the same as stopping its workloads; later chapters give the correct order for each service.

Do not use `make purge` as routine shutdown: it is destructive. Resource-group Delete is also a data-removal operation, not a way to pause a project.

Step 5: distinguish the two standard labs

`make application-up` runs the browser demonstration in its own persistent folder, `.local/application/development`. It can choose different ports, a different Docker network and different credentials. `make application-test` uses an isolated disposable environment. Never point QHQ at one lab while reading credentials or certificates from the other. Record each installation's origin, data path, trust root and network as one set.

Related reading

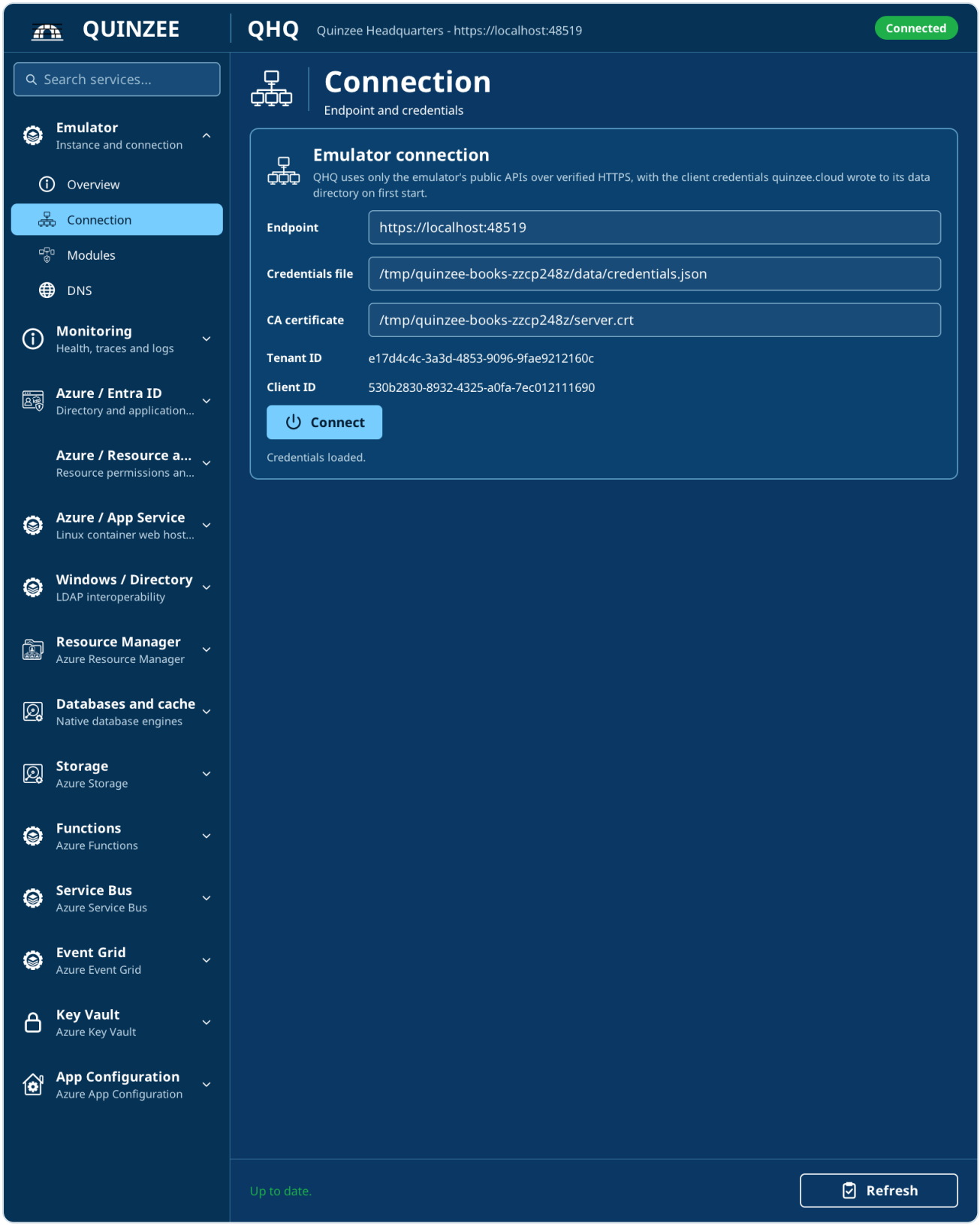
- The Book of Quinzee: Meet Quinzee and its services
- The Book of Quinzee: Run your first application
- Quinzee Internals: Architecture and trust boundaries
- Quinzee Internals: The compatibility application
- Scenario overview: Developing for Azure Without Using Azure
- Scenario overview: Learning How the Cloud Works
- Scenario overview: Building a Complete Cloud Application from Scratch

2. Connect QHQ and make changes deliberately

QHQ uses authenticated HTTPS APIs. It does not open the cloud's databases or manage Docker directly. This matters when diagnosing a failed Create: the UI can submit a valid request while a missing external helper prevents the resource from becoming ready.

Step 1: connect to the intended instance

Open Emulator → Connection. Enter its HTTPS endpoint, the path to that instance's bootstrap `credentials.json`, and the trusted public CA certificate. Connect and check the tenant/subscription shown under Azure / Entra ID → Tenant. The credentials file is not an access-token file or an application settings file. A remote QHQ workstation needs its own securely delivered credential file and public CA; it does not need the server's TLS private key or CA signing key.



For direct launch, use the following shape, substituting your installation paths:

```
bin/debug/qhq --endpoint https://localhost:11443 \
  --credentials .local/quinzee/credentials.json \
  --ca-file .local/certificates/ca/quinzee-dev-root-ca.crt
```

Step 2: understand selection and scope

Navigation categories start collapsed. Expand the required category and choose a page. Tab/Shift+Tab move between controls. A Browse list selection determines what Save selected, Delete selected, Start or Stop will affect. Entering a new name does not select an existing object. Scope fields use ARM resource IDs such as `/subscriptions/<UUID>`; display names are not interchangeable with IDs.

On the identity forms, select a row in Browse, then open Create / Edit. Create makes a new object; Save selected updates the selected one. Refresh replaces inventory and clears selection, so select again before editing. Confirm the Delete checkbox only after checking the selected ID. On resource pages, also check the subscription and resource group. They can differ between pages.

Step 3: protect values that are not inventory

Inventory omits client secrets, account keys, Function keys and private app settings. A newly generated client secret is shown once in the Credentials tab; copy it into private configuration and clear the widget. App setting values use a masked input and are cleared after submission. Screenshots do not reveal these values, and this book does not teach you to paste them into resource names, tags or diagnostic logs.

Step 4: read asynchronous state

Create acknowledges a request. Deployment operations and external runtimes can still be Updating. Refresh and inspect the resulting state. A site with no image is intentionally Stopped; an app awaiting a helper is not ready. A stale helper heartbeat stops gateway forwarding even if the container still exists. For recovery, inspect the operator logs before submitting another Create.

QHQ launch settings

Option	Value and effect
`--endpoint`	HTTPS origin; default `https://localhost:11443`
`--credentials`	Bootstrap credential file; default `/var/lib/quinzee/credentials.json`
`--ca-file`	Trust certificate for the endpoint; default `/etc/quinzee/tls/ca.crt`
`--backend`	`rasta`, `framebuffer` or `offscreen`; default `rasta`
`--width`, `--height`	320–7680 pixels; defaults 1600 × 1000
`--scale`	0–4; 0 selects automatic scaling

Option	Value and effect
`--framebuffer`	Device path or Rasta framebuffer file
`--host`, `--port`	Rasta viewer address and UDP port; use the viewer's matching settings
`--assets`	Asset directory; defaults beside the executable
`--screenshot`	Output PNG; selects offscreen rendering and exits after loading
`--page`, `--tab`	Navigation ID and zero-based tab for screenshots; defaults overview/0
`--help`	Show supported arguments

The Rasta viewer settings are local display transport, not cloud HTTP ports. Do not confuse its `--port` with the cloud's `--port`. The safe capture procedure in the final chapter uses only disposable credentials and known public views.

Related reading

- The Book of Quinzee: Meet Quinzee and its services
- The Book of Quinzee: Run your first application
- Quinzee Internals: Architecture and trust boundaries
- Quinzee Internals: The compatibility application
- Scenario overview: Developing for Azure Without Using Azure
- Scenario overview: Learning How the Cloud Works
- Scenario overview: Building a Complete Cloud Application from Scratch

3. Understand modules and their dependencies

A module is a service implementation loaded inside `quinzee.cloud`. A provider is the module's ARM-facing resource interface. A capability is a typed service that another module requires, such as authentication or the shared directory. An external helper is a separate process. These are four different boundaries.

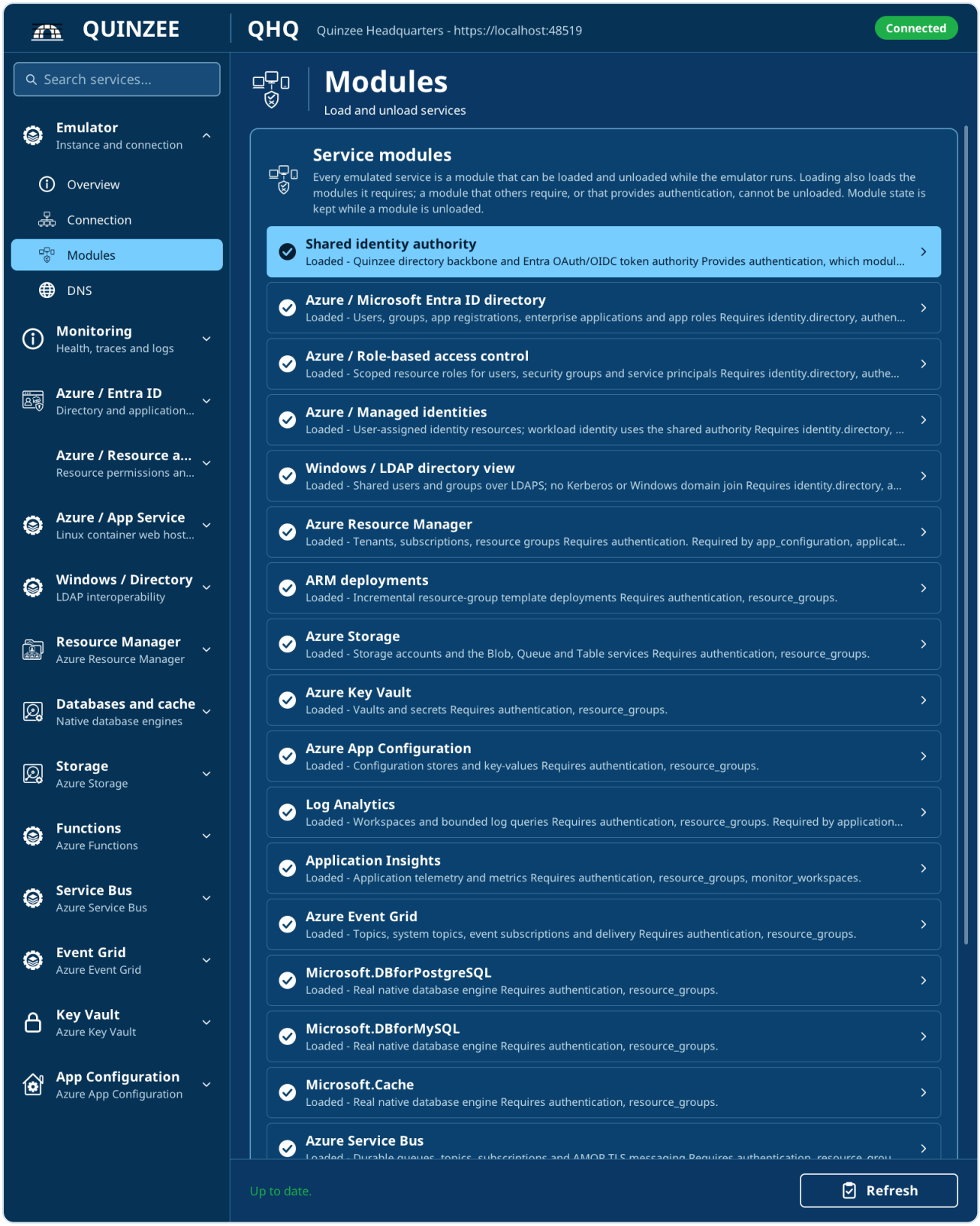
Step 1: read the module inventory

Open Emulator → Modules. Inspect each module's loaded state, what it provides, what it requires and which loaded modules depend on it. The table below follows the running catalogue rather than inferring dependencies from names.

Module ID	Family	Required modules	Data/process dependency
`identity`	Shared	None	Private JSON, or dedicated PostgreSQL when configured
`entra_directory`	azure	`identity`	Shared directory; no second database
`azure_rbac`	azure	`identity`	Shared directory
`managed_identities`	azure	`identity`	Shared directory
`windows_ldap`	Windows	`identity`	Shared PostgreSQL plus optional external OpenLDAP
`resource_manager`	control-plane	`identity`	Private durable JSON
`deployments`	control-plane	`identity`, `resource_manager`	Private history and in-process operations
`storage`	resource-provider	`identity`, `resource_manager`	Blob files, queue persistence, embedded SQLite for tables
`key_vault`	resource-provider	`identity`, `resource_manager`	Private durable secret/version files
`app_configuration`	resource-provider	`identity`, `resource_manager`	Private durable settings/revisions

Module ID	Family	Required modules	Data/process dependency
`log_analytics`	resource-provider	`identity`, `resource_manager`	Bounded durable workspace files
`application_insights`	resource-provider	`identity`, `log_analytics`, `resource_manager`	Component metadata plus Log Analytics workspace
`event_grid`	resource-provider	`identity`, `resource_manager`	Embedded SQLite delivery outbox
`postgresql`	managed-engine	`identity`, `resource_manager`	External PostgreSQL container and labelled data volume
`mysql`	managed-engine	`identity`, `resource_manager`	External MySQL container and labelled data volume
`managed_redis`	managed-engine	`identity`, `resource_manager`	External Valkey container and labelled data volume
`service_bus`	resource-provider	`identity`, `resource_manager`	Embedded SQLite; Proton AMQP listener
`functions`	resource-provider	`identity`, `resource_manager`	Private records and external Docker hosting helper
`app_service`	azure	`functions`	Functions shared Microsoft.Web host and runtime
`dns`	infrastructure	`identity`	Generated zone/Corefile; separate CoreDNS process

The identity module is retained because administrative APIs need authentication. The shared authority is not the same module as Entra's management frontend. Loading `windows_ldap` makes its status page available; a PostgreSQL configuration and LDAP socket are still required for actual LDAP service. The application PostgreSQL provider does not automatically supply the directory's PostgreSQL. Keep that dedicated directory backend outside workloads whose management depends on the directory itself.



Step 2: understand optional relationships

Event Grid collects the queue capability at delivery time. StorageQueue delivery therefore needs Storage to be loaded even though it is not a mandatory module dependency. Blob events require the Event Grid event publisher to be available. DNS collects service names from loaded providers. A declaration can load without CoreDNS running, but clients will not

resolve names through that external server. The engine modules load without Docker; their servers cannot start without the engine supervisor. App Service requires Functions' hosting capability because one provider owns Microsoft.Web plans, records and dispatch.

Step 3: load only what a lab needs

`--modules` selects an initial set and loads required dependencies. Unselected modules remain in the catalogue for later loading. To avoid a surprise, inspect the resulting Modules page instead of assuming the command-line list equals the final loaded set. For example, a Storage-only resource lab still needs the shared authority and resource-group provider. Explicit identity frontend lists should name `entra_directory`, `azure_rbac`, `managed_identities` and `windows_ldap` according to the services you intend to expose.

Step 4: unload in dependency order

1. Stop active hosted apps or database servers if their processes must stop.
2. Select the dependent module and unload it. App Service must unload before Functions; Application Insights before Log Analytics.
3. Check the module's state and any refusal explaining loaded dependants.
4. Reload and confirm that persisted resources return.

Unloading stops that module's routing/workers while retaining its data. It does not delete resources and does not generally stop external containers. Deleting a resource is different: the provider may enqueue container/volume removal. Never edit a persisted dependency or resource record to force an unload.

Platform facilities that are not extra modules

Health, module management, certificate metadata and operational telemetry are host facilities. There is no separate `certificates` catalogue module to add to `--modules`. The generated zone and CoreDNS container belong to the DNS workflow; CoreDNS is not another Azure service module. These distinctions explain why the number of pages, services, containers and module IDs is not the same.

Related reading

- The Book of Quinzee: Stop, restart and clean up safely
- The Book of Quinzee: Build Quinzee from source
- Quinzee Internals: Build System and Dependency Cache
- Quinzee Internals: Containers and runtime ownership
- Scenario overview: Working Without an Internet Connection
- Scenario overview: Building a Personal Cloud on a Home Server

- Scenario overview: AI Architecture Reviews

4. Configure the cloud and its launch environment

Configuration has three layers. The native executable reads an optional JSON file and command-line arguments. Compose/Make launchers substitute their own variables and mounts. Individual resources are configured through their service APIs. Changing a shell variable does not update a running resource, and adding an Azure resource property does not change the server's listening port.

Step 1: choose one source of startup configuration

`--config FILE` reads a JSON object of at most 64 KiB. All members are optional. Command-line options override the file. Unknown names and invalid types fail startup instead of being ignored. The native executable does not directly read Compose environment variables. Relative paths are relative to its working directory; use absolute paths in a persistent service definition.

Save a private configuration such as:

```
{
  "address": "127.0.0.1",
  "port": 11443,
  "tlsCertificate": "/srv/quinzee/tls/server.crt",
  "tlsKey": "/srv/quinzee/tls/server.key",
  "caCertificate": "/srv/quinzee/tls/ca.crt",
  "dataDirectory": "/srv/quinzee/data",
  "advertiseAddress": "127.0.0.1",
  "logLevel": "info",
  "azureEmulation": false
}
```

Create the directories for the service's non-root OS user, install matching certificates, and run `bin/debug/quinzee.cloud --config /srv/quinzee/cloud.json`. Use `bin/release/quinzee.cloud` after `make PRESET=release` for a release build. There is no plaintext HTTP mode for the cloud executable.

Complete cloud startup settings

JSON member	CLI option	Default and meaning
`address`	`--address`	Numeric IP `127.0.0.1`; listener binding

JSON member	CLI option	Default and meaning
`port`	`--port`	HTTPS 11443; explicit ports 1–65535
`serviceBusPort`	`--service-bus-port`	Omitted disables AMQP; explicit 1–65535 enables TLS AMQP
`tlsCertificate`	`--tls-cert`	`/etc/quinzee/tls/server.crt`; PEM chain
`tlsKey`	`--tls-key`	`/etc/quinzee/tls/server.key`; matching private key
`caCertificate`	`--ca-cert`	`/etc/quinzee/tls/ca.crt`; development root metadata and outbound trust where used
`dataDirectory`	`--data-dir`	`/var/lib/quinzee`; durable instance state
`modules`	`--modules`	All catalogue modules by default; JSON array or comma-separated CLI IDs
`advertiseAddress`	`--advertise-address`	IPv4 `127.0.0.1`; generated service DNS targets, not listener binding
`storageDomain`	`--storage-domain`	`storage.quinzee.cloud`; Blob/Queue/Table suffix
`vaultDomain`	`--vault-domain`	`vault.quinzee.cloud`
`appConfigurationDomain`	`--app-configuration-domain`	`appconfig.quinzee.cloud`
`azureEmulation`	`--azure-emulation`	false / `off`; CLI accepts `on` or `off`
`emulationPort`	`--emulation-port`	443; separate Azure-name HTTPS listener
`emulationTlsCertificate`	`--emulation-tls-cert`	`/etc/quinzee/azure-emulation/server.crt`
`emulationTlsKey`	`--emulation-tls-key`	`/etc/quinzee/azure-emulation/server.key`
`directoryPostgresConfig`	`--directory-postgres-config`	Absent: legacy private JSON directory; present: private libpq settings file
`ldapSocket`	`--ldap-socket`	Absent: no LDAP backend socket; requires PostgreSQL when present
`otlpEndpoint`	`--otlp-endpoint`	Empty: external telemetry export disabled; HTTPS origin
`otlpCaCertificate`	`--otlp-ca`	Optional CA for telemetry export

JSON member	CLI option	Default and meaning
`logLevel`	`--log-level`	`info`; debug, info, warning or error

`--config` and `--help` are CLI-only. Explicit TLS certificate/key overrides must be supplied as a pair. Service domains must be lowercase DNS names with at least two labels and must also be present in certificates and client DNS. TLS files are loaded at startup: replacing a file alone does not reload it. The Azure emulation listener is provisioned at startup; its DNS/route switch can then be changed in QHQ. Enabling a switch cannot create a missing TLS listener.

Compose and Make values

These values configure the standard launchers, not arbitrary Azure resources:

Name	Default / effect
`PRESET`	`debug`; native build/test preset, also release/asan/tsan
`DOCKER_PRESET` / `TRIGLAV_DOCKER_PRESET`	release; image build preset passed by launcher/Compose
`JOBS`	Host CPU count; keep parallel builds enabled
`TRIGLAV_HOST_BIND`	`127.0.0.1`; published host address
`QUINZEE_HTTPS_PORT`	11443 on host; container remains 11443
`QUINZEE_AMQP_PORT`	5671 on host; container remains 5671
`QUINZEE_DNS_PORT`	10053 on host; container DNS is 53 TCP/UDP
`QUINZEE_NET_PREFIX`	`172.29.53`; /24 network, cloud .10 and DNS .53
`QUINZEE_DATA_DIR`	`.local/quinzee`; mounted instance data
`QUINZEE_TLS_DIR`	`.local/certificates/quinzee.cloud`; service TLS mount
`QUINZEE_EMULATION_TLS_DIR`	`.local/certificates/azure-emulation`; Azure-name TLS mount
`QUINZEE_UID`, `QUINZEE_GID`	Compose defaults 65534; development launcher supplies the operator IDs
`ENGINE_DATA_DIR`	Absolute `.local/quinzee`; cloud state observed by helpers
`ENGINE_NETWORK`	`triglav-http_quinzee`; existing Docker network
`FUNCTIONS_STATE_DIR`	Absolute `.local/functions-helper`; private hosting-helper state
`FUNCTIONS_REGISTRY`	`images.json` in that helper state directory

Name	Default / effect
`QHQ_WIDTH`, `QHQ_HEIGHT`	Viewer dimensions; launcher defaults 1440 × 900
`QHQ_RASTA_PORT`	Viewer UDP port; launcher default 5100
`QHQ_PRESET`	QHQ launcher's selected binary preset
`QUINZEE_CERT_DIR`	Development launcher's certificate root, normally `.local/certificates`
`QUINZEE_OTLP_ENDPOINT`, `QUINZEE_OTLP_CA`	Optional engine-helper telemetry destination and trust

Use `docker compose config` to inspect resolved mounts, addresses and commands before startup. It can show interpolated private values in other projects; keep its output local. The default Compose file supplies startup arguments explicitly; to add a JSON config, mount it and override the service command in a separate Compose override file. Passing a host path that is not mounted into the container cannot work. Prefer a new private instance directory for a new environment.

Change one setting and verify the result

1. Record the old value, origin, paths and active resources.
2. Stop incoming writes and follow the relevant workload shutdown procedure.
3. Edit the JSON or launcher configuration, preserving private permissions.
4. Restart, check `/health`, QHQ Connection and the affected service endpoint.
5. If startup fails, restore the prior configuration; do not modify state files to make incompatible settings appear acceptable.

Fixed resource limits, SQL engine versions and supported API properties appear in later chapters. They are not hidden environment variables you can tune.

Related reading

- The Book of Quinzee: Stop, restart and clean up safely
- The Book of Quinzee: Build Quinzee from source
- Quinzee Internals: Build System and Dependency Cache
- Quinzee Internals: Containers and runtime ownership
- Scenario overview: Working Without an Internet Connection
- Scenario overview: Building a Personal Cloud on a Home Server
- Scenario overview: Giving AI Coding Agents Their Own Cloud

5. Make names, certificates and permissions agree

A connection succeeds only when three facts agree: the client resolves the name to the intended server, the server presents a certificate for that name, and the client trusts its issuer. An IP address that responds to ping proves none of the last two facts. Keep this sequence in mind when troubleshooting every service.

Step 1: distinguish native and Azure names

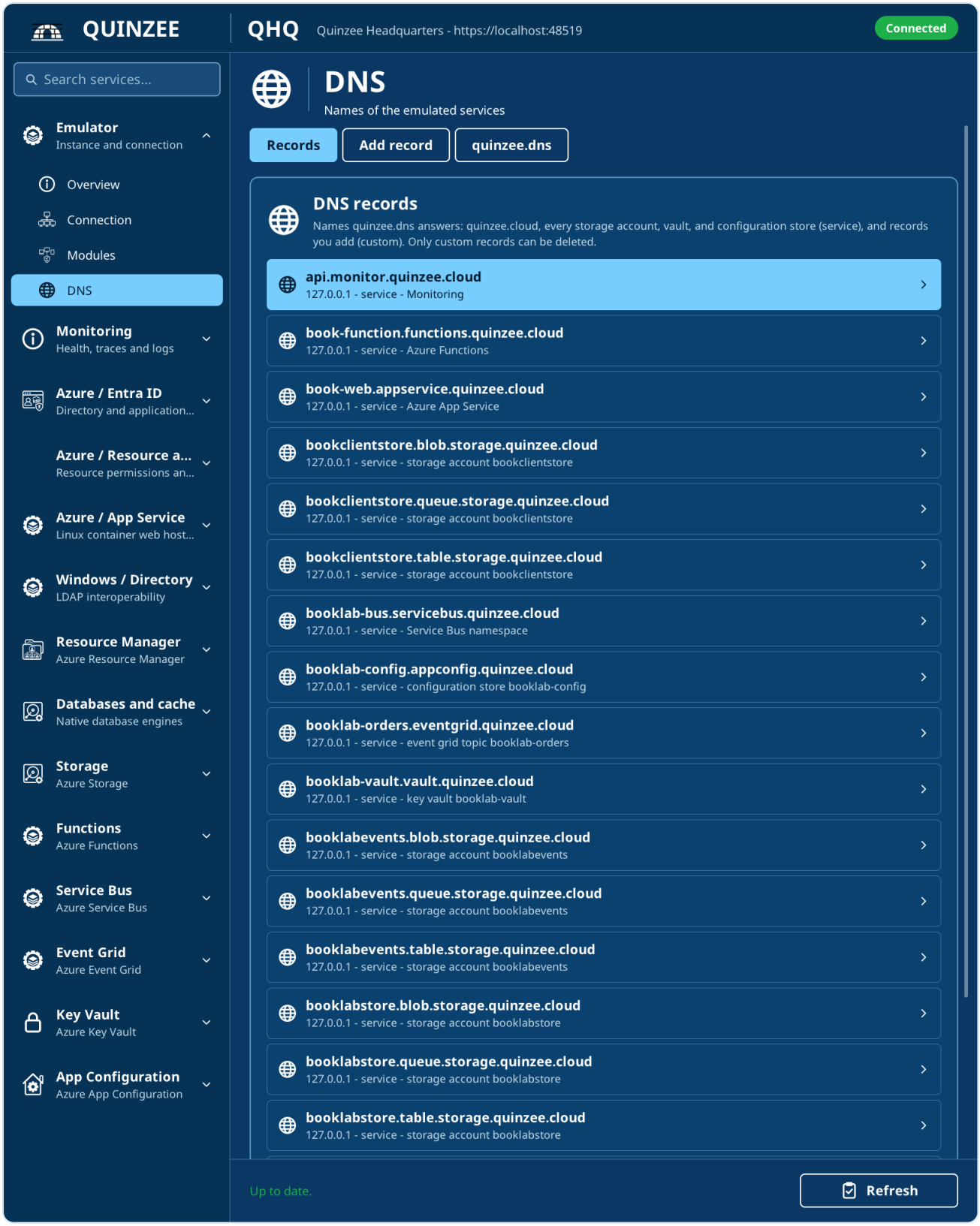
Native clients use names such as `quinzee.cloud`, `<account>.blob.storage.quinzee.cloud`, `<namespace>.servicebus.quinzee.cloud` and `<app>.appservice.quinzee.cloud`. Azure-name emulation adds supported Microsoft hostnames on a separate listener. It is for isolated clients that explicitly use Quinzee's DNS and development trust; it does not contact Azure. Do not redirect a workstation's general DNS to the emulator.

The standard network gives the cloud address `<prefix>.10` and CoreDNS `<prefix>.53`. Joining the Docker network alone does not select its DNS. Pass `--dns <prefix>.53` to a separate client container and mount only public trust material. Native database names point to the engine container's own address, not to the cloud HTTPS gateway.

Step 2: inspect and change DNS

Open Emulator → DNS → Records. Check generated records for the loaded services. Use Add record for a deliberate custom record, and use the DNS server tab to inspect the generated server configuration and Azure emulation state. Changes publish the zone/Corefile read by the separate CoreDNS process. If records look right in QHQ but clients cannot resolve them, check CoreDNS's process, network, DNS selection and cache before changing the record again.

A custom record accepts a lowercase DNS name, optional IPv4 `address` (defaulting to the advertised emulator address), and optional descriptive `detail`. Generated service names are reserved and cannot be overwritten. Deleting a custom record removes only that override. The generated zone has a fixed five-second TTL and a serial that increases on updates. AAAA and wildcard records are unsupported; changing a record does not change a certificate.



The DNS module supports its documented local record workflow; it is not the Azure DNS zone ARM provider. Do not put a `Microsoft.Network/dnsZones` resource in a template and expect this module to configure CoreDNS.

Step 3: issue and distribute certificates

`make certs` creates the development root and service certificates under `.local/certificates`. Existing material is reused. To deliberately replace it, use `make certs FORCE=1` only after planning how every affected client will trust the replacement roots. Keep CA signing keys on the operator host; containers receive service leaf keys or public roots according to their role.

Native cloud defaults look for `server.crt`, `server.key` and `ca.crt` in `/etc/quinzee/tls`. The standard Compose mounts supply those paths. Azure-name certificates use a separate name-constrained root and mount. Functions and App Service helpers also have separate name-constrained runtime CAs for upstream verification. An app presenting a valid certificate for another domain must still be rejected.

After renewal, restart the affected listeners and distribute the new public CA bundle to clients. Verify HTTPS without disabling certificate checks. Native engine clients may need the engine CA and its revocation responder reachable; copying the cloud's HTTPS root alone is insufficient for their certificates.

Step 4: separate the identities involved

The OS service user owns files and processes. The bootstrap application identity administers Quinzee. A directory user signs in as a person. A service principal is an application's tenant identity. A managed identity is a service principal whose token credential is supplied by a hosted environment. A SQL administrator is an engine-local database login. None of these names makes the others the same account, and password reset at one boundary does not reset all of them.

Bootstrap credentials grant broad local control. New principals/users need supported ARM or Service Bus role assignments. Other data planes retain their explicit compatibility boundaries. A role named Owner is not a directory administrator, and an app role called Orders.Read is not an Azure resource role. Never infer an authorization guarantee solely from an Azure-shaped property.

Step 5: protect the installation at rest

Private state, credentials, signing material and runtime mailboxes use restricted permissions. The emulator does not provide blanket encryption at rest for all application data. Use OS ownership, encrypted storage where required, private backups and a restricted management network. Do not expose the bootstrap credential file through a web server or mount it into ordinary application containers. Keep the Docker socket with the explicitly operated helpers.

Related reading

- The Book of Quinzee: Certificates, DNS and Azure URLs
- Quinzee Internals: TLS, identity and DNS

- Quinzee Internals: Follow a packet through names, sockets and namespaces
- Scenario overview: Running Applications at the Edge
- Scenario overview: Building an Isolated Private Application Environment

6. Operate the shared Entra and LDAP directory

The identity authority stores applications, service principals, managed identities, users, groups and assignments. Entra's HTTPS APIs and LDAP read the same authoritative records. There is no password synchronization job. LDAP is a protocol view, not the underlying database and not a second source of users.

Step 1: decide when to use PostgreSQL

Without a PostgreSQL configuration, existing installations retain private JSON directory storage. Actual LDAP service requires the PostgreSQL backend. Use a dedicated database owned by the directory service user. Do not place it inside an application resource group whose deletion could remove your management identity store. The ordinary PostgreSQL module provisions application databases; it does not automatically create or own the directory database.

For a local directory service, prepare a TLS server certificate valid for `localhost`, its key and CA. The standard native cloud development certificate contains localhost and can be used for this isolated lab:

```
python3 infrastructure/directory/postgres.py start \
  --directory "$PWD/.local/directory-postgres" \
  --certificate "$PWD/.local/certificates/quinzee.cloud/server.crt" \
  --key "$PWD/.local/certificates/quinzee.cloud/server.key" \
  --ca "$PWD/.local/certificates/ca/quinzee-dev-root-ca.crt"
```

The helper uses the repository's pinned PostgreSQL image and writes a mode-0600 `postgres.json` inside its mode-0700 directory. It publishes localhost port 15432. Use `--port` for a different port; 0 requests an automatically assigned host port and the generated configuration records it. `start` of an existing owned container preserves its established configuration; editing the launch arguments does not reconfigure that container. `stop --directory <same-path>` stops it and preserves data. Its folder path is part of the ownership identity; do not rename it casually.

This loopback connection file is for a native cloud process on the same host. Inside a cloud container, localhost names that container. For that deployment, provide a reachable database hostname, a matching certificate and mounted trust paths; do not copy the native loopback configuration unchanged.

Step 2: migrate deliberately

1. Stop directory changes and take a consistent backup of existing cloud data.
2. Check the generated PostgreSQL connection file and trust path.
3. Start the cloud with `--directory-postgres-config <absolute-file-path>`.
4. Verify the existing tenant, application IDs, principals and users in QHQ.
5. Restart and verify them again before allowing new changes.

For a separately supplied PostgreSQL service, the private JSON file accepts only string-valued `host`, `port`, `dbname`, `user`, `password` and `sslrootcert`. TCP uses `verify-full`; the host must match the database certificate. An absolute Unix socket directory in `host` selects local socket transport. Arbitrary libpq URIs, environment service files and connection-string expansion are not accepted. The file must be a regular private file, not a public configuration snippet.

The first configured start imports the existing directory with IDs and credentials preserved and records a destination marker. PostgreSQL then becomes authoritative. Removing its configuration, changing its destination or losing its tenant row fails startup instead of falling back to the old JSON. Do not delete the marker to bypass that protection. Back up both the database and cloud data directory.

Internally, one row in `public.quinzee_directory` holds a bounded JSONB snapshot per tenant. One PostgreSQL session owns an advisory writer lock. A second writer fails startup. Every committed mutation publishes a consistent new snapshot. Database connection loss denies directory access; recover PostgreSQL and restart Quinzee to reacquire ownership. There is no automatic reconnect using stale permission data, and direct SQL edits are unsupported.

Step 3: enable the LDAP frontend

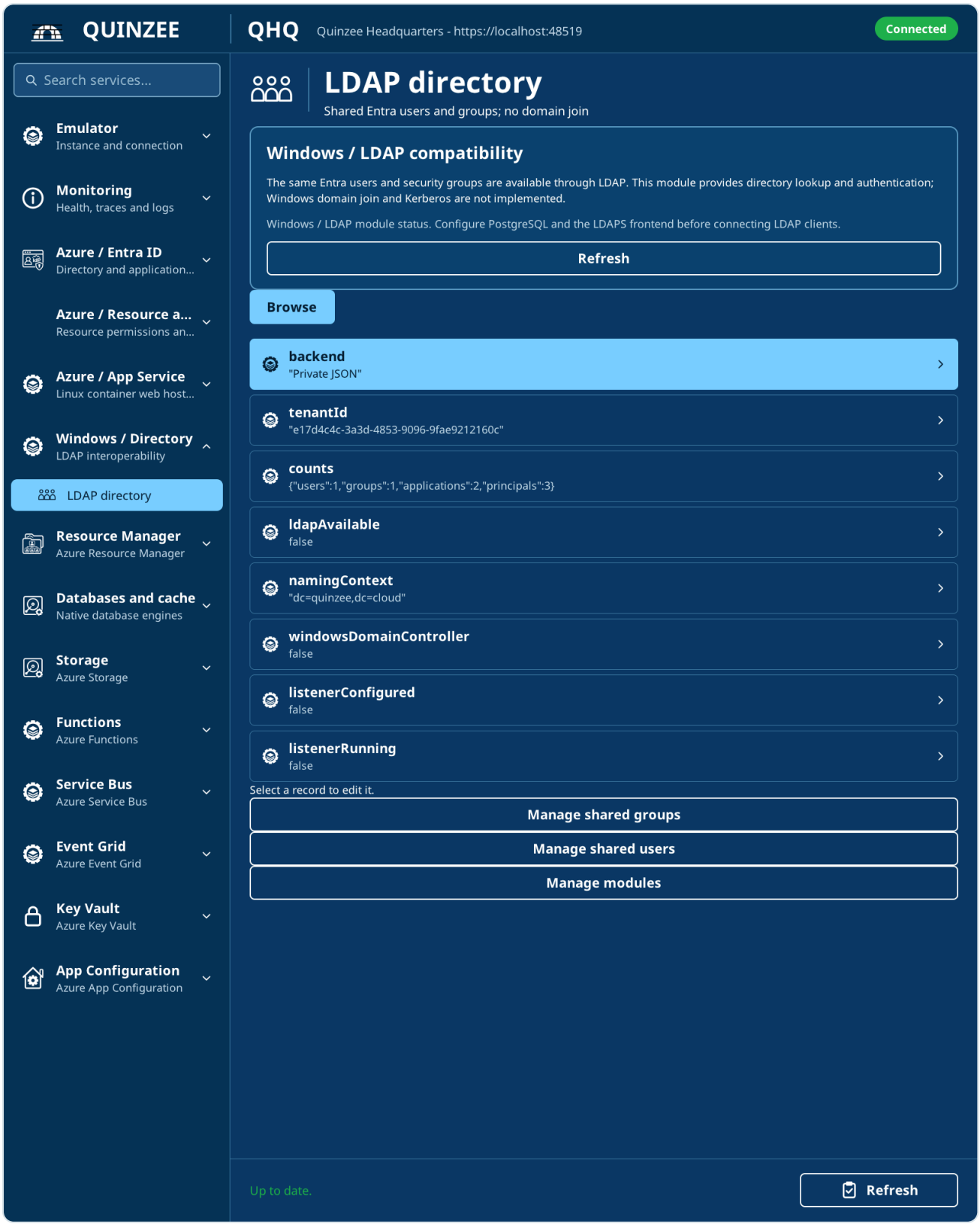
Build the optional server with `make directory-deps` and note the reported OpenLDAP installation prefix. Create a private socket parent:

```
install -d -m 700 .local/directory-run
```

Start the native cloud with its normal TLS/data settings plus `--directory-postgres-config <private-postgres.json>` and `--ldap-socket <absolute-path>/.local/directory-run/directory.sock`. Ensure `windows_ldap` is loaded. In another terminal, as the same OS user:

```
python3 infrastructure/directory/ldap.py \
  --prefix /absolute/path/to/installed-openldap \
  --socket "$PWD/.local/directory-run/directory.sock" \
  --certificate "$PWD/.local/certificates/quinzee.cloud/server.crt" \
  --key "$PWD/.local/certificates/quinzee.cloud/server.key" \
  --ca "$PWD/.local/certificates/ca/quinzee-dev-root-ca.crt" \
  --runtime "$PWD/.local/ldap-runtime" \
  --address 127.0.0.1 --port 11636
```

All six path options shown are required except address/port, whose defaults are 127.0.0.1/11636. OpenLDAP terminates TLS and forwards bind/search/compare to the private mode-0600 socket. Run it with the cloud's UID. The socket parent must remain 0700. The cloud refuses to replace an existing socket; after an unclean exit, verify there is no live owner before removing a stale socket. Ctrl+C stops the LDAP frontend without deleting directory records.



Step 4: administer users and groups

In Azure / Entra ID → Users, create a username such as `alice@example.test`, display name and 12–1,024-byte password. Account enabled controls sign-in. Select a user and use Save selected to rename, disable or reset its password. Forced change at next sign-in is not implemented. Password hashes and salts stay in the shared authority. Disabling affects LDAP and OAuth authentication.

In Groups, create a security group with a stable account name, select it, and use Members to add users or groups. Cycles are rejected. IDs remain stable; display names can change. POSIX numeric IDs start at 200000, are monotonically allocated and are not reused. Metadata such as `/home/<username>` does not create home directories or local OS accounts on the server.

The POSIX LDAP view uses `ou=users` and `ou=groups` below `dc=quinzee,dc=cloud`. It exposes direct user memberships in `memberUid`. The Windows-oriented view is below `ou=windows,dc=quinzee,dc=cloud`, with user and group subtrees. It exposes direct `member`/`memberOf`, nested group DNs, binary Microsoft-order `objectGUID`, `userPrincipalName`, stable generated `sAMAccountName`, and enabled/disabled `userAccountControl`. It has no SIDs, security descriptors, KDC, Kerberos, domain join or AD replication.

Verify using the real LDAP client, supplying the password interactively:

```
LDAPTLS_CACERT="$PWD/.local/certificates/ca/quinzee-dev-root-ca.crt" \
  ldapsearch -LLL -x -H ldaps://127.0.0.1:11636 \
  -D 'uid=alice@example.test,ou=users,dc=quinzee,dc=cloud' -W \
  -b 'dc=quinzee,dc=cloud' '(objectClass=posixAccount)'
```

Use a certificate containing the selected IP SAN or a matching DNS hostname. OpenLDAP's localhost handling can resolve to the machine hostname; do not work around a mismatch by disabling verification. Anonymous enumeration and LDAP writes are refused. Provision SSSD/PAM/NSS on Linux clients separately; Quinzee never changes the host's primary login configuration.

Step 5: manage application and resource permissions

Create App registrations, then create their Enterprise applications (service principals). Keep client ID, registration object ID and principal ID distinct. Credentials → Add secret returns the secret once; remove its key ID to revoke it. At most four client secrets are retained. Public clients use exact registered redirect URIs and no client secret. The local application enabled property is an emulator policy, not a claim that every Graph property is implemented.

Define App roles on the registration, then assign them to users, groups or service principals through App role assignments. A group app-role grant applies to direct user members only. Define `User` or `Application` allowed member types appropriately. Disable a role, remove its assignments, then delete the definition. Assignment required blocks unassigned access to

that Enterprise application. Expose an API at `api://<client-id>`; tokens for its `.default` scope carry its role values and client-ID audience. Existing external API tokens can remain usable until expiry; applications must validate their own required roles.

Azure / Resource access → Role assignments uses subscription/group/resource scopes. Supported roles are Owner, Contributor, Reader and Service Bus Data Owner/Sender/Receiver. Group-based resource RBAC includes nested user membership. User-assigned identities are separate ARM resources and survive app deletion; system identities are owned by their app. Managed identities cannot password-bind to LDAP. Attach them to supported hosted workloads to obtain tokens.

Directory management remains bootstrap-admin-only. ARM Owner and application roles do not grant that privilege. MFA, federation, Conditional Access, delegated consent and full Graph/directory emulation are outside this implementation.

Related reading

- The Book of Quinzee: Sign in and create a resource group
- The Book of Quinzee: Give applications and people the right identity
- Quinzee Internals: TLS, identity and DNS
- Quinzee Internals: One directory behind HTTPS and LDAP
- Scenario overview: Testing Security and Access Controls

7. Control resources and infrastructure deployments

A subscription is a local administrative boundary. A resource group owns related resources and gives deletion a concrete scope. A deployment is a recorded attempt to create or update resources. It is not their owner: deleting deployment history does not delete the resources that the deployment created.

Step 1: create a group and set its location

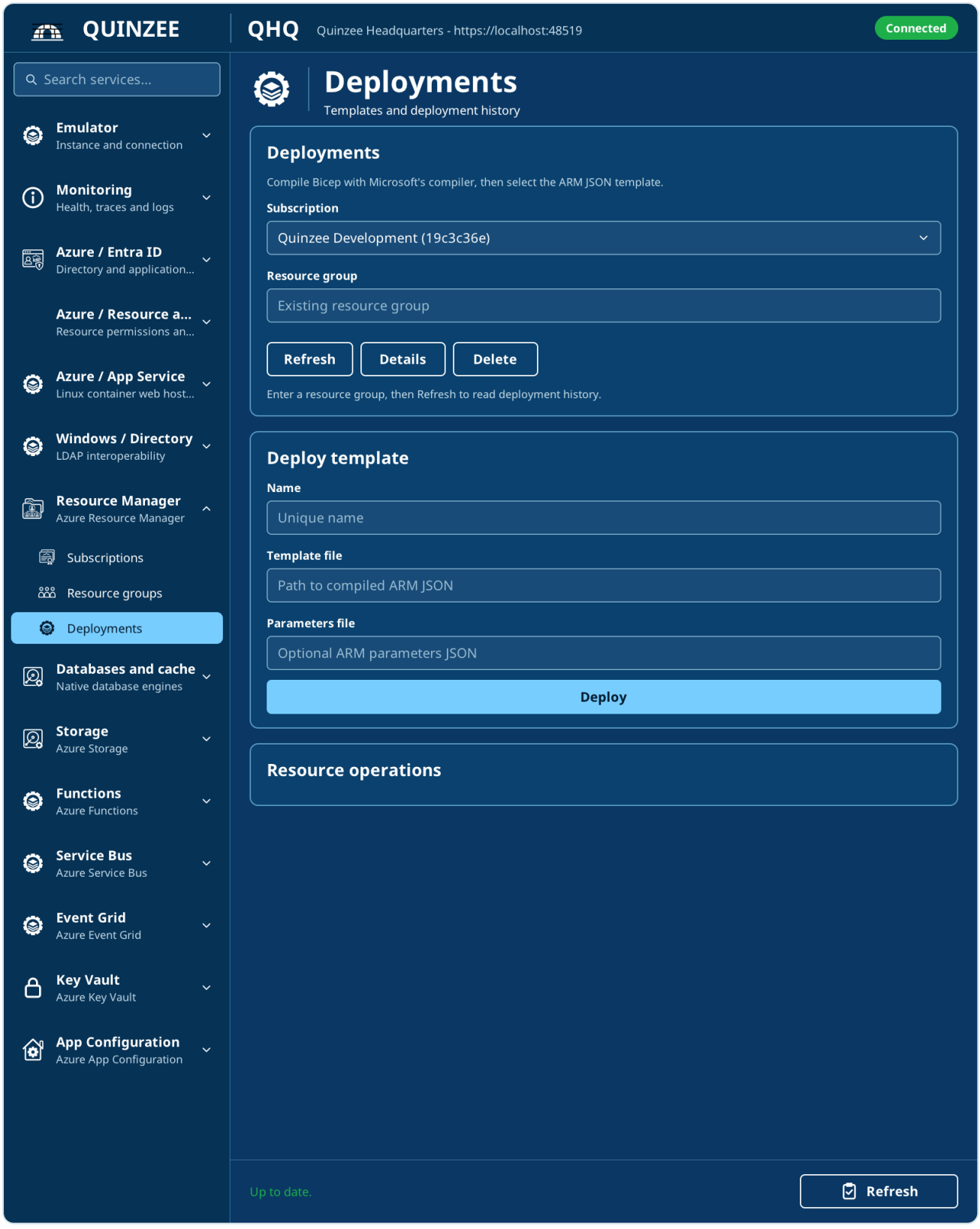
Open Resource Manager → Resource groups, choose the subscription, then Create. Enter a simple name such as `admin-training`, a supported location and optional string tags. Tags are public metadata, not a secret store. Note the resource ID. Most services require their group to exist before creation. Location is metadata used for compatible resource relationships and endpoints; it does not move data to an Azure region.

Step 2: compile infrastructure as code

Bicep is authoring source. ARM JSON is the deployment document. Compile the provided workshop files on the host:

```
mkdir -p .local/admin-training
python3 scripts/deps/bicep.py build examples/book-iac/storage.bicep \
  --outfile .local/admin-training/storage.json
```

In Resource Manager → Deployments, select the subscription and group. Supply a deployment name, compiled template path and optional private parameter-file path. The files are read by QHQ on its own workstation, then submitted over HTTPS. A path inside the cloud container is not automatically a path QHQ can read. Submit and refresh until the operation succeeds or fails; inspect resource operations and outputs rather than assuming that an accepted request is complete.



Step 3: supply parameters safely

A standard parameter document can contain `$schema`, `contentVersion` and a `parameters` object; each parameter contains a `value`. The native tutorial client also accepts the bare parameter mapping. Keep secret parameter files mode 0600 and out of source control. Bicep `@secure()`

and ARM `secureString` / `secureObject` prevent those values from appearing in deployment history. They do not encrypt an operator's local file.

Ordinary names and tags must not contain secrets. Secure parameters cannot be used as resource names or deployment outputs, including through variables. Provider GET responses still follow each provider's redaction rules. Treat returned connection strings as private even when the platform exposes them through an authorized operation.

Step 4: recover a failed deployment

1. Identify the first failed resource operation and the actual service error.
2. Check required modules, group/location, API version and helper health.
3. Correct the template/input or service prerequisite.
4. Redeploy incrementally with the same intended resource names.
5. Inspect application data and outputs after success.

A failed template is not automatically rolled back. Resources completed earlier remain. Active deployments interrupted by cloud restart are marked failed; their secure inputs are not persisted to resume later. Re-submit from your protected source and parameter files. Deleting completed deployment history preserves resources; deleting the group invokes every provider's cleanup and can destroy SQL volumes, queue messages, telemetry and secrets.

Supported deployment settings and fixed limits

Setting	Accepted behavior
Scope	Resource group only
Mode	Incremental only
Template	Inline ARM JSON, array-form resources
Inputs	string/secureString, object/secureObject, array, int, bool; supported bounds/defaults
Dependencies	`dependsOn`, supported resource references and parent relationships
API versions	Provider discovery reports the deployed subset; examples use verified versions
Request/resource limits	1 MiB template request, 128 resources, 16 queued deployments
Retention	256 deployment records, 1,024 operation records

Supported expressions include parameters/variables, resourceGroup/subscription, current-group resourceId, concat/format, case/string transformations, limited condition helpers, and reference to resources in the same deployment. Complete mode, loops, linked/nested templates, symbolic resource maps, resource conditions and arbitrary ARM expressions fail explicitly. Validation checks the deployment language; service-specific errors and missing runtimes can still occur later.

Related reading

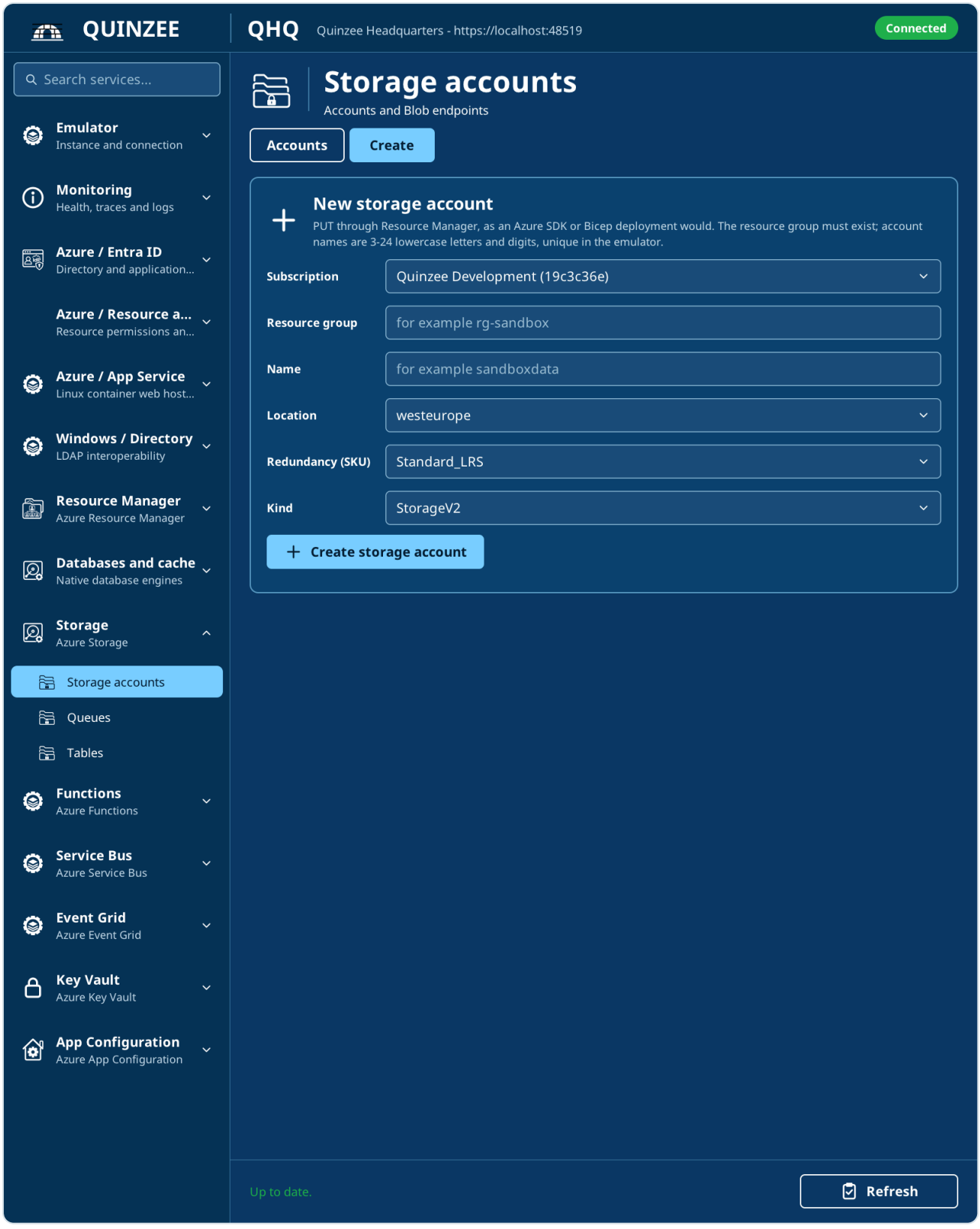
- The Book of Quinzee: Deploy your application's resources with Bicep
- The Book of Quinzee: Turn infrastructure into repeatable deployments
- Quinzee Internals: Deployments and native engines
- Scenario overview: Testing Bicep Deployments
- Scenario overview: Learning Infrastructure as Code with Bicep
- Scenario overview: Creating Reproducible Development Environments

8. Administer storage, secrets and configuration

These services are in-process modules. They do not need the external SQL engine supervisor merely to store data. Table Storage uses embedded SQLite; the shared LDAP directory's optional PostgreSQL is unrelated. Back up the complete cloud state consistently instead of trying to substitute an external database.

Step 1: provision a storage account

Open Storage → Storage accounts, select a group and use Create. Enter a globally unique local account name, location, kind and supported SKU. The developer IaC sample uses `StorageV2` and `Standard_LRS`. The provider exposes separate Blob, Queue and Table endpoints under that account. Provision containers/queues/tables through the service APIs or the corresponding QHQ controls; an account alone contains no application objects.



For Blob workloads, use private containers (`publicAccess: None` in ARM). Clients control content type, metadata, block IDs, range requests and supported SAS fields through the Blob API. SAS grants are separate from Entra resource roles; restrict their resource, permissions and expiry. Key regeneration changes which account-key signatures remain valid. Account deletion removes its data. There is no Azure Files, page/append Blob, lease or snapshot implementation.

Step 2: inspect queue work without accidentally consuming it

Open Storage → Queues. Select the account and queue. Create/delete queues, send synthetic messages, and use Peek to inspect pending work without changing visibility. A receive gives a message a visibility lease and pop receipt; deletion requires the current receipt. An expired lease makes work visible again, so consumers must tolerate duplicates. Application-level completion and a successful HTTP send are different events. Queue SAS is not implemented.

Queue metadata, message visibility timeout and time-to-live are API values. Do not confuse their duration with the server HTTP timeout. Updating a message changes its receipt; a stale receipt cannot settle it. Before deleting a queue, stop producers and confirm whether pending work is disposable.

Step 3: administer tables as typed data

Open Storage → Tables, select the account and table, then work with entities. Every entity has `PartitionKey` and `RowKey`. Use ETags for conditional updates so another writer's change is not silently replaced. Supported types include strings, booleans, numeric values, DateTime, GUID, binary and explicit Int64 encoding. The table API supports its bounded OData filters, paging and same- partition entity group transactions. An ARM table declaration does not create any entities. There is no Table SAS, ACL or service-properties implementation.

Treat SQLite database files and their WAL/SHM files as one live store. Copying only the main file while writers are active is not a consistent backup. The backup chapter's stopped-instance procedure avoids that mistake.

Step 4: operate Key Vault secrets

Open Key Vault → Vaults. Create a vault with name, location, tenant and supported SKU metadata. In Secrets, set a synthetic value, retrieve it through authorized access, then change it and observe a new version. Version metadata can include content type, tags, enabled state, not-before and expiry. Deleting a secret moves it into the soft-deleted workflow; recovery and purge are separate operations. **QHQ’s Delete selected performs soft delete followed immediately by purge.** Use the secret API/SDK directly when you need a recoverable deletion. Purge is irreversible. A vault resource's Azure-shaped RBAC/access-policy fields do not implement the full Azure Key Vault authorization model locally.

Quinzee stores secrets and versions privately but does not claim blanket at-rest encryption. Do not put real production credentials into a development screenshot or public deployment output. Cryptographic keys and certificate objects are not implemented by this module. The host's TLS certificate metadata is a separate platform function.

Step 5: operate App Configuration

Open App Configuration → Configuration stores, create a store and select Key-values. Enter the key, optional label, value and content type. Labels select separate variants; an absent label and a named label are not interchangeable. Use ETags for concurrent edits. Lock a value when it should become read-only, and deliberately unlock it before changing it. Revisions retain prior values according to the implemented service workflow. Deleting the store deletes its configuration history.

A Key Vault reference is a JSON setting with the expected content type and URI; it does not make the configuration service fetch or authorize that secret for your application. The application resolves it using its own secret client. Snapshots, replicas and full feature-management behavior are not implemented.

Configuration choices at these boundaries

Surface	Values an operator/application controls
Storage ARM	Account name/location/tags, supported kind/SKU; container privacy, queue metadata, table name
Blob data	Object name/content/type/metadata, staged blocks, conditions, supported SAS scope/permissions/expiry
Queue data	Queue name/metadata, message content, visibility and TTL, pop-receipt-based settlement
Table data	Table name, partition/row keys, typed properties, ETag conditions, bounded filters/paging
Vault ARM	Name/location/tags, tenant/SKU and retained Azure-shaped properties; do not assume policy enforcement from metadata
Secret data	Name/value, version attributes, content type/tags, enabled/not-before/expiry, delete/recover/purge
Configuration ARM	Name/location/tags and supported SKU/properties
Key-values	Key/label/value/content type/tags, ETag conditions, lock/unlock

Choose concrete values and plan for fixed limits

Storage account names use 3–24 lowercase letters/digits. Accepted `kind` values are StorageV2, Storage, BlobStorage, BlockBlobStorage and FileStorage; the name FileStorage does not enable Azure Files. Accepted SKUs are Standard_LRS, Standard_GRS, Standard_RAGRS, Standard_ZRS, Standard_GZRS, Standard_RAGZRS, Premium_LRS and Premium_ZRS. These describe metadata, not replicated disks or a performance guarantee.

`supportsHttpsTrafficOnly: false` is rejected. The accepted accessTier values are Hot (default) and Cool; they are metadata, not different local storage engines. Archive storage is unsupported. Service properties such as Blob retention/CORS are not tunable merely because a PUT is accepted. Container public access is refused.

Boundary	Value or fixed bound
Blob request	At most 256 MiB upload body; staged commit at most 1 GiB
Blob listing	maxresults at most 5,000
Blob SAS	Resource b/c, ordered r/w/d/l permissions, UTC start/expiry, HTTPS only; account-key signatures
Queue receive	1–32 messages; positive visibility up to seven days
Queue enqueue	Initial visibility may be zero; TTL defaults to seven days, positive seconds or -1 for no expiry
Queue payload	64 KiB message text including any Base64 expansion; 8 KiB metadata
Queue capacity	1,024 queues/account, 1,024 messages/queue, 16 MiB serialized queue file; listing at most 5,000
Table query	$top 1–1,000; at most 15 filter comparisons, nesting 32, filter text 16 KiB
Table entities	252 custom properties; key at most 1 KiB UTF-16 without slash/backslash/#/?/controls; string/binary value 64 KiB; entity 1 MiB
Table transactions	1–100 operations in one table/partition; 4 MiB batch, 2 MiB ordinary request
Table capacity	1,000 tables/account; 4 GiB total table data per emulator; continuation also after 5,000 examined rows or 16 MiB loaded data
Vault	3–24-character unique name; matching local tenantId; standard/premium SKU
Secret	1–127 letters/digits/hyphens, case-insensitive; value at most 25 KiB; 500 versions retained
Secret attributes	enabled is enforced; not-before and expiry are retained metadata and do not prevent retrieval
Configuration store	5–50 letters/digits/hyphen/underscore; free/developer/standard/premium SKU
Configuration authentication	disableLocalAuth rejects HMAC; otherwise two read-write and two read-only keys, each individually regenerable

Boundary	Value or fixed bound
Configuration data	Key plus value at most 10 KiB; last 1,000 revisions/store; lists are one page

App Configuration accepts keys containing `/` (URL-encode it), an optional label, content type and tags. Key/label filters support comma lists and trailing `*`; `\0` selects the null label. If-Match/If-None-Match enforce conditional changes. Read-only keys cannot mutate values; disabling local auth does not disable bearer access. Regenerating one key invalidates that key's old signatures. Vault access-policy and enableRbacAuthorization fields are retained but are not an enforced per-principal data-plane ACL. Storage data-plane RBAC is likewise not implemented. These token data planes retain bootstrap/administrator access; new scoped identities are refused until service-specific authorization exists. An ARM Reader assignment does not grant access to their contents. Shared keys and supported SAS have their own authorization rules. Some older providers retain additional Azure metadata; retention of a field is not proof that its operational behavior is emulated. Do not edit private JSON/SQLite files to invent an unsupported feature.

Related reading

- The Book of Quinzee: Store documents in Blob Storage
- The Book of Quinzee: Process work with Queue Storage
- Quinzee Internals: Module host and service state
- Quinzee Internals: Files, transactions and what a crash can leave behind

9. Keep events and messages moving

Event Grid delivers notifications. Service Bus retains work until a consumer settles it. Queue Storage is a third, simpler messaging service. Choose the service according to the application contract; their keys, locks, endpoints and failure modes are not interchangeable.

Step 1: provision an Event Grid route

Prepare the supplied Event Grid deployment on the host:

```
mkdir -p .local/admin-training
python3 scripts/deps/bicep.py build examples/book-iac/event-grid.bicep \
  --outfile .local/admin-training/event-grid.json
```

Use Resource Manager → Deployments to submit that JSON in your training group, as practiced in chapter 7. Defaults need no parameter file. It creates a storage account/queue, a storage system topic and a custom topic, each with a queue destination. Open Event Grid → Topics. Select the custom topic, inspect its subscriptions, and send a synthetic event using the page's test operation. Then inspect the StorageQueue destination. For a system topic, upload a Blob to its source account and observe the BlobCreated notification.

A successful Blob upload is not a delivery receipt for its event. The module uses a durable SQLite outbox and one serial delivery worker. Slow webhooks can delay other deliveries. Pending deliveries survive restart; when limits are exceeded or retries are exhausted, events are counted/logged as dropped. There is no dead-letter destination to recover those dropped events from.

Event Grid settings

Object	Supported configurable values
Custom topic	Name/location/tags, `inputSchema` EventGridSchema or CloudEventSchemaV1_0, `disableLocalAuth`
System topic	Storage account `source`, `topicType: Microsoft.Storage.StorageAccounts`; one topic per source
Destination	HTTPS WebHook URL, or StorageQueue resource ID and queue name

Object	Supported configurable values
Filter	includedEventTypes, subjectBeginsWith/subjectEndsWith, isSubjectCaseSensitive
Delivery schema	Supported Event Grid or CloudEvents 1.0 output conversion
Retry policy	1–30 attempts, 1–1440 minutes time-to-live
Topic credentials	Key1/key2 listing and regeneration, SAS expiry/scope, or supported bearer token

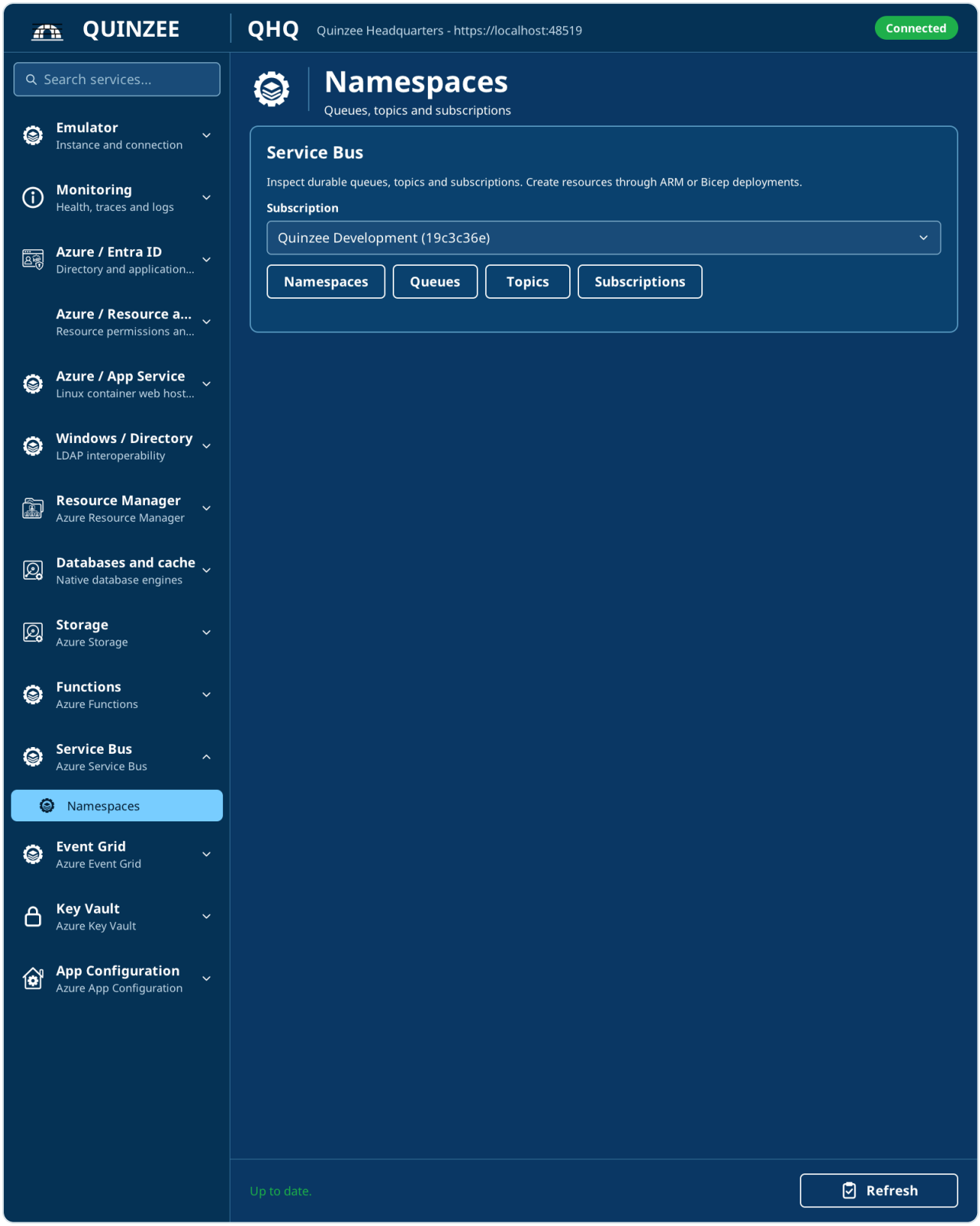
Webhook creation performs a validation handshake. Event Grid schema endpoints must return the supplied validation code; CloudEvents endpoints answer OPTIONS with WebHook-Allowed-Origin. Delivery uses verified HTTPS, no redirects and a 30-second deadline. A 2xx response acknowledges delivery. A handler that returns success before durably accepting work risks losing that work after a crash. Subscriptions can reach the network visible to the cloud, so manage destination URLs as privileged configuration. Advanced filters, Service Bus destinations, managed-identity delivery, namespaces and MQTT are unsupported.

Fixed bounds include 1 MiB publish bodies, 5,000 events per request and 10,000 pending deliveries. The retry sequence begins at 10 seconds, 30 seconds, one minute, five minutes, ten minutes, thirty minutes, then hourly. These are fixed implementation limits, not undocumented startup knobs.

Step 2: enable Service Bus transport

The standard Compose launch enables TLS AMQP on container port 5671. Native launches need `--service-bus-port 5671`; omission leaves HTTPS messaging available but disables AMQP. Use the same verified native certificate chain or the Azure SNI certificate for Azure namespace names. Clients use AMQP TCP over TLS, not plaintext AMQP or WebSockets.

Compile `examples/book-iac/service-bus.bicep` with the same command pattern, using `service-bus.json` as the output. Submit it as a separate deployment. It creates a Standard namespace, queue, topic and subscriptions. In QHQ's Service Bus page, choose the namespace, then Queues or Topics; select a topic before Subscriptions. The page inspects inventory. Create and change those resources through ARM/Bicep deployments. Namespace keys are not shown in its inventory.


QUINZEE
QHQ
Quinzee Headquarters - https://localhost:48519
Connected
Search services...
Emulator
Instance and connection
Monitoring
Health, traces and logs
Azure / Entra ID
Directory and application...
Azure / Resource a...
Resource permissions an...
Azure / App Service
Linux container web host...
Windows / Directory
LDAP interoperability
Resource Manager
Azure Resource Manager
Databases and cache
Native database engines
Storage
Azure Storage
Functions
Azure Functions
Service Bus
Azure Service Bus
Namespaces
Event Grid
Azure Event Grid
Key Vault
Azure Key Vault
App Configuration
Azure App Configuration
Namespaces
Queues, topics and subscriptions
Service Bus
Inspect durable queues, topics and subscriptions. Create resources through ARM or Bicep deployments.
Subscription
Quinzee Development (19c3c36e)
Namespaces
Queues
Topics
Subscriptions
Up to date.
Refresh

Service Bus settings

Object	Accepted values
Namespace	Standard SKU/tier, name/location/tags, disableLocalAuth, minimumTlsVersion `1.2`, publicNetworkAccess `Enabled`
Queue/topic/subscription	status: Active, Disabled, SendDisabled or ReceiveDisabled
Queue/subscription	lockDuration PT1M–PT5M; maxDeliveryCount 1–100, default 10
Message	Body and supported properties; default/maximum TTL 14 days, shorter per-message TTL accepted
SAS	Primary root rule, namespace/entity scope and expiry; disabled when disableLocalAuth is true
Bearer	Service Bus audience plus Data Sender, Receiver or Owner assignment for new identities

The primary RootManageSharedAccessKey `listKeys` operation returns a connection string to an authorized ARM caller. Keep it private. Custom authorization rules, secondary key rotation, sessions, duplicate detection, schedules, transactions, batch sends and partitioning are not implemented. An ARM Reader/Contributor role does not itself grant messaging access.

Step 3: diagnose locks and dead letters

PeekLock reserves a message with a current lock token. Complete removes it; Abandon makes it available again; renew extends a supported current lock. ReceiveAndDelete removes on receipt and cannot recover a consumer crash. When delivery attempts reach maxDeliveryCount, work moves to `$DeadLetterQueue`. Inspect and process it explicitly; deleting the namespace is not a repair. Lock state survives restart, so immediately after restart a message can still be locked rather than lost.

The broker commits SQLite WAL transactions with synchronous FULL before acknowledging successful mutations. Limits include 1,000 resources, 10,000 retained message copies, 256 MiB store, 256 KiB AMQP messages, 250 KiB REST bodies, 32 connections and 64 concurrent links. One topic message can consume multiple retained copies across subscriptions. Stop producers before capacity work; consumer catch-up and deliberate dead-letter handling are usually safer than removing state files.

Related reading

- The Book of Quinzee: Deliver work with Service Bus

- The Book of Quinzee: Run HTTP and Service Bus functions
- Quinzee Internals: Service Bus and Functions execution
- Quinzee Internals: Files, transactions and what a crash can leave behind
- Scenario overview: Experimenting with Cloud Architecture

10. Run PostgreSQL, MySQL and Valkey safely

The managed database modules implement ARM lifecycle and durable resource metadata. A separate operator-run supervisor creates the actual containers and volumes. Clients connect to those native engines directly. This division keeps the cloud process away from the Docker socket and allows the same cloud binary to run natively or in a container.

Step 1: prepare the approved engines

Start the cloud/network, then run on the Docker administrator's host:

```
make engine-images
make engines-run
```

The first command prepares the digest-pinned engine images. The second runs a foreground reconciliation loop using `.local/quinzee` and `triglav-http_quinzee` by default. Leave it running. When using another instance, set `ENGINE_DATA_DIR` and `ENGINE_NETWORK` together, or invoke the helper directly:

```
python3 infrastructure/engines/runtime.py \
  --data-dir /absolute/path/to/cloud-data \
  --state-dir /absolute/path/to/private-engine-state \
  --network your-existing-docker-network
```

Complete engine-helper options

Argument / environment	Meaning
`--data-dir`	Required cloud data root; reads/writes `engine-runtime` mailboxes
`--network`	Docker network name; omitted uses `quinzee-engines-<owner>`; a missing network is created
`--state-dir`	Private helper state outside the cloud data mount; omitted uses `<data-parent>/engine-state/<owner>`
`--once`	Perform one reconciliation pass and exit; not a persistent supervisor
`QUINZEE_OTLP_ENDPOINT`	Optional HTTPS telemetry collector origin

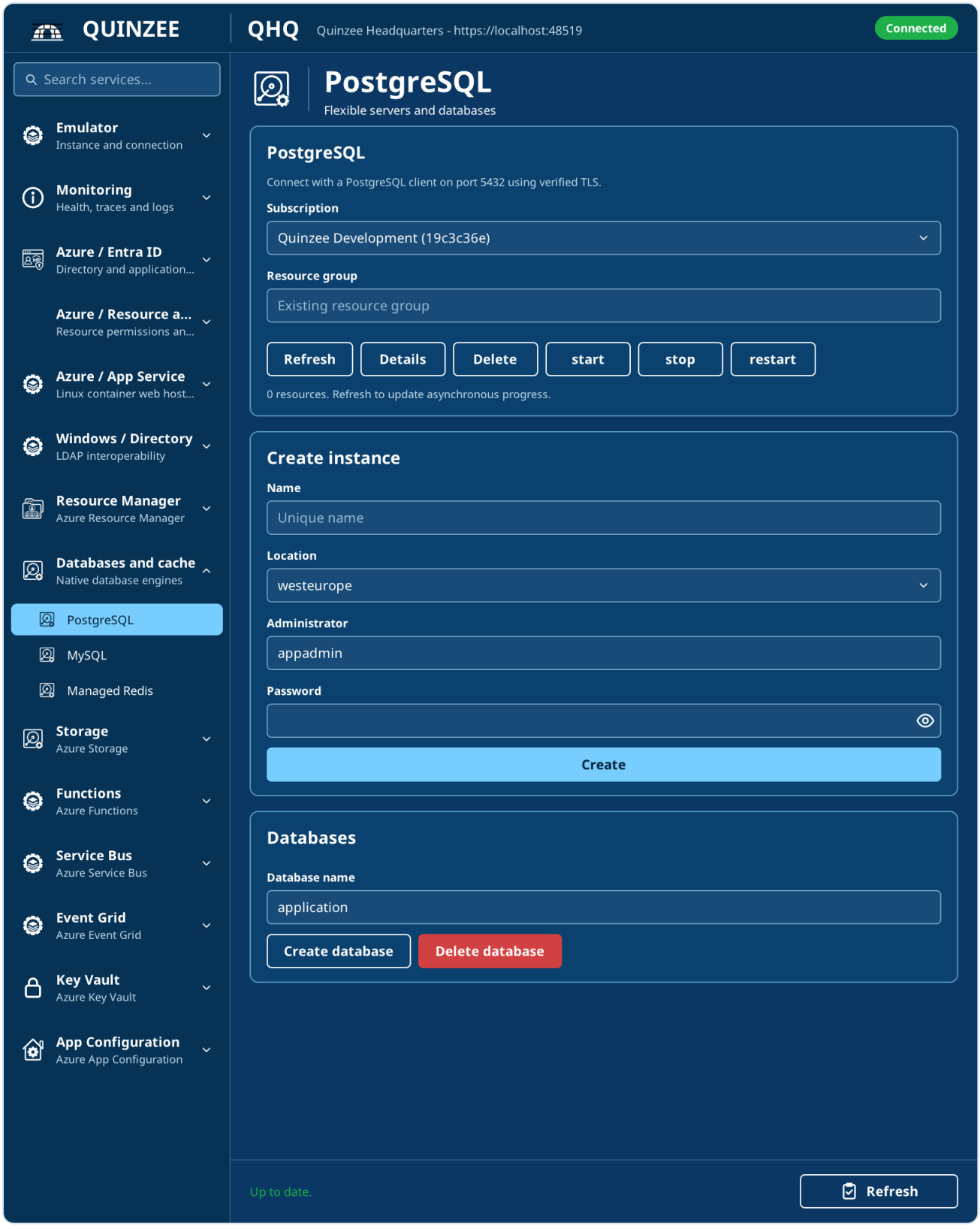
Argument / environment	Meaning
`QUINZEE_OTLP_CA`	Optional additional collector CA

`<owner>` is the first 16 hexadecimal characters of SHA-256 over the resolved `<data-dir>/engine-runtime` path. Moving the data directory therefore changes the default ownership identity and paths.

Record the resolved private state path when commissioning. It holds signing material, ownership and credential state needed for continuity. Do not put it inside the broadly shared cloud data mount. Each helper uses an ownership label and lock; unrelated containers must never be removed to repair this instance.

Step 2: create a server and database

In Databases and cache → PostgreSQL or MySQL, select subscription/group, enter the server name, location, administrator and password, and create it. Use PostgreSQL version 18 or MySQL 8.4. Passwords are 8–128 characters without newline or NUL; reserved administrator/system database names are rejected. Wait for Succeeded/Ready. Then create an `application` database under that server.



SQL clients use verified TLS on PostgreSQL 5432 or MySQL 3306 and the returned hostname. They need the engine CA, not merely the cloud's HTTPS CA. The helper publishes its public CA at `<cloud-data>/engine-runtime/ca.crt`; private signing material remains in helper state. It also runs a private bridge-bound CRL responder. Keep it reachable for clients that check certificate revocation.

The engine supervisor accepts only a fixed mailbox schema. A management request cannot choose arbitrary commands, images, mounts or Docker privileges. Engine containers use fixed limits: 1 GiB memory, two CPUs and 256 processes. SKU metadata does not resize those limits. The application role model does not implement managed-identity database login or engine-local user administration.

Step 3: create the cache

Open Managed Redis and create a resource using Balanced_B0, minimum TLS 1.2, then its `default` database with Encrypted/NoCluster/NoEviction. The real engine is Valkey. Read the access key through the authorized database `listKeys` operation and store it privately. The two returned key fields currently refer to one password; rotation is not implemented. Connect on TLS port 10000. The cache has an internal 512 MiB maxmemory/noeviction configuration in addition to the container limit. Plan for rejected writes when capacity is reached.

Step 4: pause, resume or remove

Use SQL Stop and wait for completion while the helper is running. Start and Restart are explicit lifecycle operations. Stopping the helper alone leaves engine containers running; it also stops its heartbeat and CRL responder. Module unload preserves volumes. SQL/cache Delete queues deliberate removal of the instance's labelled data volume and helper credentials. There is no built-in backup/restore service to reverse that deletion.

For an orderly host shutdown, quiesce applications, stop engines through their supported lifecycle, wait for reconciled status, then stop the helper and cloud. For recovery after a crash, restart the same helper with the same paths/network, check ownership and runtime status, then reconnect clients. Never launch a second helper against the same mailbox tree.

What can and cannot be changed

Surface	Supported values / boundary
SQL server	Name/location/tags, administratorLogin and private password at creation, supported version, lifecycle
SQL database	Child name, create/get/list/delete; reserved system names rejected
Cache	Supported SKU metadata, minimum TLS, default database protocol/clustering/eviction values
Capacity	Fixed local limits; at most 128 retained server records/provider and 64 databases/SQL server
Not implemented	HA, replicas, engine upgrades, backups API, private endpoints/firewall emulation, administrator rotation, Entra engine login

Use engine-native administration only within the application database's intended ownership. Changing helper-managed bootstrap credentials behind its back can break readiness and future reconciliation. Backup and restore drills belong to the next operations chapters, not to ad hoc edits of runtime JSON.

Related reading

- The Book of Quinzee: Use PostgreSQL, MySQL and Redis
- The Book of Quinzee: Provision databases and a cache with real engines
- Quinzee Internals: Deployments and native engines
- Quinzee Internals: From desired state to a real container
- Scenario overview: Testing Application Behaviour During Service Failures

11. Operate Functions and App Service workloads

A Functions app runs the real Azure Functions host and .NET isolated worker. An App Service web app runs its image's ordinary entrypoint. Both use one Microsoft.Web provider for plans, state, settings, identities and lifecycle, and one separately operated Docker helper. App Service is a distinct module that enables web-app admission and routing through the shared hosting capability.

Step 1: prepare and approve executable images

For an isolated test lab, the provided fixtures can be prepared as follows. The web fixture includes a token-returning identity probe for interoperability tests: never expose that fixture to untrusted users. For a shared installation, approve your own application image and its authentication policy instead.

```
make functions-example
make app-service-example
python3 infrastructure/functions/register.py \
  --image quinzee-functions-example:local \
  --registry .local/functions-helper/images.json
python3 infrastructure/functions/register.py \
  --image quinzee-app-service-example:local \
  --registry .local/functions-helper/images.json
make functions-run
```

Registration resolves a local image name to an immutable image ID. It authorizes that executable code. The helper does not pull an arbitrary image named in ARM. Restart the helper after changing the registry. The registry file is not a network container registry; it is the private allowlist used by this helper. `register.py` accepts exactly `--image` and `--registry` as required inputs.

Complete hosting-helper settings

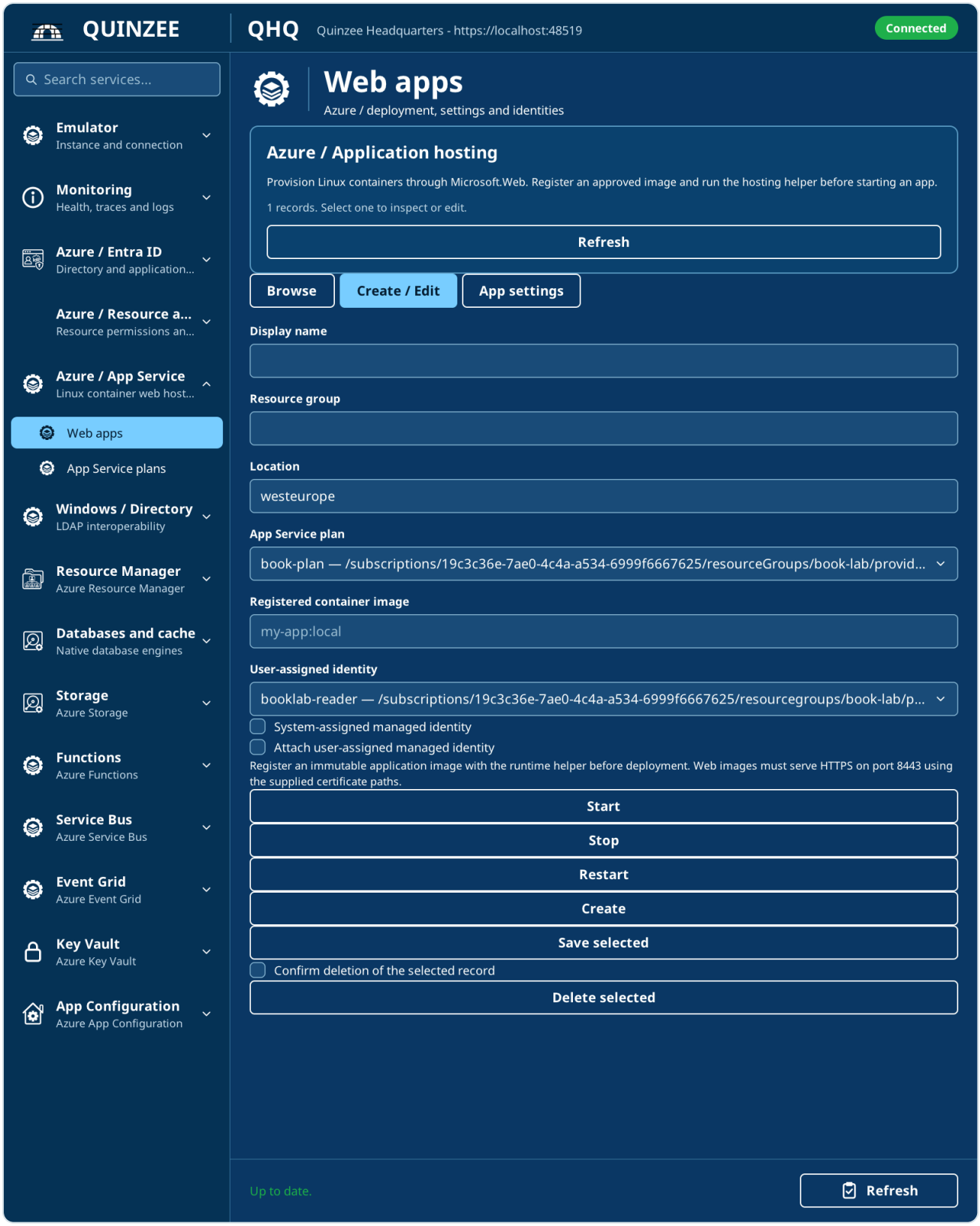
Option	Meaning
`--data-dir`	Required cloud data root; mailbox stays under `functions-runtime` for both hosting kinds
`--state-dir`	Required private state outside cloud data; preserve its CA and ownership files

Option	Meaning
`--registry`	Required approved image-name → immutable image-ID JSON map
`--network`	Required existing Docker network that can reach the cloud and dependent services
`--dns`	Optional DNS server IPv4 for containers
`--identity-endpoint`	HTTPS cloud origin; default `https://quinzee.cloud:11443`
`--add-host`	Repeatable administrator `name=IPv4` override
`--trust-ca`	Repeatable additional public CA file trusted inside hosted containers

`make functions-run` supplies ENGINE_DATA_DIR, ENGINE_NETWORK, FUNCTIONS_STATE_DIR/FUNCTIONS_REGISTRY and the standard DNS/CA values. For a custom lab, call the helper directly with the matching origin/network/trust. Its private state and data root must not overlap. Run it as a non-root Docker administrator. The cloud and application containers do not receive the Docker socket.

Step 2: create a plan and app in QHQ

1. Create the resource group.
2. Open Azure / App Service → App Service plans; enter scope, name, group and location. Create a Linux B1/Basic plan with capacity one.
3. Open Web apps, or Functions → Function apps. Enter the same group/location, select the plan and approved image name, and choose desired identity types.
4. Create and refresh until provisioning completes and runtime is Running.
5. Test the returned hostname from a client with the lab DNS and CA.



Functions native hosts use `<app>.functions.quinzee.cloud`; web apps use `<app>.appservice.quinzee.cloud`. Azure emulation also supports the corresponding azurewebsites.net route. Functions forwards only `/api/...`; its host enforces the function invocation key and blocks administrative paths. Web apps forward normal paths and must implement their own application authentication. A resource management token is not automatically an application login session.

Hosting resource and setting values

Surface	Accepted configuration
Plan	kind linux, reserved true, B1/Basic, capacity 1
Site	Immutable kind functionapp,linux,container or app,linux,container; same-group/location plan, httpsOnly true
Site config	linuxFxVersion `DOCKER
Identity	None, SystemAssigned, UserAssigned or both; at most eight supported user-assigned attachments
App setting names	1–128 letters/digits/underscore; protected runtime prefixes/names rejected
App setting values	Strings up to 8,192 bytes, no newline/NUL; at most 64 settings and 32 KiB aggregate
Workload lifecycle	Start/Stop/Restart; image/settings changes reconcile a new revision

In App settings, Load settings fetches private values internally and displays names. Set setting reads/merges/writes so unrelated settings survive. Remove setting deletes the selected name. Two administrators can still race because this API has no ETag merge contract; coordinate concurrent changes.

The helper reserves ASPNETCORE_, DOTNET_, COMPlus_, LD_, FUNCTIONS_, WEBSITE_, IDENTITY_ and MSI_ families and the exact names AzureWebJobsScriptRoot, AzureWebJobsSecretStorageType, SSL_CERT_FILE, SSL_CERT_DIR, HOME and PATH (case-insensitive). Do not replace the helper’s identity endpoint/header or TLS paths. For Service Bus managed identity, use the supported binding prefix such as `ServiceBusConnection__fullyQualifiedNamespace`, `__credential=managedidentity` and optional `__clientId`. Those are application binding settings, not helper credentials.

Step 3: make identity grants explicit

System-assigned identity belongs to the site and is removed with it. A user-assigned identity is an independent resource. Attaching either grants no resource access by itself. Assign the required Azure resource role or API app role to its principal ID. The helper injects a protected workload endpoint/header and the server checks the site's actual attachments. Ordinary clients cannot claim to be a managed identity merely by supplying its client ID.

QHQ preserves existing user-assigned attachments when adding a selected one. Clearing its attachment checkbox removes all user-assigned attachments; use ARM for precise multi-identity removal. Public inventory includes identity IDs but omits token headers, Function keys and setting values.

Step 4: diagnose readiness and lifecycle

Functions readiness is a verified TLS host-status request using a private master key. A web app must listen on TLS port 8443 and return 2xx/3xx at `/`. The ASP.NET example uses injected Kestrel certificate settings. An arbitrary HTTP-only image will not pass that contract. Separate name-constrained CAs protect upstream Functions and App Service connections.

No image means Stopped. Missing helper, unapproved image or failed readiness leaves Updating until failure; the timeout is 180 seconds. Only the current revision's status is accepted. A stale heartbeat stops gateway forwarding. Gateway bodies are bounded at 64 KiB and exchanges at ten seconds; there is no streaming/WebSocket forwarding.

Stop the app and wait for Stopped before stopping the helper when execution must cease. Helper shutdown or module unload alone can leave containers and queue triggers running. App Service unload blocks its gateway but retains containers/state. Unload it before Functions. Delete queues runtime removal; keep the helper running to complete that cleanup. Deleted app names remain reserved by tombstones, so use a fresh name for a new exercise.

Slots, autoscaling, Windows containers, ZIP/Kudu deployment, managed custom TLS, VNet integration and Easy Auth are not implemented. Functions' verified triggers are HTTP and Service Bus queues with .NET 10 isolated; an unrelated binding is not supported simply because an image can contain its package.

Related reading

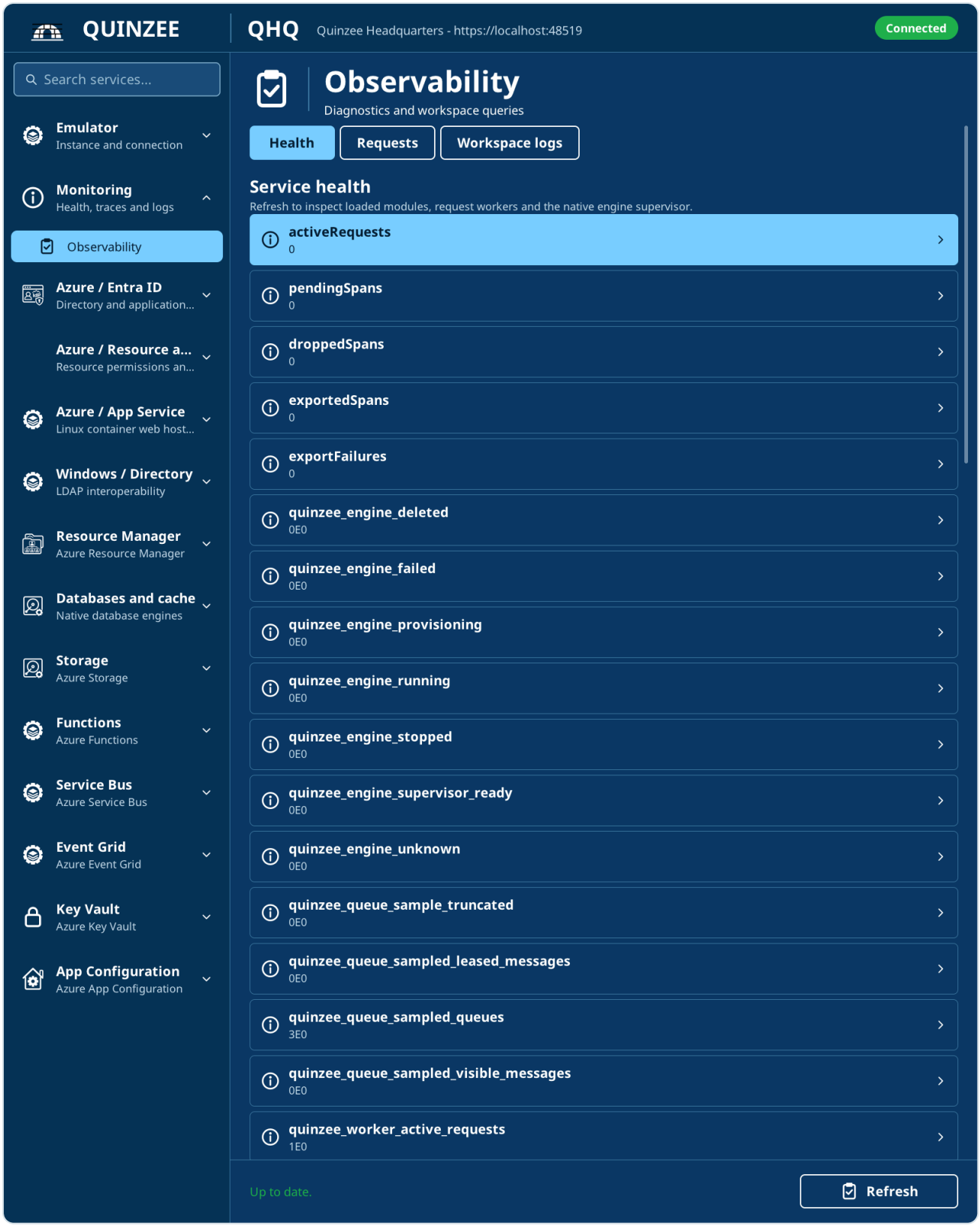
- The Book of Quinzee: Run HTTP and Service Bus functions
- The Book of Quinzee: Host a web application
- Quinzee Internals: Service Bus and Functions execution
- Quinzee Internals: From desired state to a real container
- Scenario overview: Developing an Azure Functions Application Locally
- Scenario overview: Using Quinzee as an Internal Server for a Small Office

12. Observe the platform and the application separately

Quinzee's platform diagnostics describe the emulator, workers and helpers. Application Insights and Log Analytics store application telemetry through Azure-compatible APIs. An optional external Grafana/OTLP stack observes the platform. These are related but separate pipelines: enabling one does not silently configure the others.

Step 1: check basic health

`GET /health` is public liveness. `GET /ready` says Identity and Resource Manager are loaded; it does not certify that every optional module or external database is available. In QHQ → Monitoring → Observability → Health, inspect workers, module state, engine status and heartbeat freshness. An old runtime status can remain visible after its helper stops; use the heartbeat as well as the last reconciliation result.



An engine heartbeat older than ten seconds is unavailable. Request handler metrics exclude socket transfer and executor wait, while transport metrics span the parsed HTTP exchange. Queue depth samples cover at most 32 queues with a bounded sampling budget and ten-second cache; a truncated sample is not a complete count of pending work. Use service inventory for exact operational investigation.

Step 2: correlate one request

Open Requests, reproduce one action, and filter by its trace ID. A trace ID links request/operation records; it is not an authentication token. Logs omit headers, queries, bodies and private credential material. Application log messages and custom properties remain application-controlled data, so producers must avoid secrets and personal data themselves. Redaction of known names is not a general sensitive-data detector.

Administrator APIs include `/quinzee/v1/observability` and `/quinzee/v1/observability/metrics`. The latter returns Prometheus text. Both require an administrator bearer token; do not publish them anonymously to make scraping convenient.

Step 3: provision and query application monitoring

Compile `examples/book-iac/monitoring.bicep` into `.local/admin-training/monitoring.json` with `python3 scripts/deps/bicep.py build` and `--outfile`, then submit it through QHQ Deployments in your training group. The default parameters create a Log Analytics workspace and a workspace-backed Application Insights component. The component needs the workspace capability, so unload the component module before the workspace module. Configure the application's official exporter with its returned connection string and supported token credential when local authentication is disabled. Use 100% sampling for this subset.

Open Workspace logs, enter the customer/workspace ID and a query:

```
AppTraces | order by TimeGenerated desc | take 20
```

Use AppRequests, AppDependencies, AppTraces, AppExceptions, AppEvents or AppMetrics. Supported stages are simple where comparisons, project, one-column order by, take/limit, count and summarize count(). Unsupported operators return errors; there is no full KQL engine behind the textbox.

Resource / API	Configurable values and bounds
Workspace	Name/location/tags, PerGB2018 SKU, retentionInDays 30–730
Component	Name/location/tags, web/other application type, immutable workspace binding, DisableLocalAuth
Ingestion	Uncompressed JSON object/array/NDJSON, supported envelope types, 1 MiB/256 envelopes, 8 KiB rows
Stored data	At most 64 workspaces/components; latest 4,096 rows/workspace in addition to time retention
Query	Query and optional timespan; 4,096 characters, 16 stages, at most 1,000 result rows; QHQ displays up to 100

Resource / API	Configurable values and bounds
ARM metrics	Supported request/dependency count/failed/duration; Count/Total/Average/Minimum/Maximum; PT1M/PT5M/PT1H/FULL; at most ten names, 30 days and 1,440 buckets

The 4,096-row cap can evict data much sooner than the configured days. Do not promise a month of audit history based only on retentionInDays. Live metrics, profiling, alerts, diagnostic settings, gzip/sample ingestion and arbitrary cross-workspace queries are unsupported.

Step 4: enable external platform telemetry

The simplest complete local stack is `make application-observability-up`. It prepares a separate application environment, certificates, private telemetry credentials and the Grafana/Collector/Prometheus/Loki/Tempo services. Record the URLs and paths it reports. For a separately managed cloud, use the normal TLS arguments plus `--otlp-endpoint https://<collector-origin>` and `--otlp-ca <collector-ca-file>`. The engine helper has matching QUINZEE_OTLP_ENDPOINT/QUINZEE_OTLP_CA environment values.

The stack Compose settings include QUINZEE_UID/GID, QUINZEE_OBSERVABILITY_DIR (private persisted data and TLS/auth files), QUINZEE_GRAFANA_PORT (default 13001) and QUINZEE_OTLP_PORT (14318). Its configuration files are mounted from `infrastructure/observability`: collector.yaml routes telemetry, prometheus.yaml sets collection, loki.yaml and tempo.yaml control stores, grafana.ini and the provisioning directory set dashboard/datasource behavior. Review those explicit files when operating the optional third-party stack; they are not quinzee.cloud JSON members. Keep the healthcheck and authentication/trust mounts intact.

Export is bounded and best effort. Collector failure increments export/drop counters instead of blocking application requests. Check those counters before assuming that an empty dashboard proves no requests occurred. Stop/backup this stack separately from the cloud's own monitoring resource files.

Related reading

- The Book of Quinzee: Observe your application
- The Book of Quinzee: Troubleshoot a connection or workflow
- Quinzee Internals: HTTP library and asynchronous execution
- Quinzee Internals: Connection and streaming lifecycle
- Scenario overview: Debugging Cloud Applications Locally
- Scenario overview: AI-Assisted Cloud Debugging

13. Back up, recover and rehearse failure

A backup is useful only if the associated identities, configuration and service data can be restored together. Quinzee has several persistence boundaries. A copy of one convenient JSON file is not a complete environment backup, and a copy of a live SQLite main file may omit committed WAL content.

Step 1: inventory what must survive

Boundary	Preserve
Cloud instance	Complete data directory, bootstrap credentials, tenant/signing identity, module state, directory destination marker, runtime mailboxes/tombstones
PostgreSQL directory	Consistent database backup plus the corresponding cloud identity/marker and private connection configuration
Application SQL/cache	Engine-native consistent backups or stopped labelled volumes; record engine versions and helper ownership
Engine helper	Private state, CA/signing/CRL metadata, credentials and ownership mapping
Hosting helper	Private state, constrained CAs, registry and available immutable application images
TLS/DNS startup	Configuration files, service certificates/keys, public trust distribution and private CA material under operator control
External observability	Its separate persistent data, configuration, private auth/TLS files and retention choices

Decide the recovery point and downtime you can tolerate. This local platform has no built-in distributed backup coordinator, HA failover or point-in-time restore API. Use a maintenance window for a coordinated stopped-state snapshot. Application databases can use their normal engine backup tools, but consistency across database rows, blobs and queued work is the application's responsibility.

Step 2: take a consistent stopped-state backup

1. Announce maintenance and stop producers, consumers and interactive changes.
2. Stop hosted apps through their lifecycle APIs while the hosting helper is running. Wait for Stopped so queue triggers cannot keep mutating state.
3. Take engine-native database backups or stop the managed engines cleanly. Stop the engine supervisor after reconciliation; account for its CRL service.
4. Stop the LDAP frontend and cloud. Stop directory PostgreSQL cleanly if taking a cold filesystem backup instead of a PostgreSQL-native backup.
5. Snapshot all listed state/configuration boundaries at that same maintenance point. Keep owner/mode metadata and encrypt/protect the backup as appropriate.
6. Record checksums, image IDs, engine versions, instance paths and the date.
7. Restart dependencies first, then cloud/helpers/apps, and verify a small create/read/update workflow before reopening access.

For a cloud-data archive only, after confirming the cloud and writers are stopped:

```
umask 077
mkdir -p .local/backups
tar --acls --xattrs -cpf .local/backups/cloud-data.tar \
  -C .local quinzee
sha256sum .local/backups/cloud-data.tar \
  > .local/backups/cloud-data.tar.sha256
```

That archive is one component, not the complete backup when PostgreSQL or external engine volumes are in use. It contains credentials and application data. Never attach it to a public issue or store it beside public web assets. Choose a new backup filename for each retained recovery point; the example filename is deliberately simple and would be overwritten on another run.

Step 3: restore into an isolated drill

Keep the original installation stopped while restoring its identity into another location. Running two copies of one tenant against the same PostgreSQL directory violates the writer lease. Restore the matching PostgreSQL snapshot, destination configuration/marker and cloud data as a set. Preserve helper ownership paths where possible; changing paths changes some helper ownership hashes. Do not manually rewrite ownership labels to adopt unknown containers.

Start the database dependency, cloud and helpers with an isolated network and unpublished/alternate host ports. Verify tenant ID, a known application identity, one shared group, one object per important service, a queued message and an application database row.

Confirm private file permissions. For hosting, confirm the approved immutable image IDs still exist. Restart once more and repeat the checks. A successful extraction command alone is not a recovery test.

Restore procedures for external engines should use their own consistent dump or cold-volume method for the same version. Engine bootstrap/password/CA state must match the restored ownership. Quinzee does not currently provide an in-place resource import or volume adoption API. If a clean-room restore requires a new resource identity, plan an application-level data migration rather than editing private records to impersonate the old instance.

Step 4: respond to common recovery events

Directory database outage: stop administrative writes, recover PostgreSQL, then restart the cloud to reacquire its writer connection. Never remove the migration marker to revive stale JSON accounts.

Helper crash: restart the same helper with the same private state, mailbox root and network. Check current revision and heartbeat before restarting applications. Some containers may have continued running during the outage.

Interrupted deployment: inspect failed history, repair dependencies and redeploy from protected source/parameters. Completed resources can remain.

Lost TLS key or planned CA renewal: stop the affected endpoint, replace material through the certificate workflow, distribute the new public trust and restart. A fresh CA changes trust; it is not a transparent repair.

Accidental Delete: stop further cleanup/writes and assess backups. Resource and group deletion can remove data/volumes. Secret soft-delete recovery applies only to the supported secret workflow; it is not a universal undelete service.

Related reading

- The Book of Quinzee: Stop, restart and clean up safely
- Quinzee Internals: Files, transactions and what a crash can leave behind
- Scenario overview: Testing Backup and Recovery

14. Diagnose failures from the outside inward

Start with a single failing operation and identify its boundary. Repeatedly changing multiple settings destroys evidence. Record the origin, timestamp, resource ID and trace ID, but omit passwords, tokens and private response bodies.

Step 1: use the symptom to choose the next check

Symptom	Check first	Meaning and next action
Connection refused	Process, bind IP, host/container port mapping	Start the intended listener; verify you are in the correct lab
TLS chain failure	Matching public CA bundle	Trust the intended issuer in this client; keep verification enabled
TLS hostname failure	URL/DNS/SAN agreement	Use the certificate's name or issue the correct certificate
401	Credential, tenant, audience, expiry	Obtain the correct service token; an ARM token is not a Graph token
403	Principal state and scoped assignment	Check app roles versus Azure RBAC and the resource scope
404	Host routing, path, API version/resource existence	Do not assume the same path works on every service hostname
409 unloading	RequiredBy list	Unload dependants first; stop external work separately
409 creating	Existing name, tombstone, duplicate assignment	Inspect the actual ID; use a new intended name only when appropriate
Updating then Failed	Helper heartbeat, image approval, engine/readiness error	Fix the external dependency and redeploy
LDAP bind fails	TLS, DN, account enabled/password/throttle	Use the generated DN and shared account; do not reset a second LDAP password store
Directory fails after DB recovery	Writer connection/marker	Restart cloud with the original destination configuration
Empty telemetry	Workspace/component, token audience, exporter/drop counters	Distinguish missing events from unsupported query or eviction

Symptom	Check first	Meaning and next action
Queue work repeats	Lock/visibility expiry, missing settlement	Make processing idempotent and settle with the current token

Step 2: check process health without exposing state

Use `/health`, `/ready`, QHQ status and the operator's service logs. In the standard environment, `docker compose logs quinzee-cloud quinzee-dns` shows the frontends. Helper output belongs to its own terminal/service log. A useful incident report includes the failed API/resource and a redacted trace excerpt, not credentials.json, a private settings dump or a full state archive.

Step 3: understand the implementation details that affect operations

HTTP handlers run on a bounded worker executor. Queueing can contribute latency before a handler runs, so inspect transport and executor metrics as well as service duration. Large bodies, slow clients and over-capacity work are bounded; there is no supported startup knob for every library-level limit.

Durable stores commit mutations before reporting success where their contract requires it. SQLite-backed tables, broker messages and Event Grid delivery rows need their full journal state for backup. A module unload removes routes/workers but retains durable state. An external runtime status is accepted only for its current generated revision; stale status cannot resurrect a deleted or replaced app. Tombstones prevent helpers from recreating deleted resources after restart.

The directory's one-writer lease and no-fallback marker protect permission consistency. Failure to authenticate during a database outage is intentional. Do not turn a fail-closed security boundary into stale authorization by deleting its state marker or copying an old directory.json over current records.

The native engine and hosting helpers validate fixed request schemas and approved images. Giving the cloud a Docker socket bypasses that design and is not an operational fix. Managed identity headers exist only at the protected workload boundary; copying one into a developer's terminal is not the supported login flow.

Step 4: separate test environment restrictions from product defects

`make test` runs debug checks; release and ASan profiles check other build modes. ThreadSanitizer can fail before application code because its runtime requests ADDR_NO_RANDOMIZE and the execution environment denies it. Record that as an

uncompleted sanitizer run. Do not claim it passed and do not weaken ASLR or container controls as a routine troubleshooting shortcut. The repository's TSan exception requires explicit approval.

Step 5: collect a reproducible report

State the build revision, OS/architecture, selected modules, relevant public configuration, operation sequence and expected/actual behavior. Reproduce with synthetic data in a disposable environment. For a new client library version, re-run its compatibility exercise; a compiling SDK call can still use an unsupported API shape. Preserve logs privately and share only the minimal redacted information needed to reproduce the failure.

Related reading

- The Book of Quinzee: Deliver work with Service Bus
- The Book of Quinzee: Service compatibility and limits
- Quinzee Internals: Testing and verification
- Quinzee Internals: The compatibility application
- Scenario overview: Integration Testing Across Multiple Azure Services
- Scenario overview: Running Azure Integration Tests in CI/CD
- Scenario overview: Debugging Cloud Applications Locally

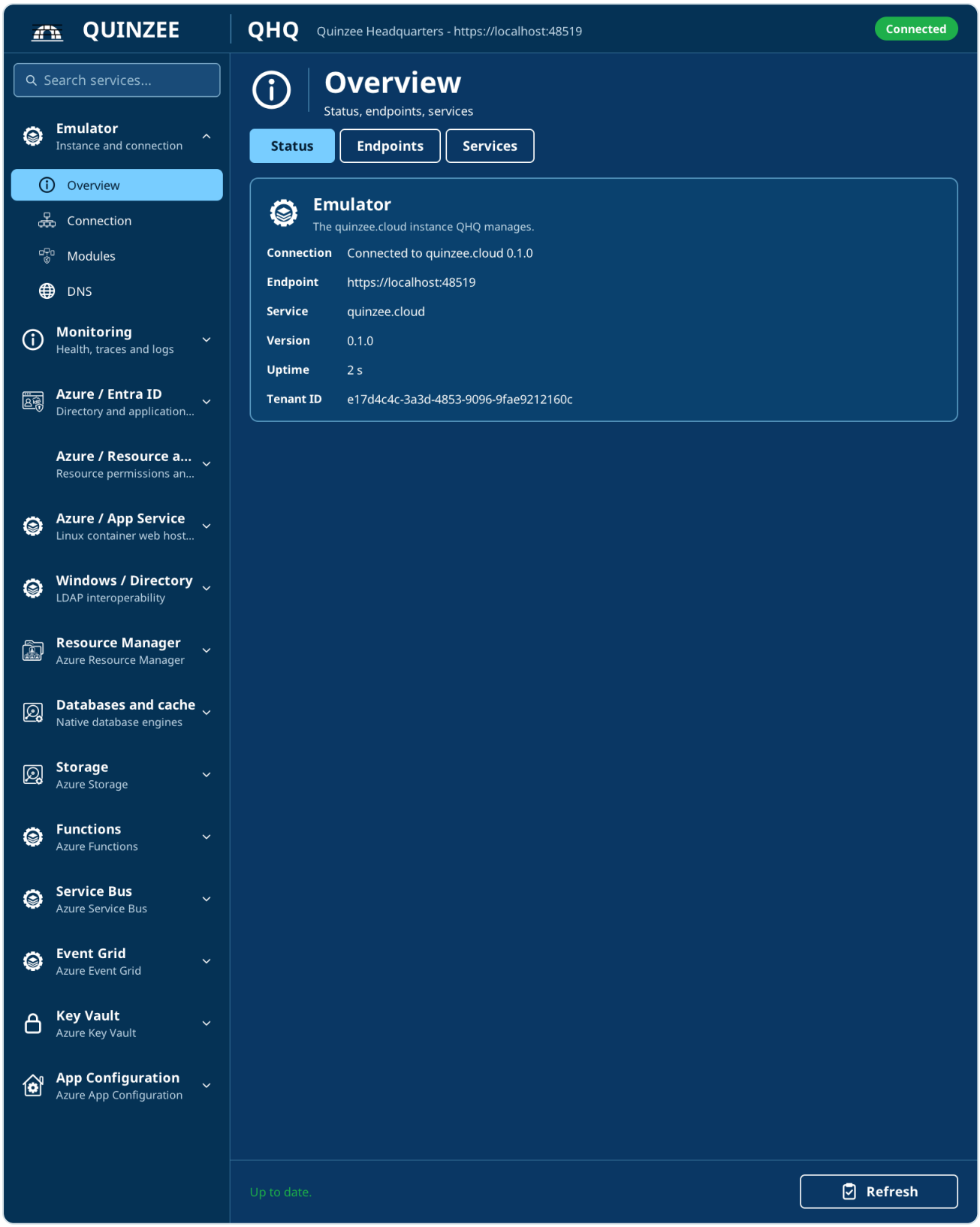
15. Find every control in QHQ

This chapter is a visual tour of every QHQ page and its implemented tabs. The captures come from one disposable HTTPS cloud with synthetic users, groups, registrations and deployed resources. No secret values were requested in the UI. The directory uses JSON mode here, so the LDAP page correctly reports an unconfigured listener. Native engine pages show creation controls without running engine containers. Sample hosted sites are intentionally Stopped with no image.

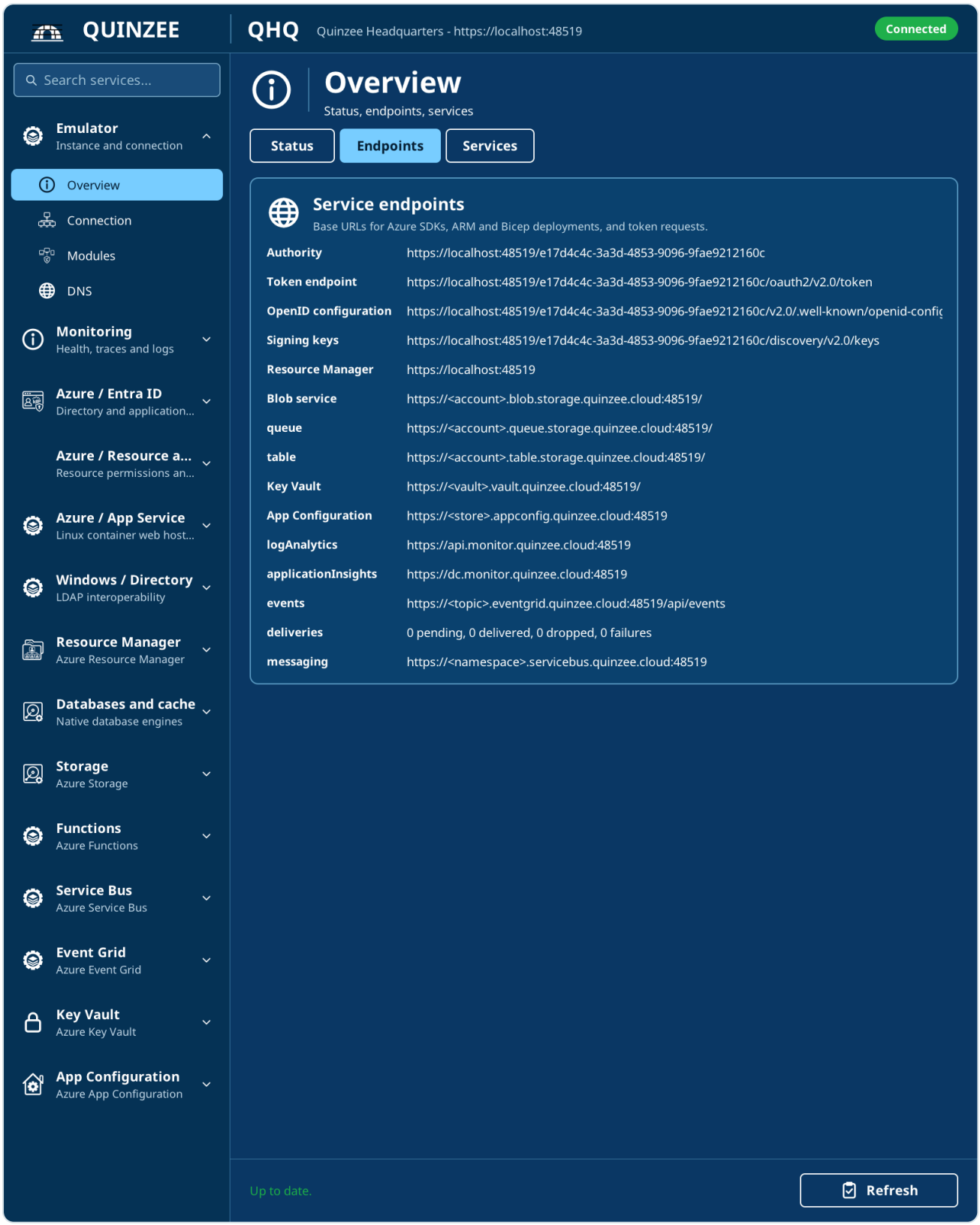
The tall capture window reveals long forms. On a smaller display, scroll to reach the same controls. Dynamic selections can reveal additional result lists. A screenshot is orientation, not proof that an external helper is running. Use the preceding service procedure and live status before making changes.

Overview

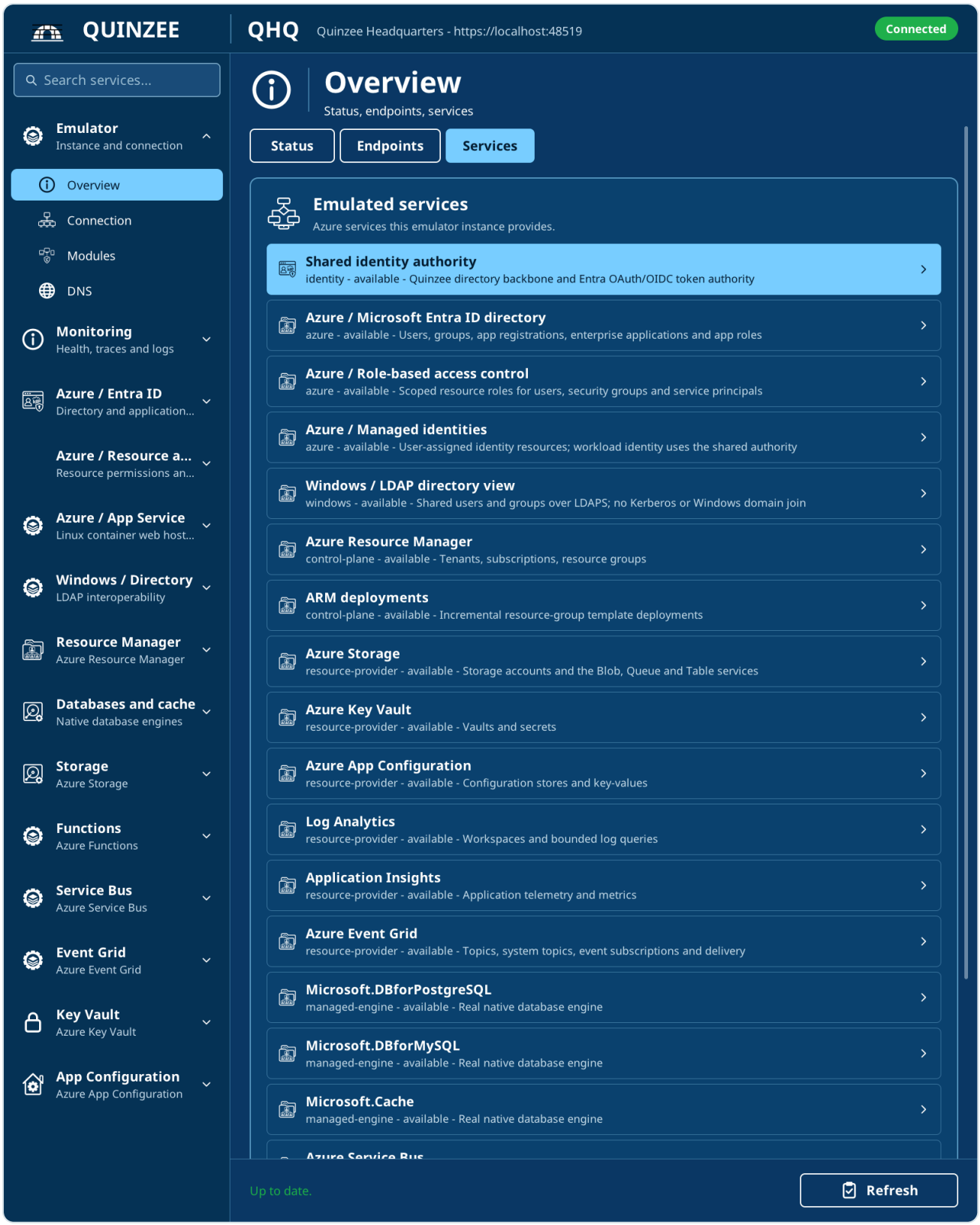
Status



Endpoints

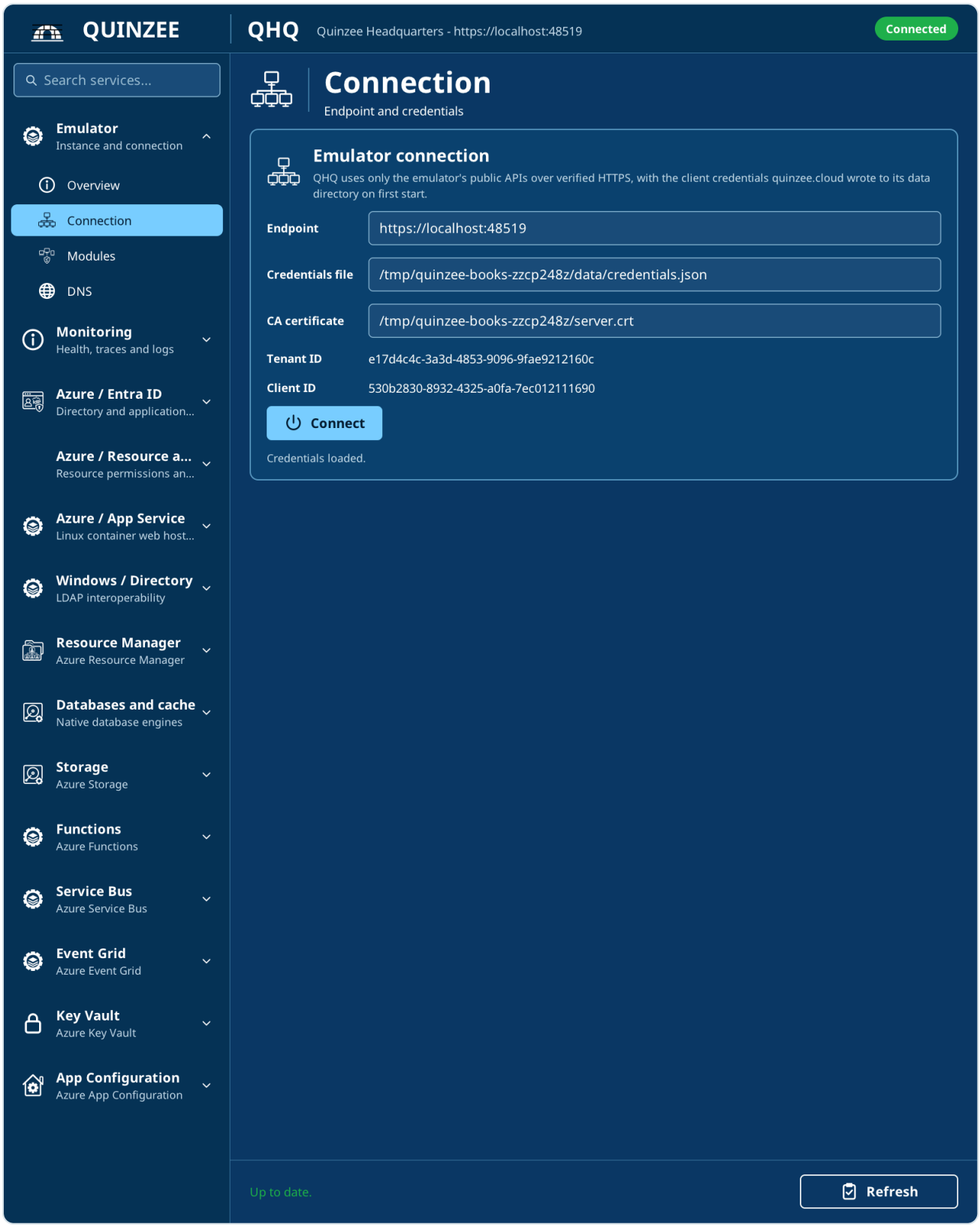

QUINZEE
QHQ
Quinzee Headquarters - https://localhost:48519
Connected
Search services...
Emulator
Instance and connection
Overview
Connection
Modules
DNS
Monitoring
Health, traces and logs
Azure / Entra ID
Directory and application...
Azure / Resource a...
Resource permissions an...
Azure / App Service
Linux container web host...
Windows / Directory
LDAP interoperability
Resource Manager
Azure Resource Manager
Databases and cache
Native database engines
Storage
Azure Storage
Functions
Azure Functions
Service Bus
Azure Service Bus
Event Grid
Azure Event Grid
Key Vault
Azure Key Vault
App Configuration
Azure App Configuration
Overview
Status, endpoints, services
Status
Endpoints
Services
Service endpoints
Base URLs for Azure SDKs, ARM and Bicep deployments, and token requests.
Authority https://localhost:48519/e17d4c4c-3a3d-4853-9096-9fae9212160c
Token endpoint https://localhost:48519/e17d4c4c-3a3d-4853-9096-9fae9212160c/oauth2/v2.0/token
OpenID configuration https://localhost:48519/e17d4c4c-3a3d-4853-9096-9fae9212160c/v2.0/.well-known/openid-confi
Signing keys https://localhost:48519/e17d4c4c-3a3d-4853-9096-9fae9212160c/discovery/v2.0/keys
Resource Manager https://localhost:48519
Blob service https://<account>.blob.storage.quinzee.cloud:48519/
queue https://<account>.queue.storage.quinzee.cloud:48519/
table https://<account>.table.storage.quinzee.cloud:48519/
Key Vault https://<vault>.vault.quinzee.cloud:48519/
App Configuration https://<store>.appconfig.quinzee.cloud:48519
logAnalytics https://api.monitor.quinzee.cloud:48519
applicationInsights https://dc.monitor.quinzee.cloud:48519
events https://<topic>.eventgrid.quinzee.cloud:48519/api/events
deliveries 0 pending, 0 delivered, 0 dropped, 0 failures
messaging https://<namespace>.servicebus.quinzee.cloud:48519
Up to date.
Refresh

Services


QUINZEE
QHQ
Quinzee Headquarters - https://localhost:48519
Connected
Search services...
Emulator
Instance and connection
Overview
Connection
Modules
DNS
Monitoring
Health, traces and logs
Azure / Entra ID
Directory and application...
Azure / Resource a...
Resource permissions an...
Azure / App Service
Linux container web host...
Windows / Directory
LDAP interoperability
Resource Manager
Azure Resource Manager
Databases and cache
Native database engines
Storage
Azure Storage
Functions
Azure Functions
Service Bus
Azure Service Bus
Event Grid
Azure Event Grid
Key Vault
Azure Key Vault
App Configuration
Azure App Configuration
Overview
Status, endpoints, services
Status
Endpoints
Services
Emulated services
Azure services this emulator instance provides.
Shared identity authority
identity - available - Quinzee directory backbone and Entra OAuth/OIDC token authority
Azure / Microsoft Entra ID directory
azure - available - Users, groups, app registrations, enterprise applications and app roles
Azure / Role-based access control
azure - available - Scoped resource roles for users, security groups and service principals
Azure / Managed identities
azure - available - User-assigned identity resources; workload identity uses the shared authority
Windows / LDAP directory view
windows - available - Shared users and groups over LDAPS; no Kerberos or Windows domain join
Azure Resource Manager
control-plane - available - Tenants, subscriptions, resource groups
ARM deployments
control-plane - available - Incremental resource-group template deployments
Azure Storage
resource-provider - available - Storage accounts and the Blob, Queue and Table services
Azure Key Vault
resource-provider - available - Vaults and secrets
Azure App Configuration
resource-provider - available - Configuration stores and key-values
Log Analytics
resource-provider - available - Workspaces and bounded log queries
Application Insights
resource-provider - available - Application telemetry and metrics
Azure Event Grid
resource-provider - available - Topics, system topics, event subscriptions and delivery
Microsoft.DBforPostgreSQL
managed-engine - available - Real native database engine
Microsoft.DBforMySQL
managed-engine - available - Real native database engine
Microsoft.Cache
managed-engine - available - Real native database engine
Up to date.
Refresh

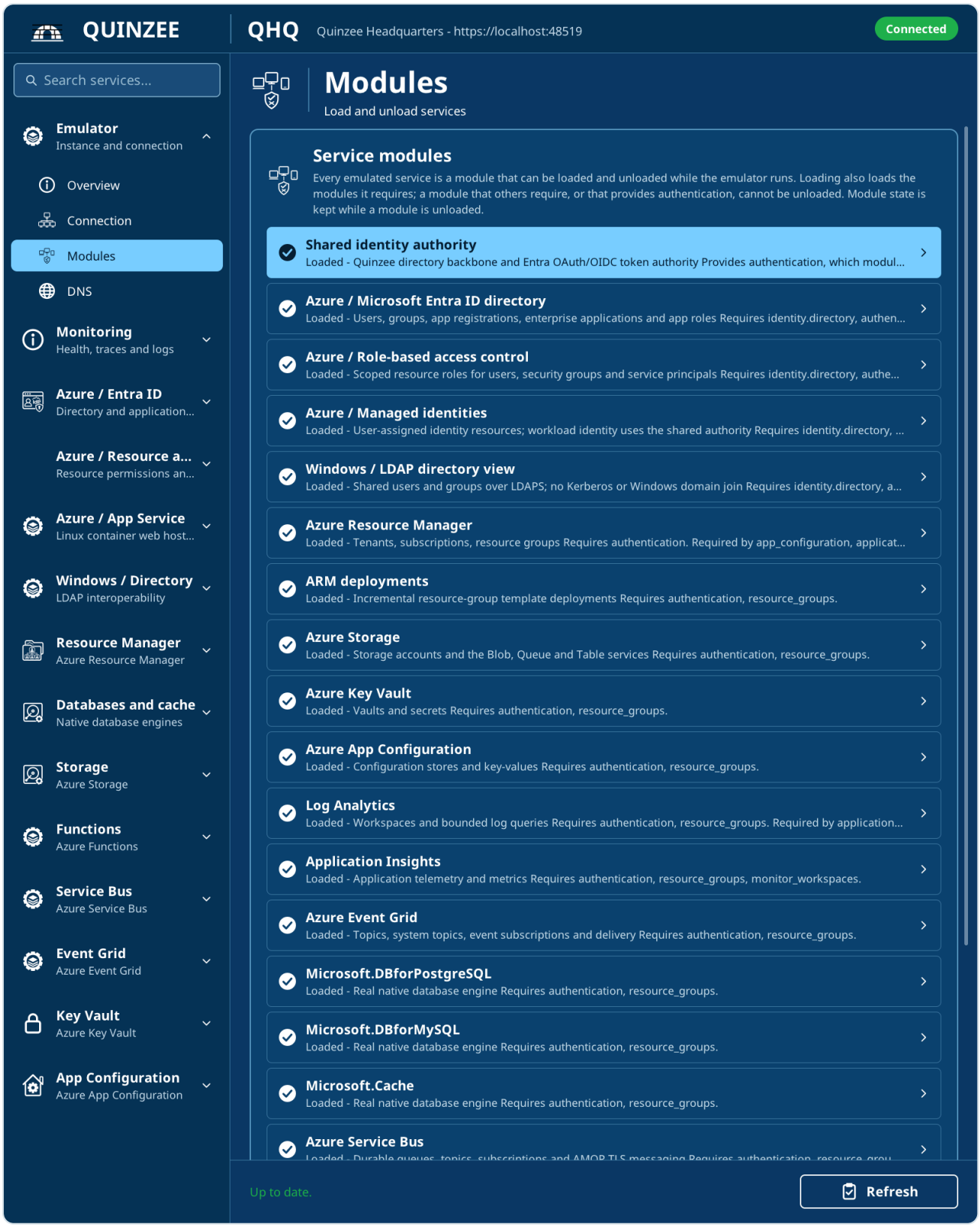
Connection

Connection



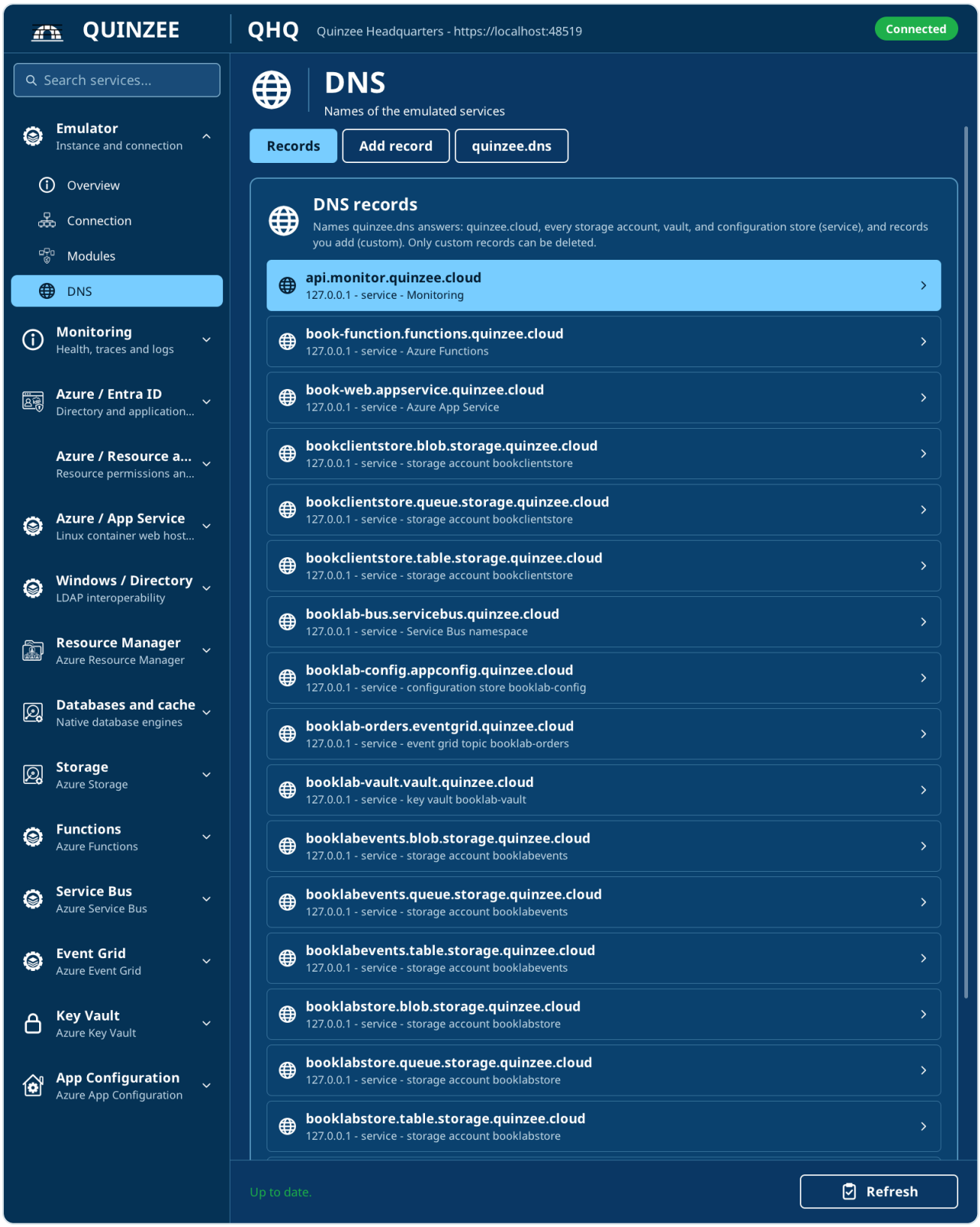
Modules

Modules

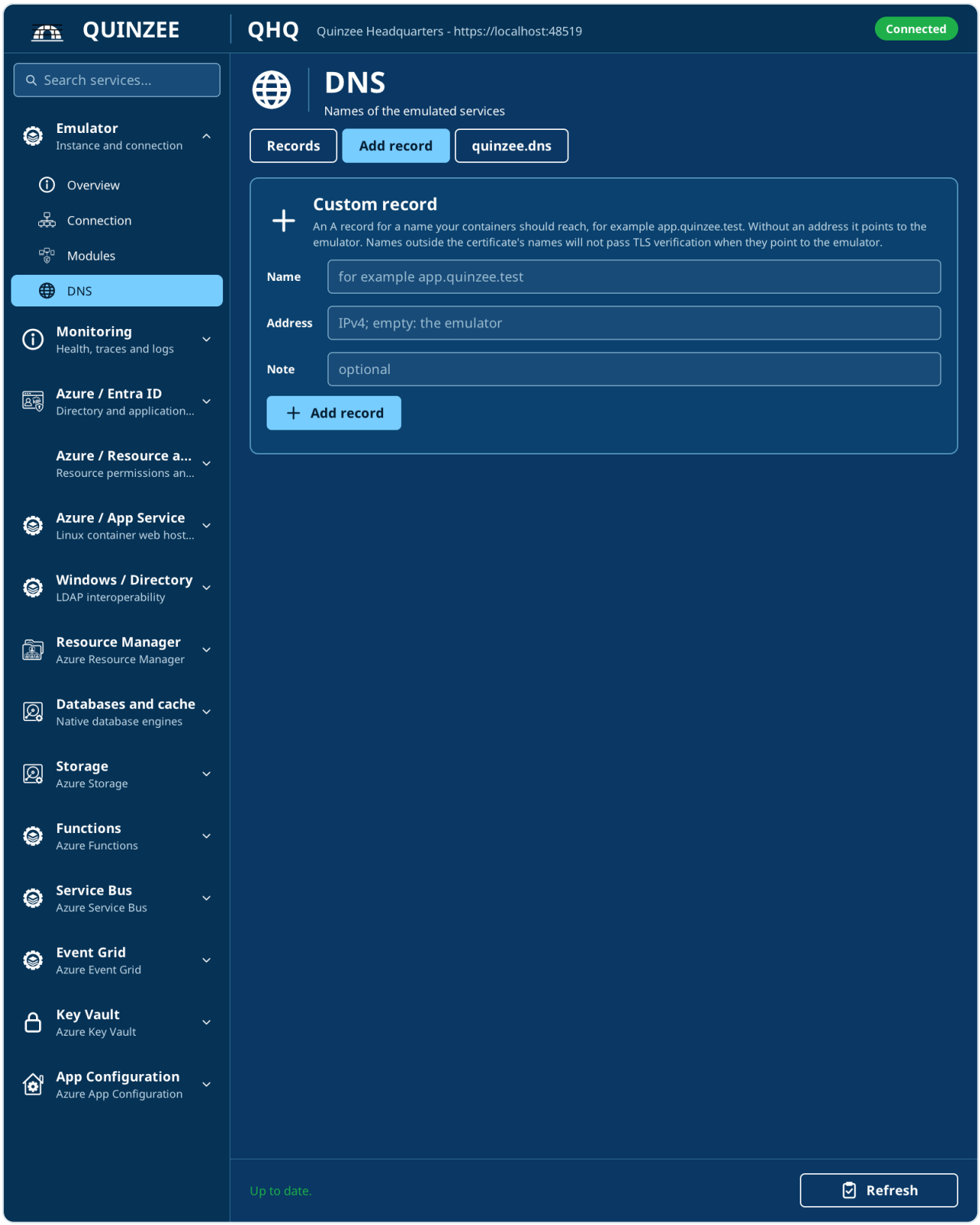


DNS

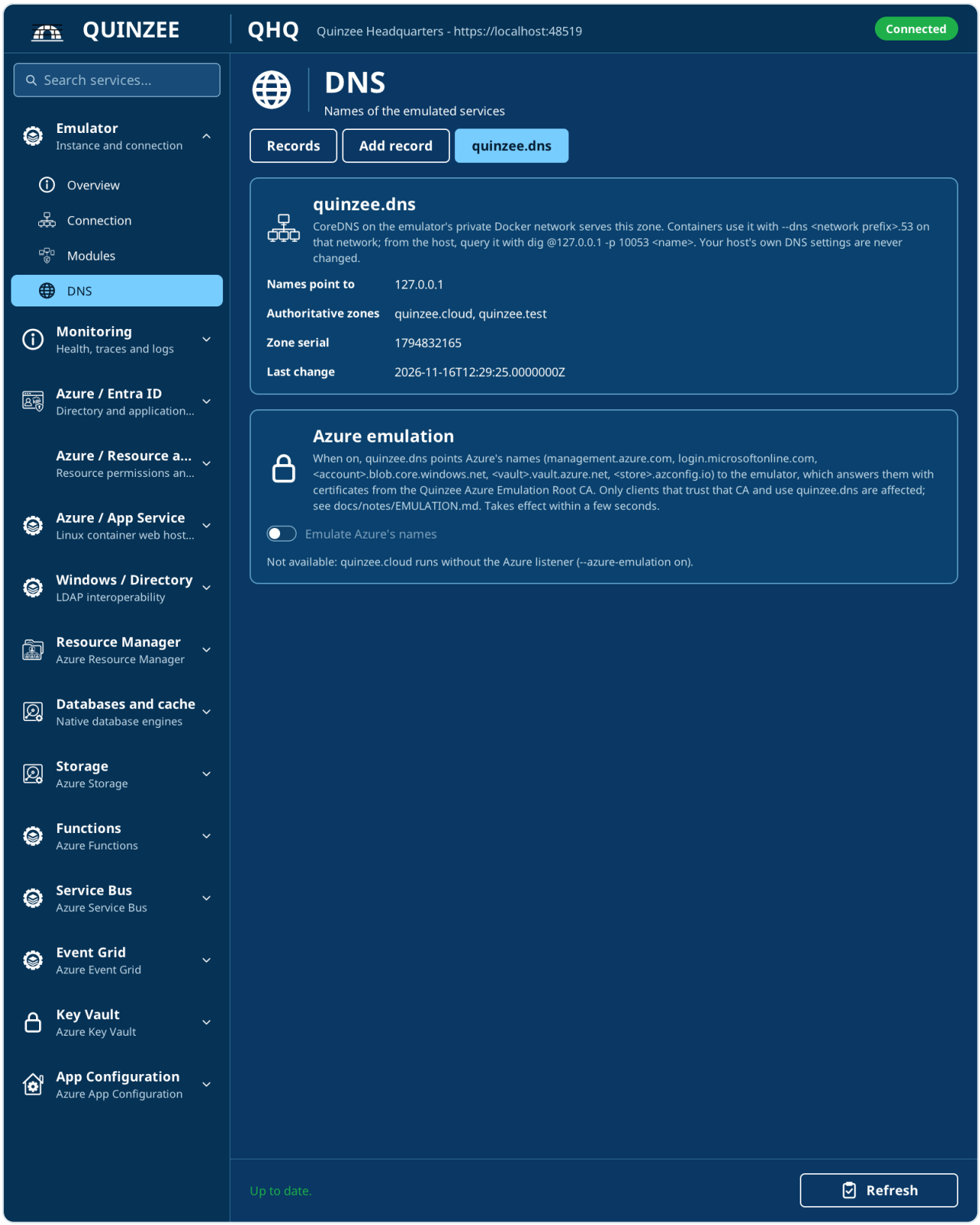
Records


QUINZEE
QHQ
Quinzee Headquarters - https://localhost:48519
Connected
Search services...
Emulator
Instance and connection
Overview
Connection
Modules
DNS
Monitoring
Health, traces and logs
Azure / Entra ID
Directory and application...
Azure / Resource a...
Resource permissions an...
Azure / App Service
Linux container web host...
Windows / Directory
LDAP interoperability
Resource Manager
Azure Resource Manager
Databases and cache
Native database engines
Storage
Azure Storage
Functions
Azure Functions
Service Bus
Azure Service Bus
Event Grid
Azure Event Grid
Key Vault
Azure Key Vault
App Configuration
Azure App Configuration
DNS
Names of the emulated services
Records
Add record
quinzee.dns
DNS records
Names quinzee.dns answers: quinzee.cloud, every storage account, vault, and configuration store (service), and records you add (custom). Only custom records can be deleted.
api.monitor.quinzee.cloud
127.0.0.1 - service - Monitoring
book-function.functions.quinzee.cloud
127.0.0.1 - service - Azure Functions
book-web.appservice.quinzee.cloud
127.0.0.1 - service - Azure App Service
bookclientstore.blob.storage.quinzee.cloud
127.0.0.1 - service - storage account bookclientstore
bookclientstore.queue.storage.quinzee.cloud
127.0.0.1 - service - storage account bookclientstore
bookclientstore.table.storage.quinzee.cloud
127.0.0.1 - service - storage account bookclientstore
booklab-bus.servicebus.quinzee.cloud
127.0.0.1 - service - Service Bus namespace
booklab-config.appconfig.quinzee.cloud
127.0.0.1 - service - configuration store booklab-config
booklab-orders.eventgrid.quinzee.cloud
127.0.0.1 - service - event grid topic booklab-orders
booklab-vault.vault.quinzee.cloud
127.0.0.1 - service - key vault booklab-vault
booklabevents.blob.storage.quinzee.cloud
127.0.0.1 - service - storage account booklabevents
booklabevents.queue.storage.quinzee.cloud
127.0.0.1 - service - storage account booklabevents
booklabevents.table.storage.quinzee.cloud
127.0.0.1 - service - storage account booklabevents
booklabstore.blob.storage.quinzee.cloud
127.0.0.1 - service - storage account booklabstore
booklabstore.queue.storage.quinzee.cloud
127.0.0.1 - service - storage account booklabstore
booklabstore.table.storage.quinzee.cloud
127.0.0.1 - service - storage account booklabstore
Up to date.
Refresh

Add record

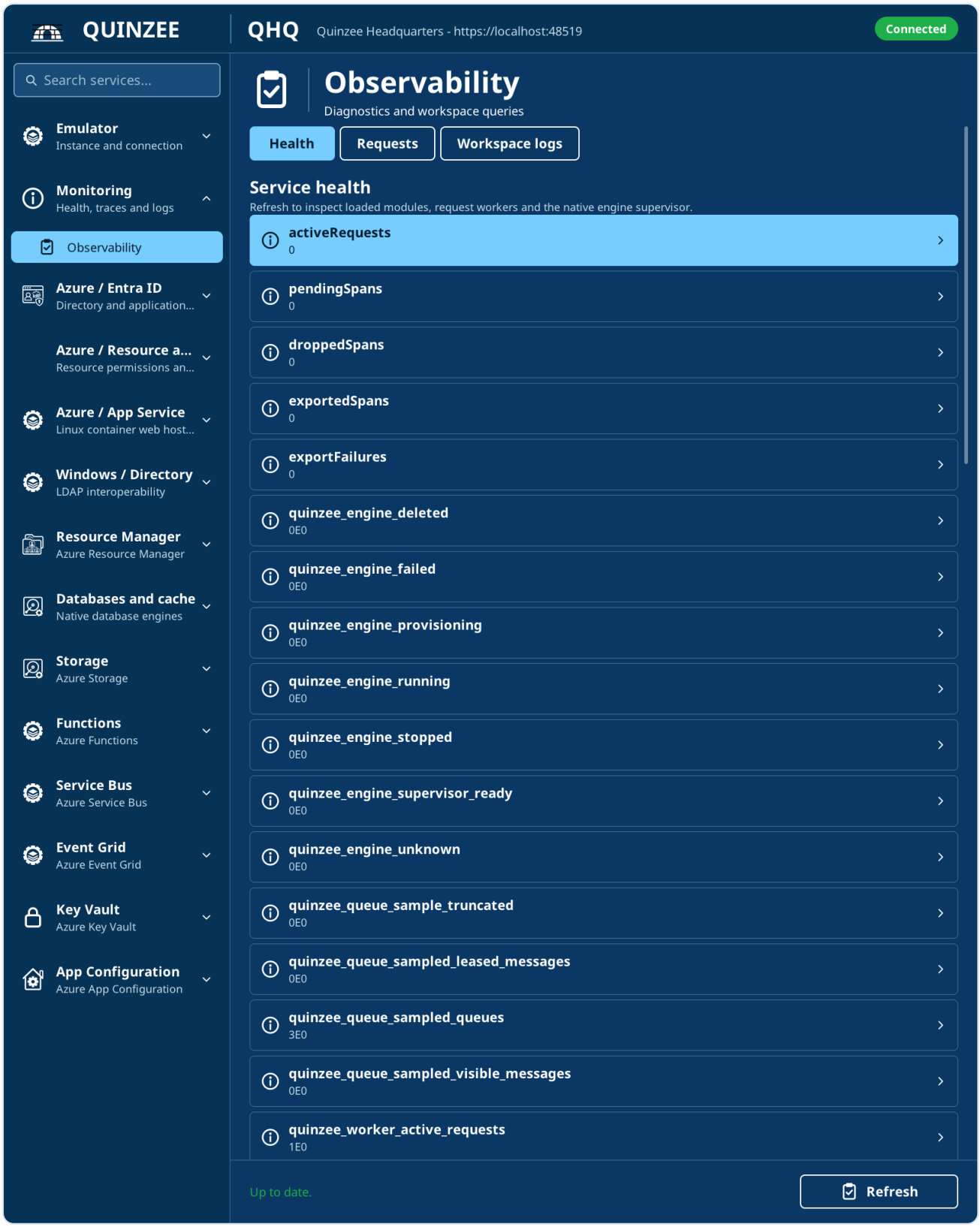

QUINZEE
QHQ
Quinzee Headquarters - https://localhost:48519
Connected
Search services...
Emulator
Instance and connection
Overview
Connection
Modules
DNS
Monitoring
Health, traces and logs
Azure / Entra ID
Directory and application...
Azure / Resource a...
Resource permissions an...
Azure / App Service
Linux container web host...
Windows / Directory
LDAP interoperability
Resource Manager
Azure Resource Manager
Databases and cache
Native database engines
Storage
Azure Storage
Functions
Azure Functions
Service Bus
Azure Service Bus
Event Grid
Azure Event Grid
Key Vault
Azure Key Vault
App Configuration
Azure App Configuration
DNS
Names of the emulated services
Records
Add record
quinzee.dns
Custom record
An A record for a name your containers should reach, for example app.quinzee.test. Without an address it points to the emulator. Names outside the certificate's names will not pass TLS verification when they point to the emulator.
Name
for example app.quinzee.test
Address
IPv4; empty: the emulator
Note
optional
Add record
Up to date.
Refresh

DNS server

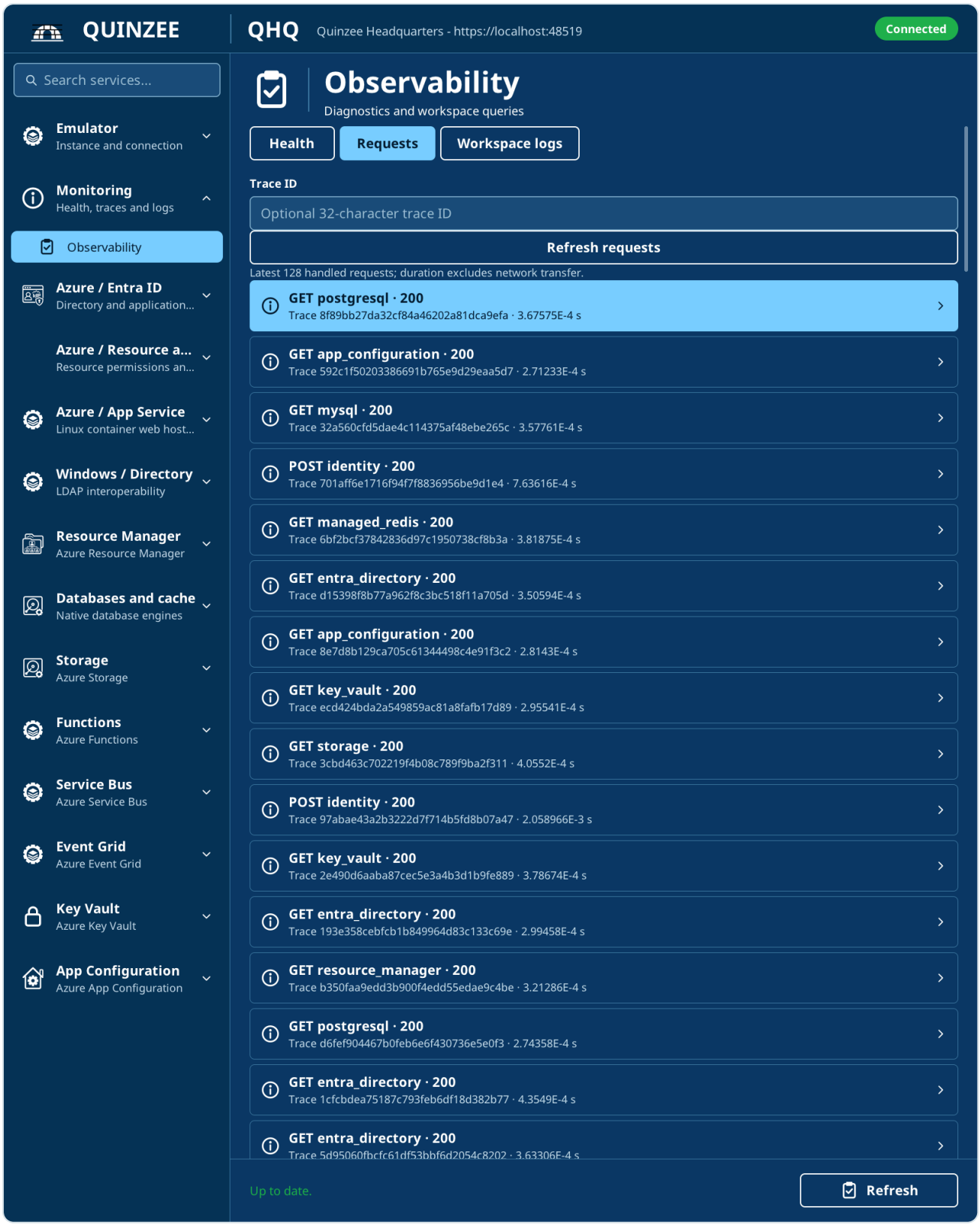

QUINZEE
QHQ
Quinzee Headquarters - https://localhost:48519
Connected
Search services...
Emulator
Instance and connection
Overview
Connection
Modules
DNS
Monitoring
Health, traces and logs
Azure / Entra ID
Directory and application...
Azure / Resource a...
Resource permissions an...
Azure / App Service
Linux container web host...
Windows / Directory
LDAP interoperability
Resource Manager
Azure Resource Manager
Databases and cache
Native database engines
Storage
Azure Storage
Functions
Azure Functions
Service Bus
Azure Service Bus
Event Grid
Azure Event Grid
Key Vault
Azure Key Vault
App Configuration
Azure App Configuration
DNS
Names of the emulated services
Records
Add record
quinzee.dns
quinzee.dns
CoreDNS on the emulator's private Docker network serves this zone. Containers use it with --dns <network prefix>.53 on that network; from the host, query it with dig @127.0.0.1 -p 10053 <name>. Your host's own DNS settings are never changed.
Names point to
127.0.0.1
Authoritative zones
quinzee.cloud, quinzee.test
Zone serial
1794832165
Last change
2026-11-16T12:29:25.0000000Z
Azure emulation
When on, quinzee.dns points Azure's names (management.azure.com, login.microsoftonline.com, <account>.blob.core.windows.net, <vault>.vault.azure.net, <store>.azconfig.io) to the emulator, which answers them with certificates from the Quinzee Azure Emulation Root CA. Only clients that trust that CA and use quinzee.dns are affected; see docs/notes/EMULATION.md. Takes effect within a few seconds.
Emulate Azure's names
Not available: quinzee.cloud runs without the Azure listener (--azure-emulation on).
Up to date.
Refresh

Observability

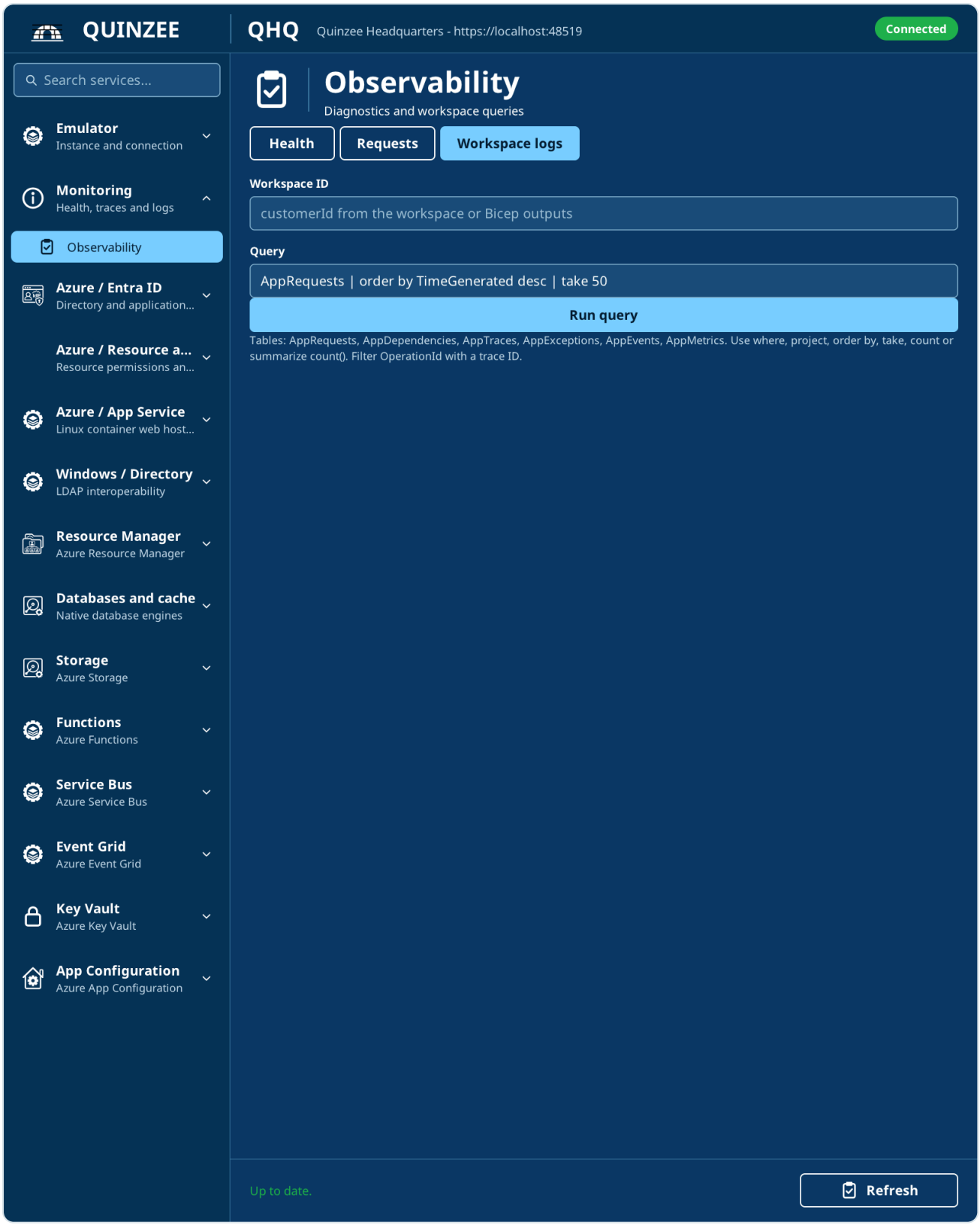
Health



Requests

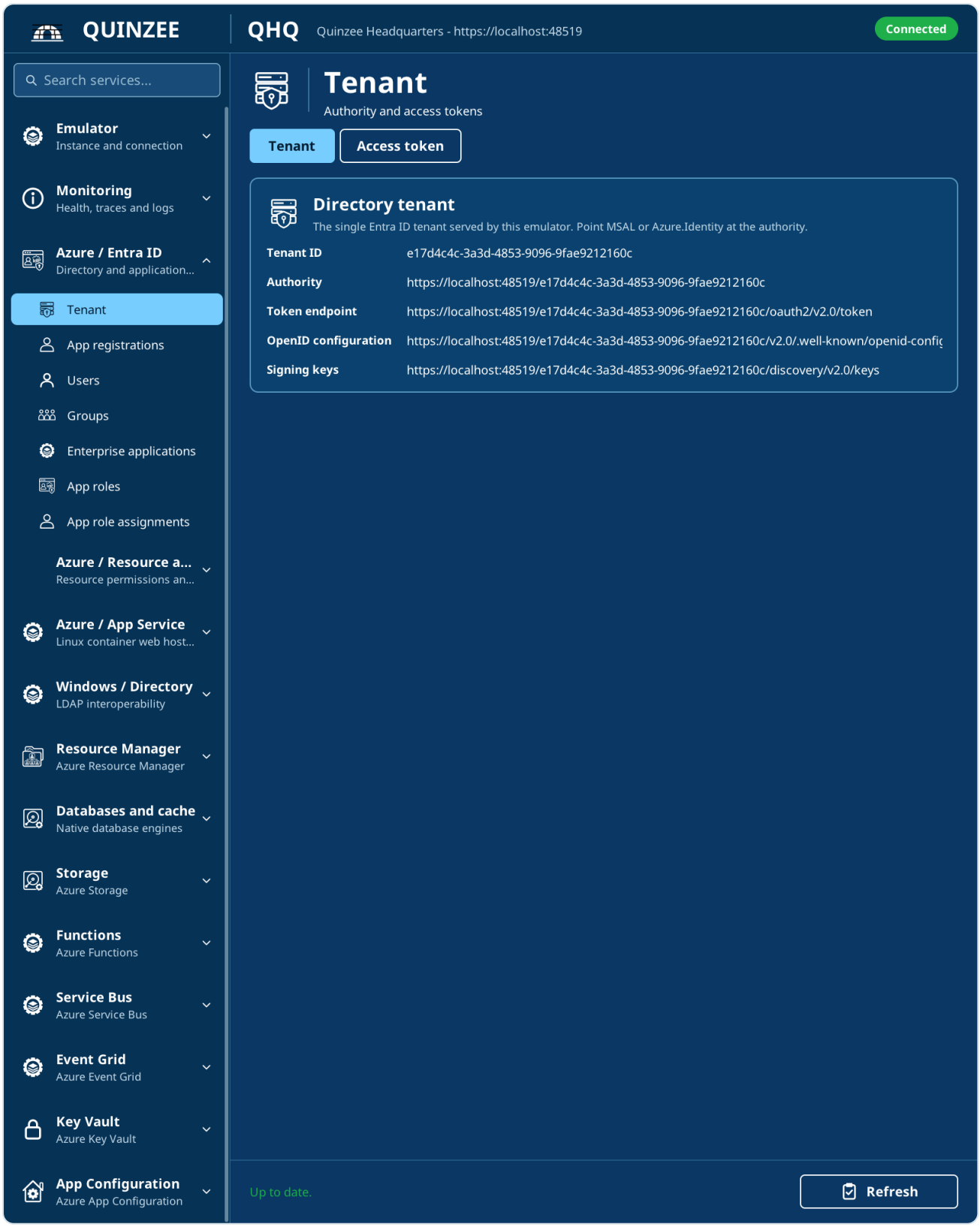

QUINZEE
QHQ
Quinzee Headquarters - https://localhost:48519
Connected
Search services...
Emulator
Instance and connection
Monitoring
Health, traces and logs
Observability
Azure / Entra ID
Directory and application...
Azure / Resource a...
Resource permissions an...
Azure / App Service
Linux container web host...
Windows / Directory
LDAP interoperability
Resource Manager
Azure Resource Manager
Databases and cache
Native database engines
Storage
Azure Storage
Functions
Azure Functions
Service Bus
Azure Service Bus
Event Grid
Azure Event Grid
Key Vault
Azure Key Vault
App Configuration
Azure App Configuration
Observability
Diagnostics and workspace queries
Health
Requests
Workspace logs
Trace ID
Optional 32-character trace ID
Refresh requests
Latest 128 handled requests; duration excludes network transfer.
GET postgresql · 200
Trace 8f89bb27da32cf84a46202a81dca9efa · 3.67575E-4 s
GET app_configuration · 200
Trace 592c1f50203386691b765e9d29eaa5d7 · 2.71233E-4 s
GET mysql · 200
Trace 32a560cfd5dae4c114375af48ebe265c · 3.57761E-4 s
POST identity · 200
Trace 701aff6e1716f94f7f8836956be9d1e4 · 7.63616E-4 s
GET managed_redis · 200
Trace 6bf2bcf37842836d97c1950738cf8b3a · 3.81875E-4 s
GET entra_directory · 200
Trace d15398f8b77a962f8c3bc518f11a705d · 3.50594E-4 s
GET app_configuration · 200
Trace 8e7d8b129ca705c61344498c4e91f3c2 · 2.8143E-4 s
GET key_vault · 200
Trace ecd424bda2a549859ac81a8fafb17d89 · 2.95541E-4 s
GET storage · 200
Trace 3cbd463c702219f4b08c789f9ba2f311 · 4.0552E-4 s
POST identity · 200
Trace 97abae43a2b3222d7f714b5fd8b07a47 · 2.058966E-3 s
GET key_vault · 200
Trace 2e490d6aaba87cec5e3a4b3d1b9fe889 · 3.78674E-4 s
GET entra_directory · 200
Trace 193e358cebfcb1b849964d83c133c69e · 2.99458E-4 s
GET resource_manager · 200
Trace b350faa9edd3b900f4edd55edae9c4be · 3.21286E-4 s
GET postgresql · 200
Trace d6fef904467b0feb6e6f430736e5e0f3 · 2.74358E-4 s
GET entra_directory · 200
Trace 1cfcbdea75187c793feb6df18d382b77 · 4.3549E-4 s
GET entra_directory · 200
Trace 5d95060fbcfc61df53bbf6d2054c8202 · 3.63306E-4 s
Up to date.
Refresh

Workspace logs

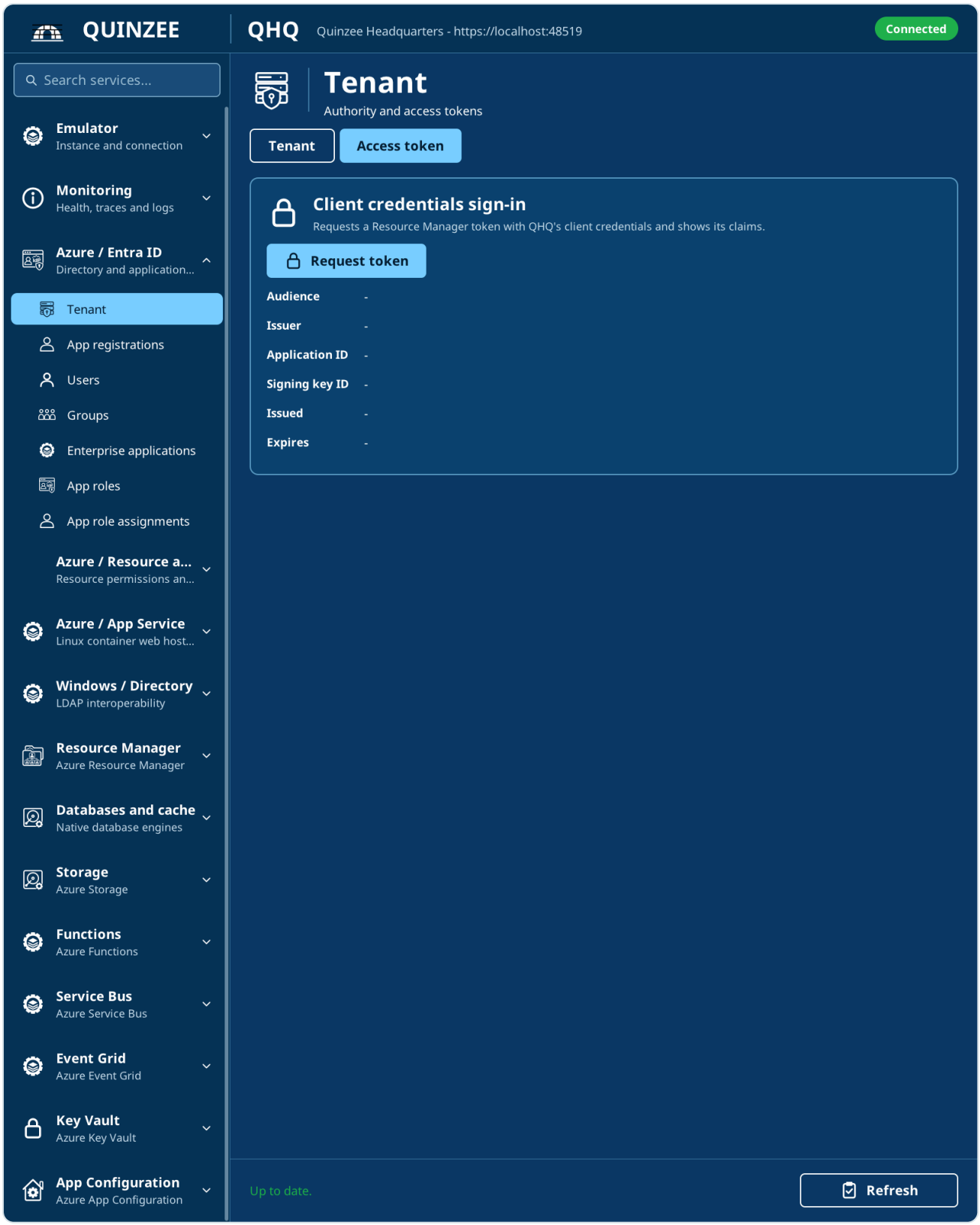


Tenant

Tenant

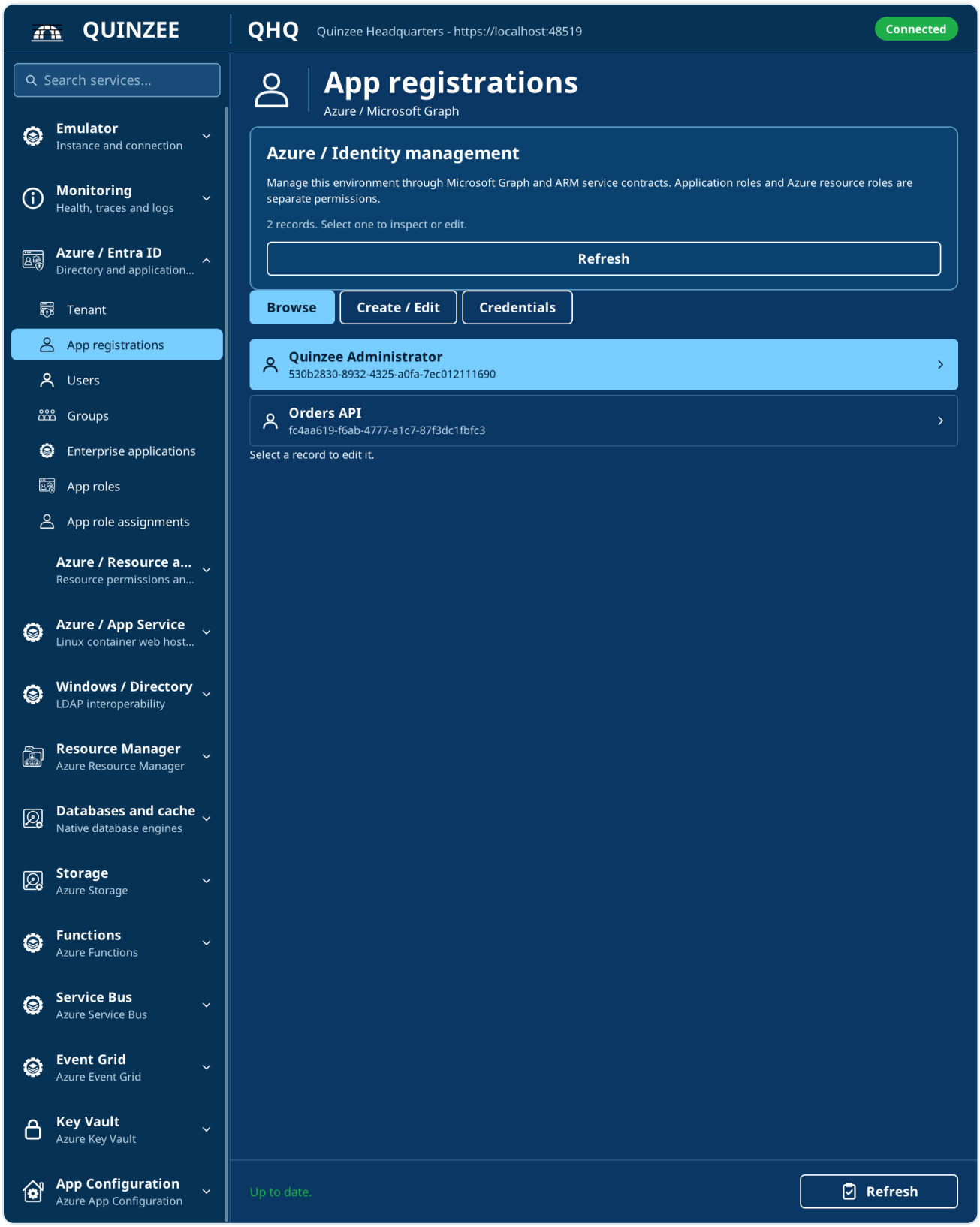


Access token

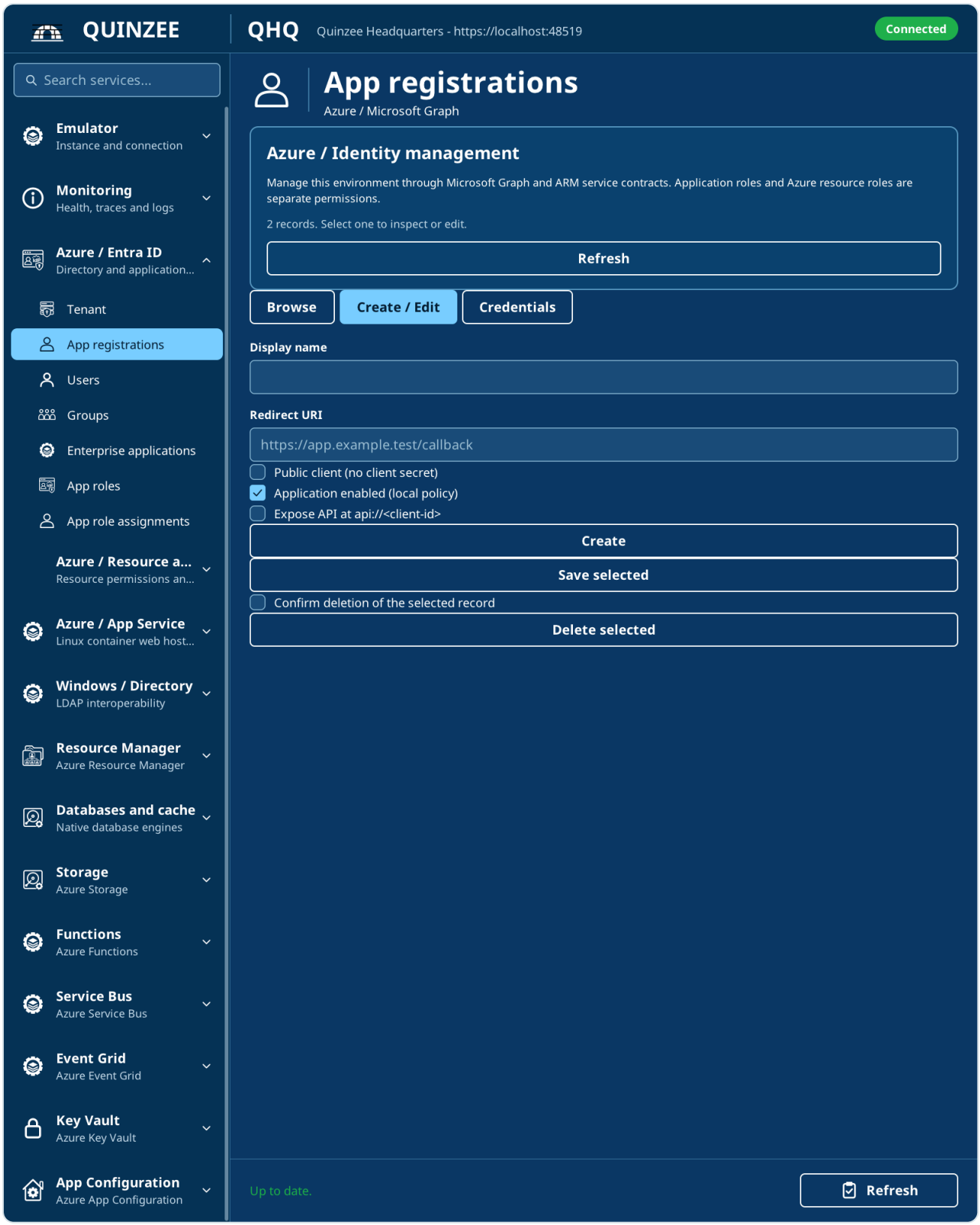


Applications

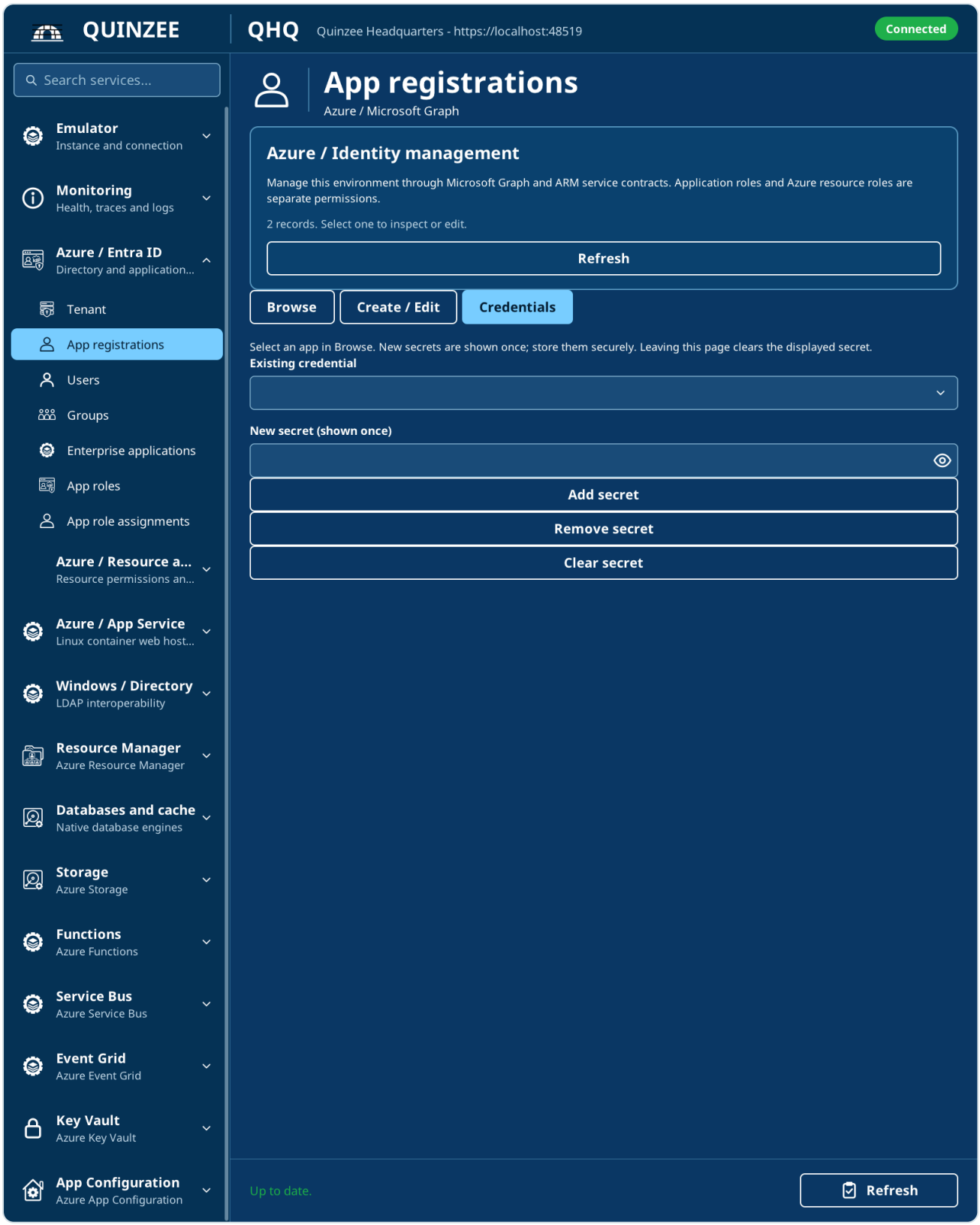
Browse



Create and edit

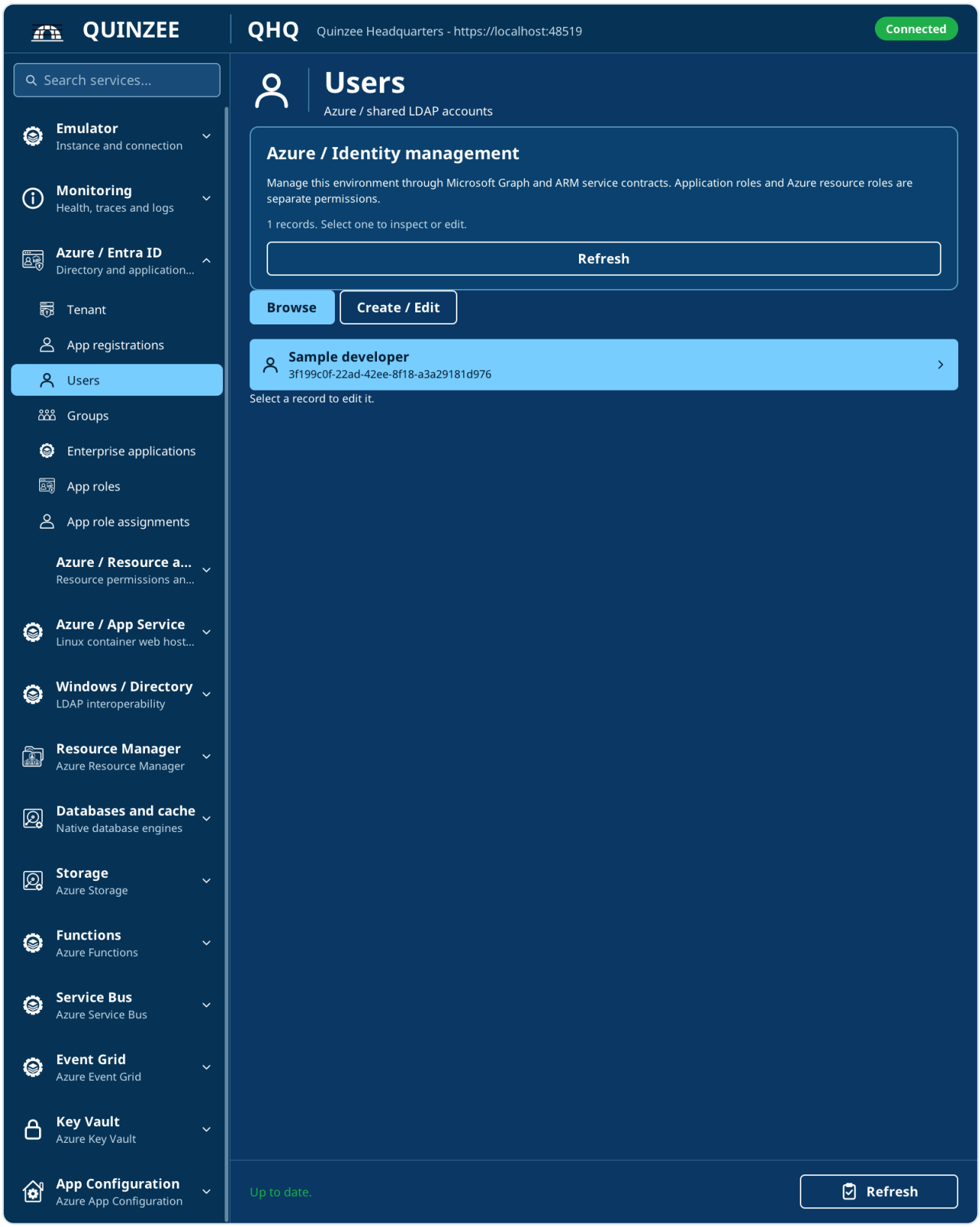


Credentials

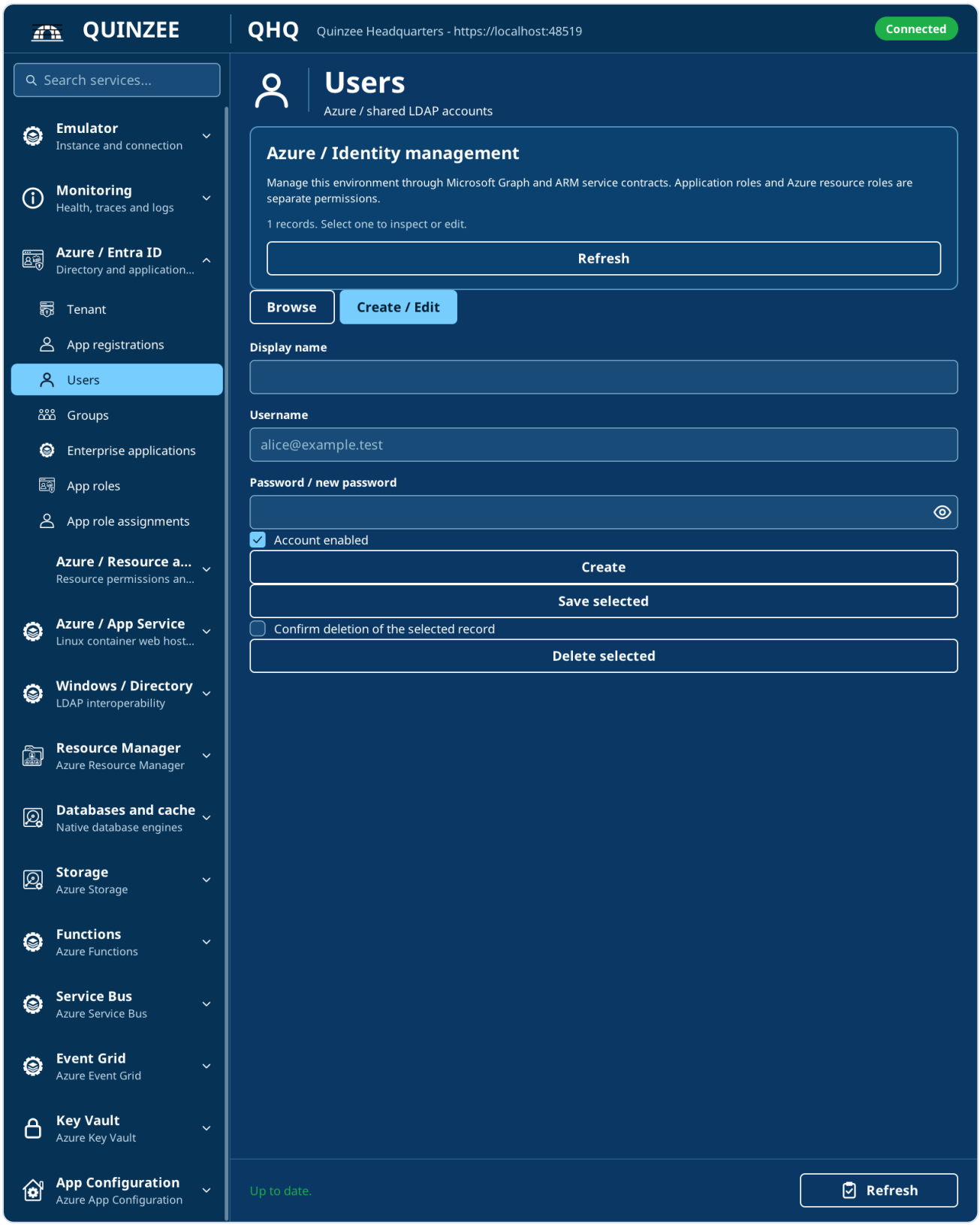


Directory Users

Browse

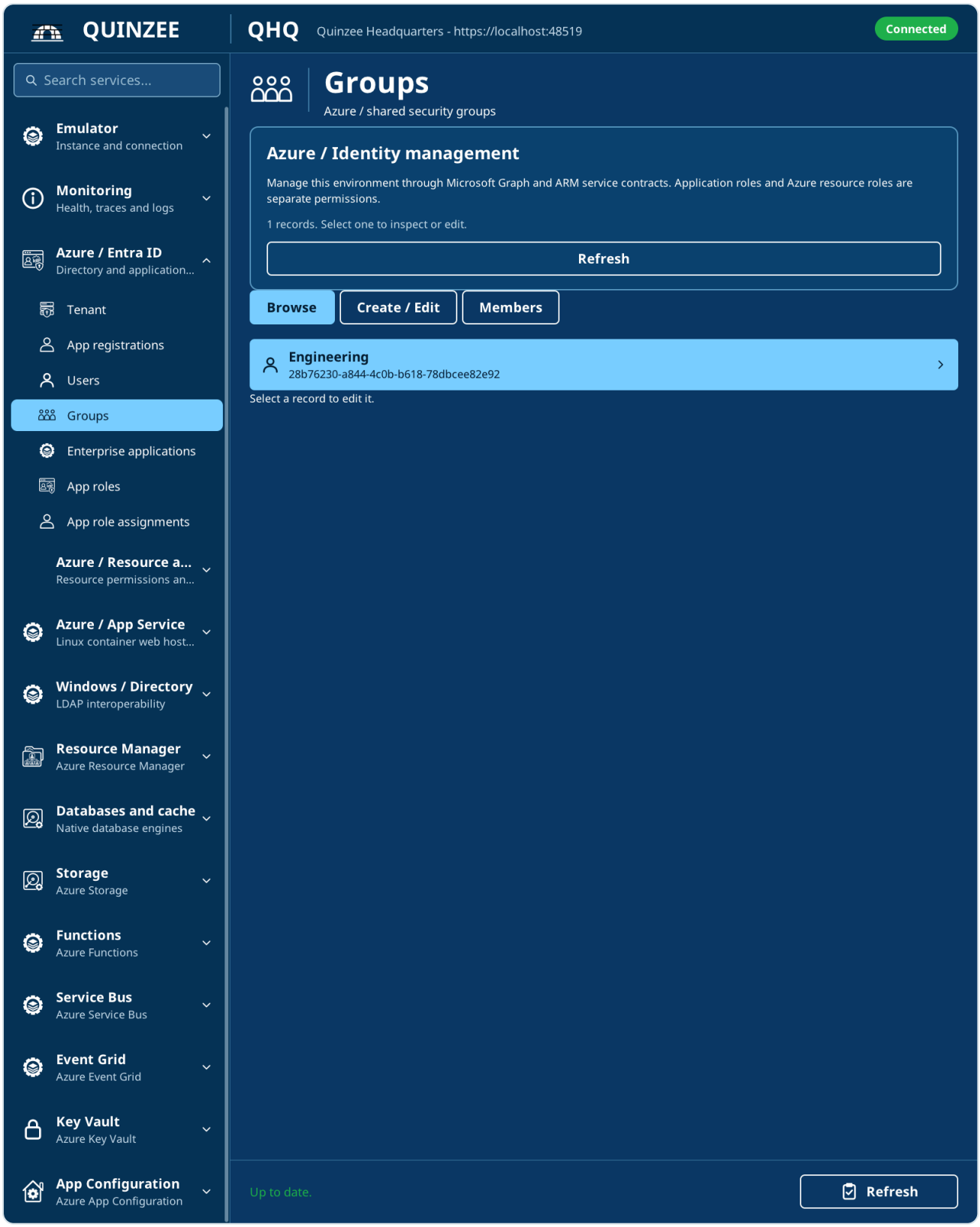


Create and edit

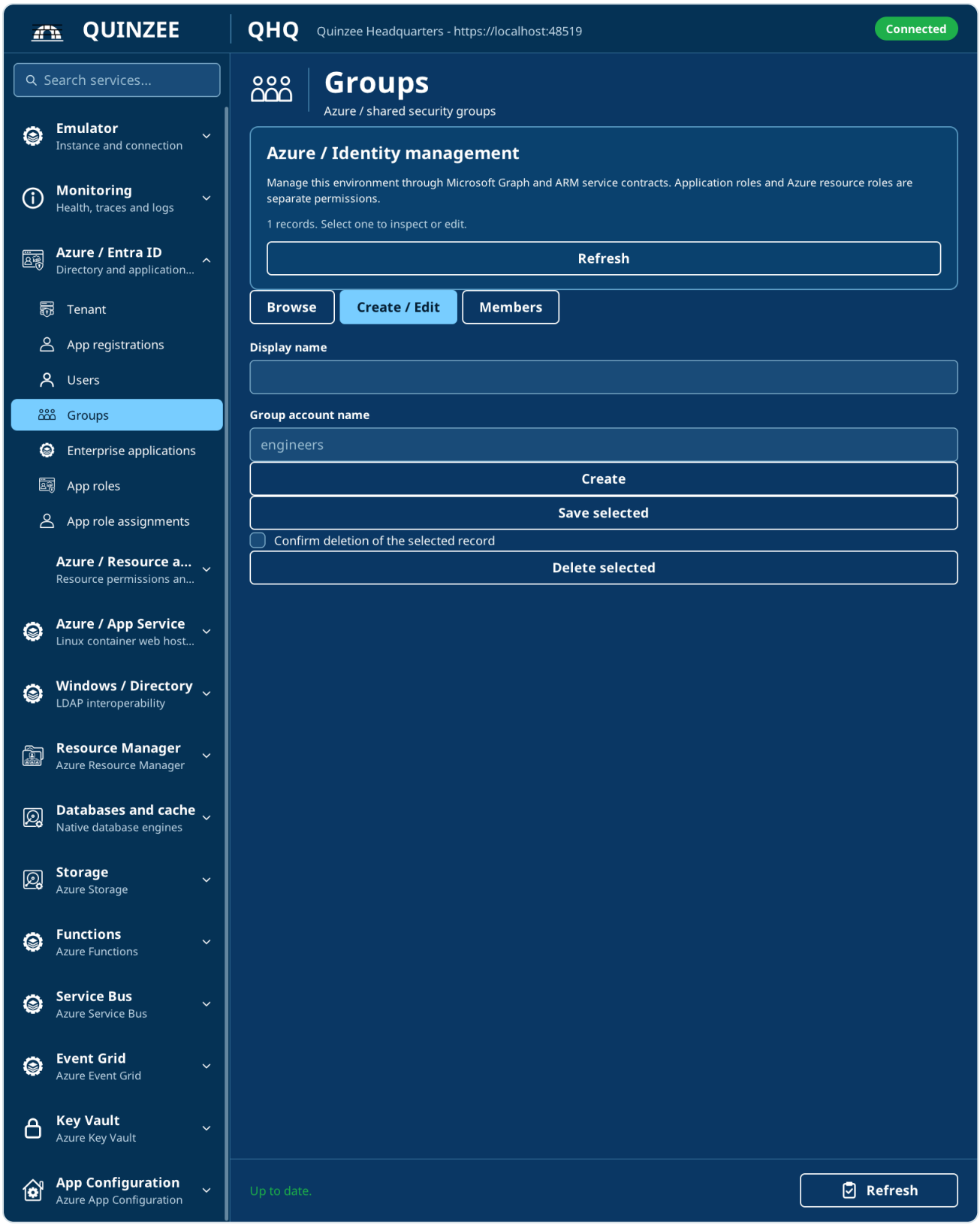


Directory Groups

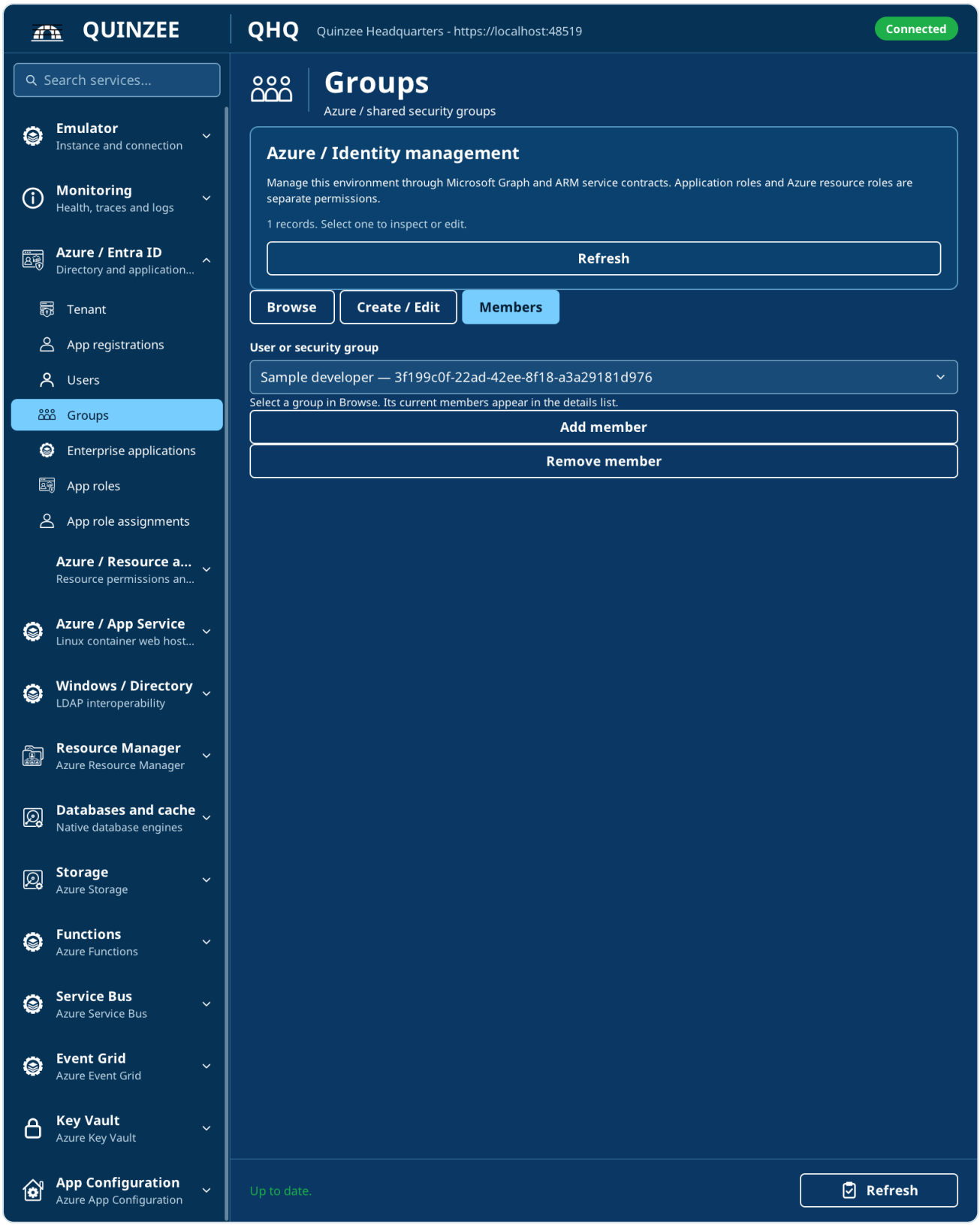
Browse



Create and edit

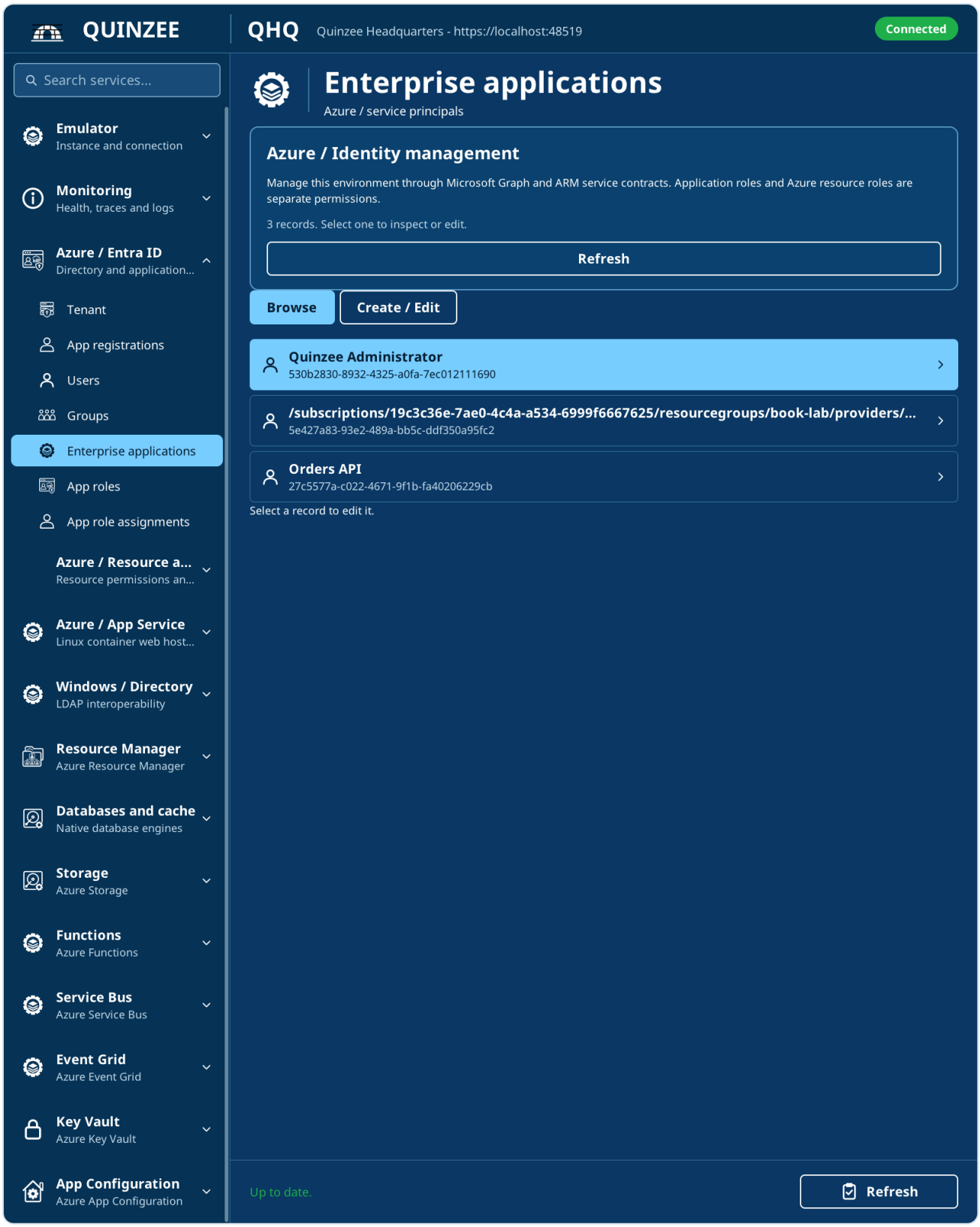


Members

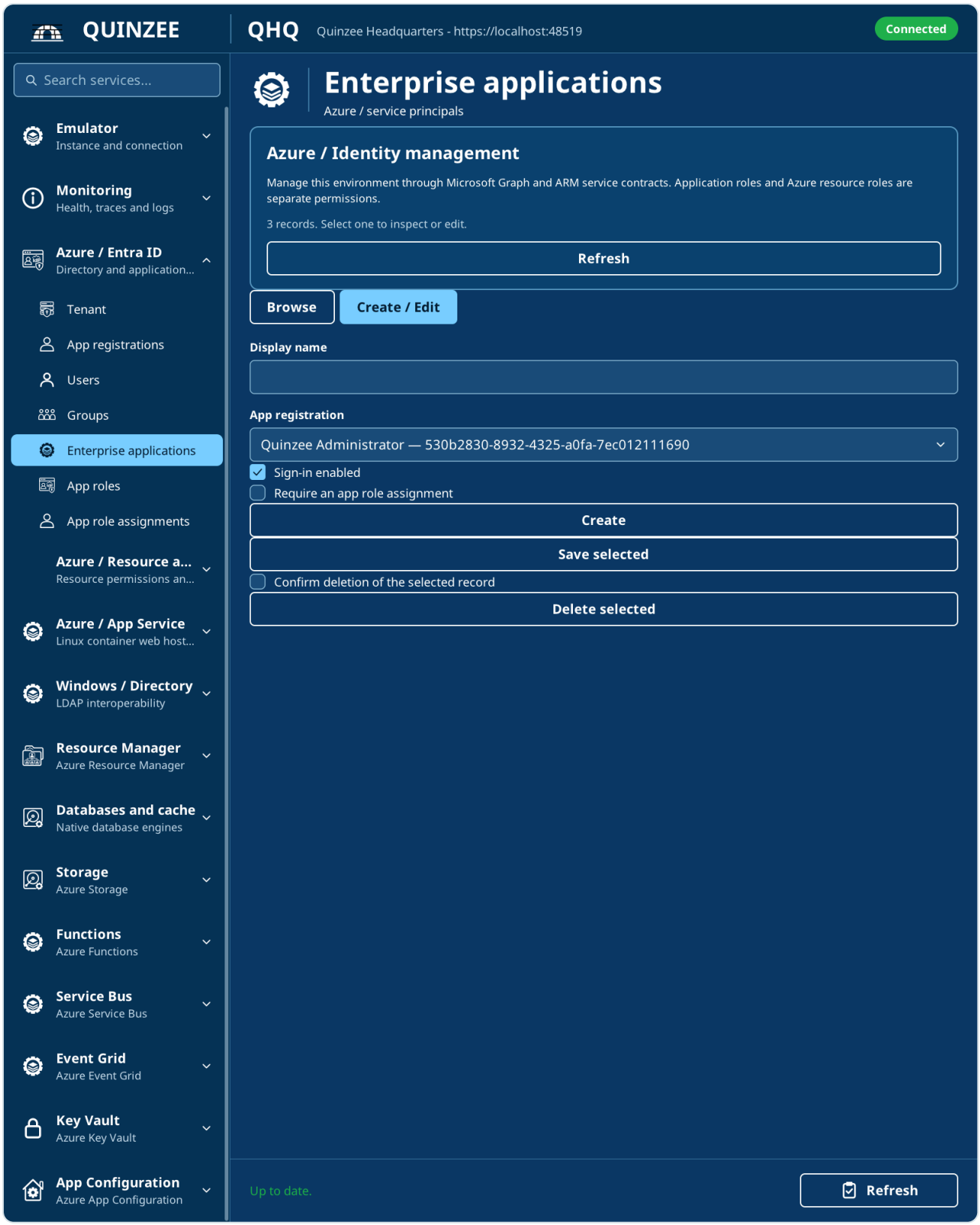

QUINZEE
QHQ
Quinzee Headquarters - https://localhost:48519
Connected
Search services...
Emulator
Instance and connection
Monitoring
Health, traces and logs
Azure / Entra ID
Directory and application...
Tenant
App registrations
Users
Groups
Enterprise applications
App roles
App role assignments
Azure / Resource a...
Resource permissions an...
Azure / App Service
Linux container web host...
Windows / Directory
LDAP interoperability
Resource Manager
Azure Resource Manager
Databases and cache
Native database engines
Storage
Azure Storage
Functions
Azure Functions
Service Bus
Azure Service Bus
Event Grid
Azure Event Grid
Key Vault
Azure Key Vault
App Configuration
Azure App Configuration
Groups
Azure / shared security groups
Azure / Identity management
Manage this environment through Microsoft Graph and ARM service contracts. Application roles and Azure resource roles are separate permissions.
1 records. Select one to inspect or edit.
Refresh
Browse
Create / Edit
Members
User or security group
Sample developer — 3f199c0f-22ad-42ee-8f18-a3a29181d976
Select a group in Browse. Its current members appear in the details list.
Add member
Remove member
Up to date.
Refresh

Enterprise Apps

Browse

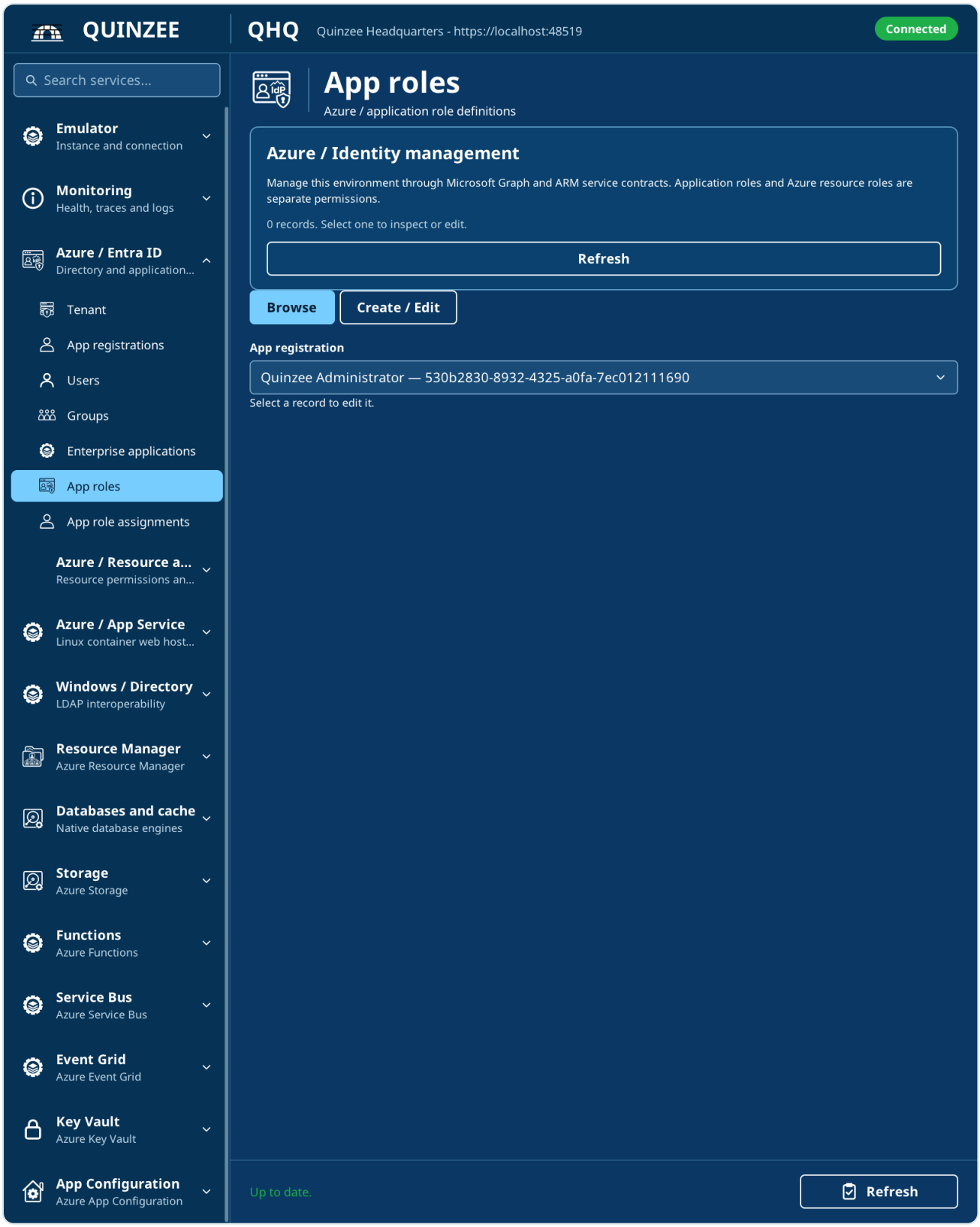


Create and edit

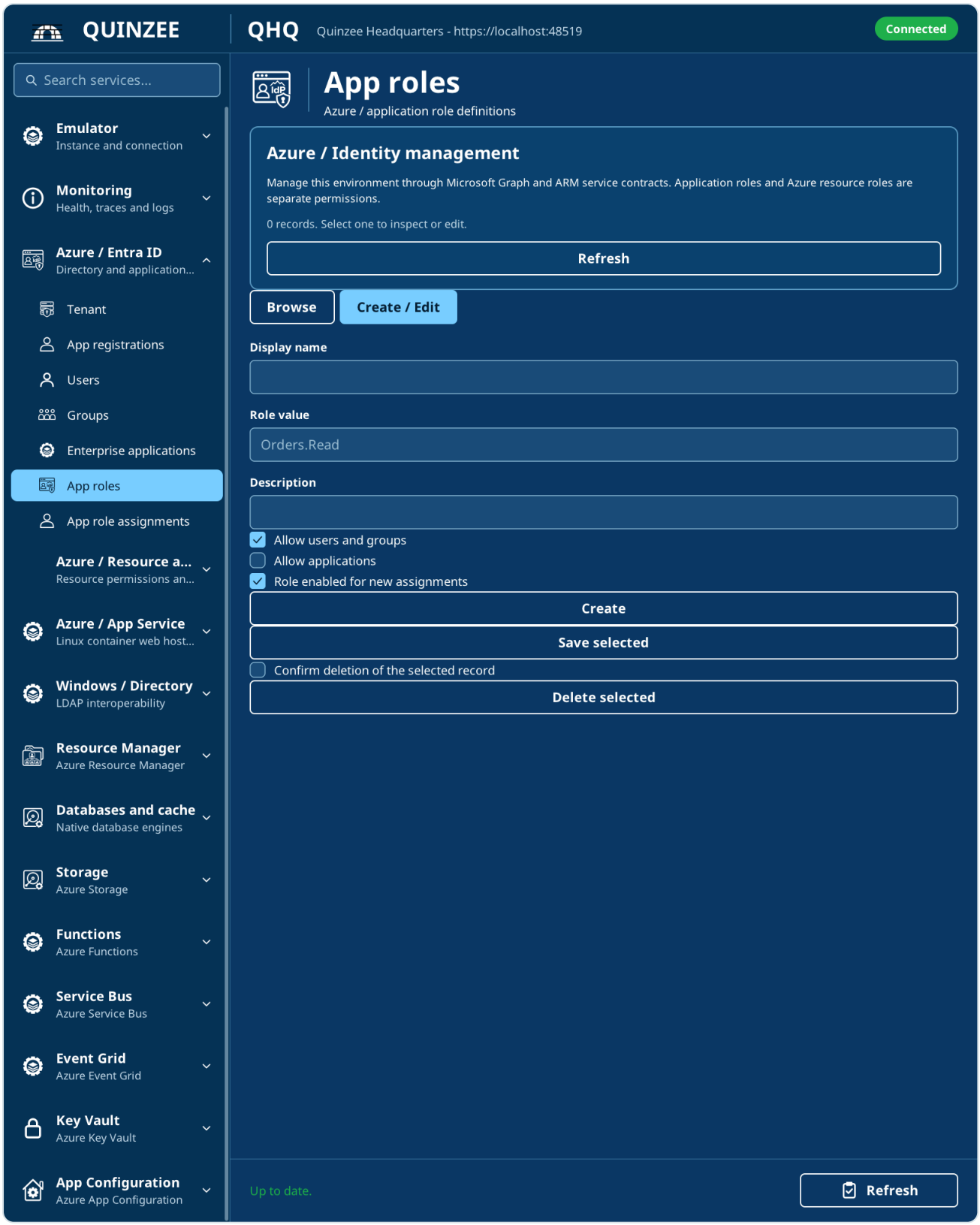


App Roles

Browse

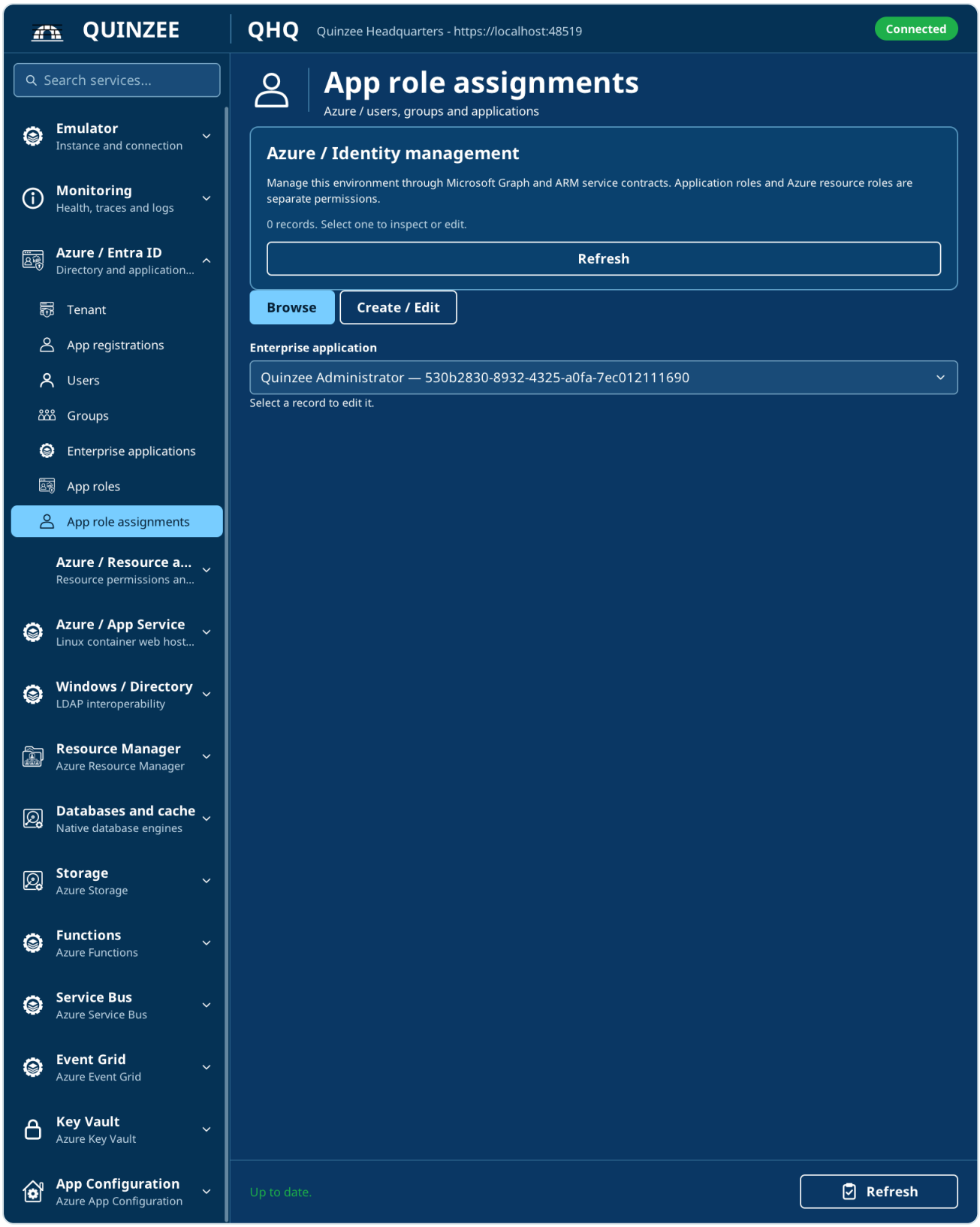


Create and edit

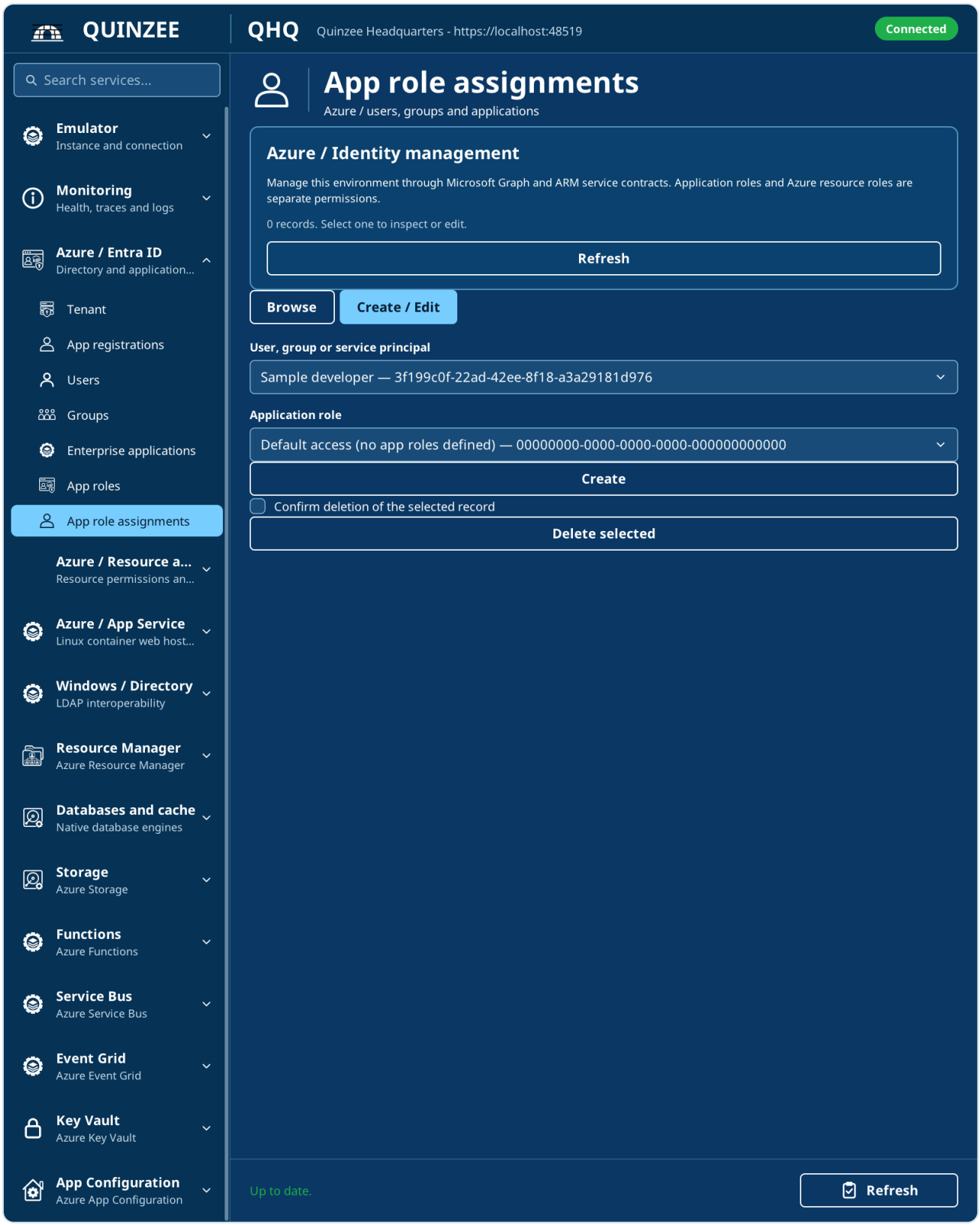


App Assignments

Browse

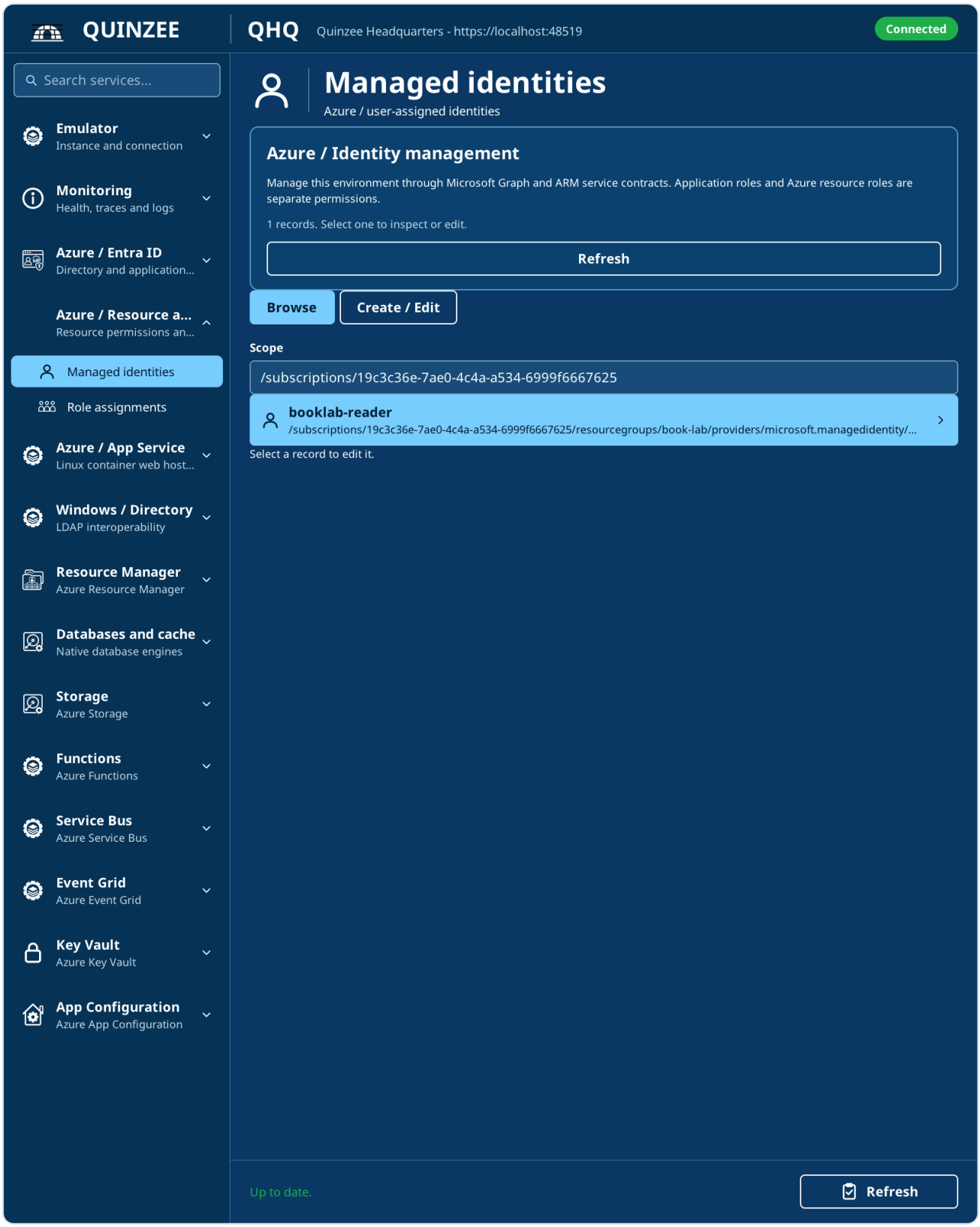


Create and edit

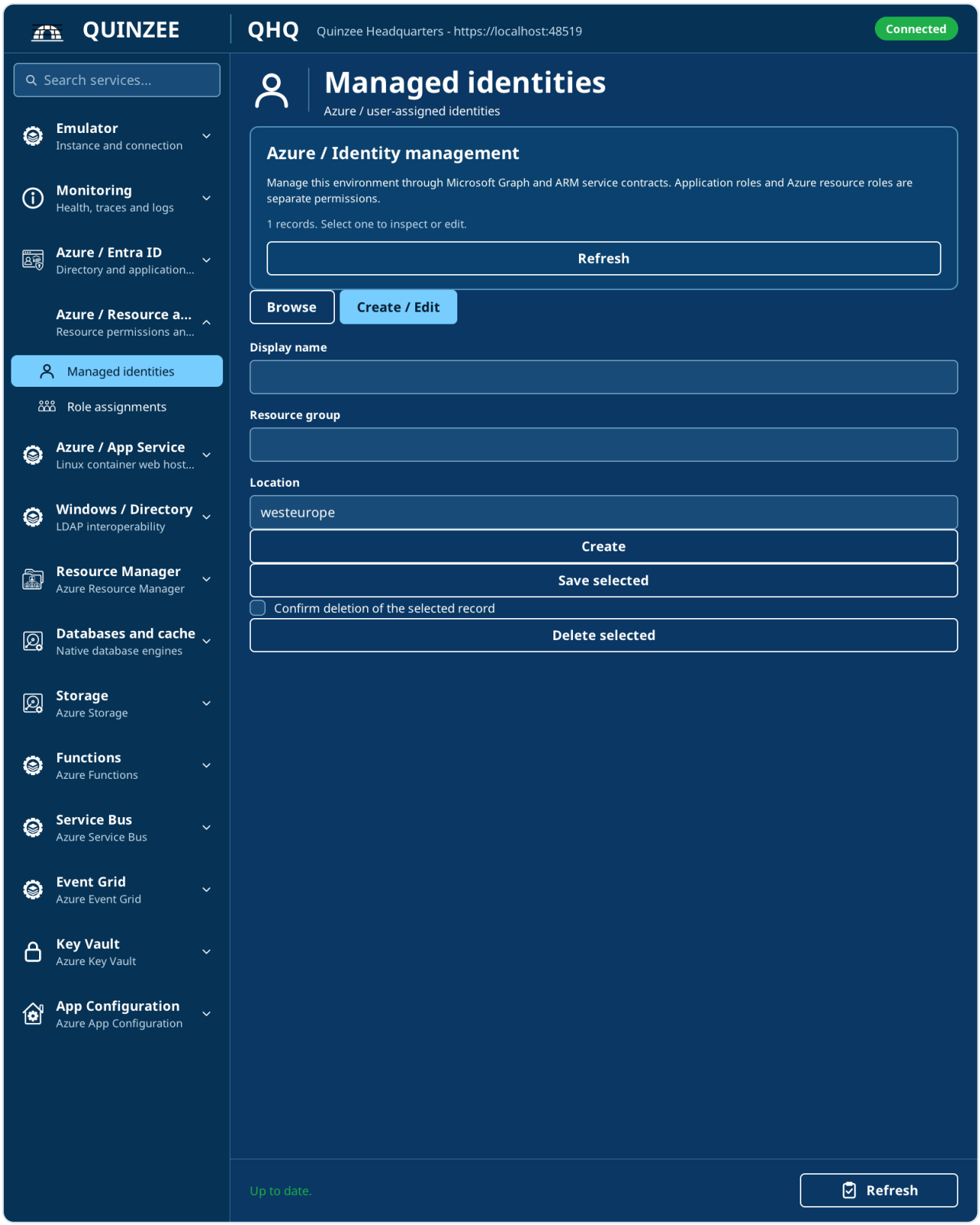


Managed Identities

Browse

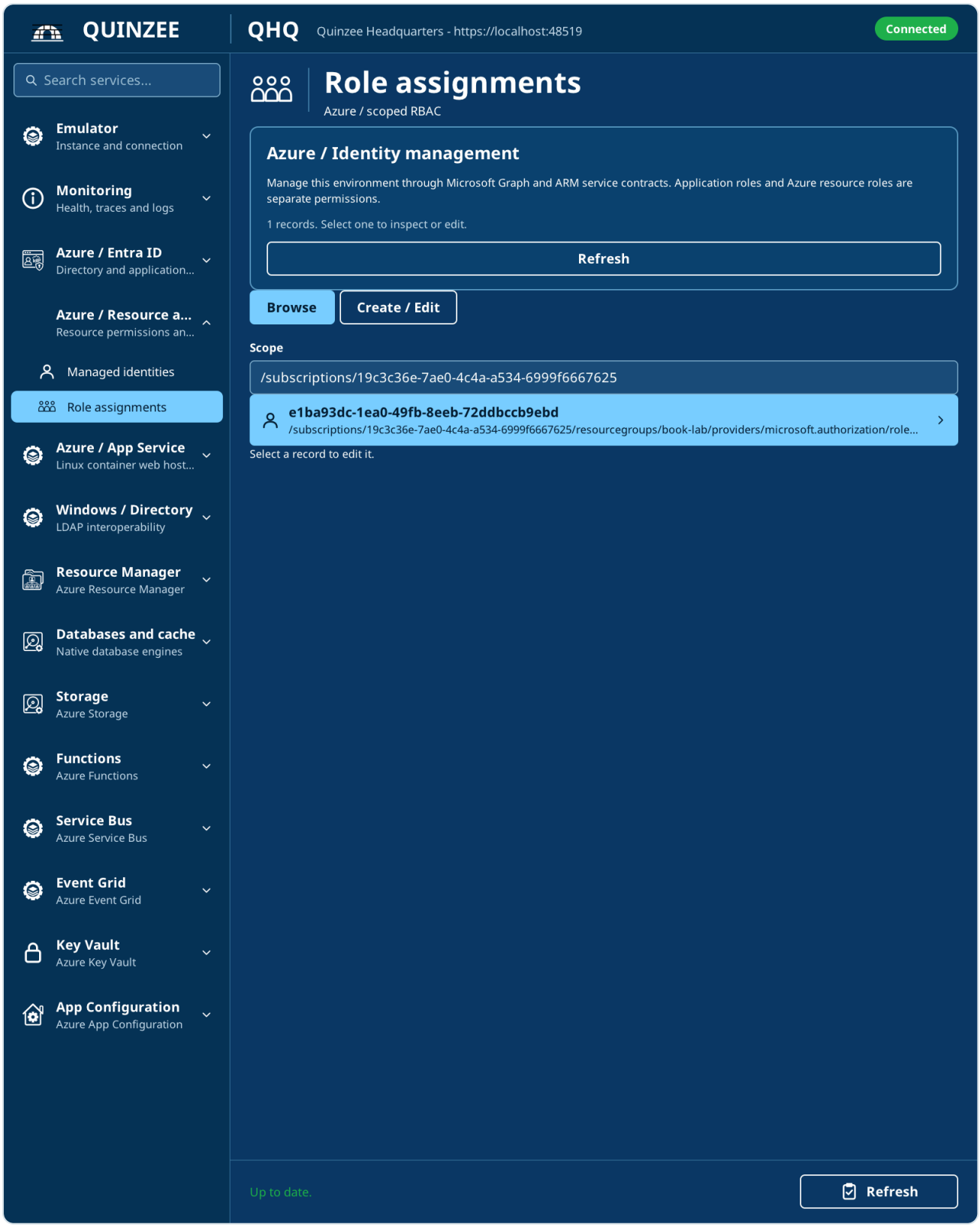


Create and edit

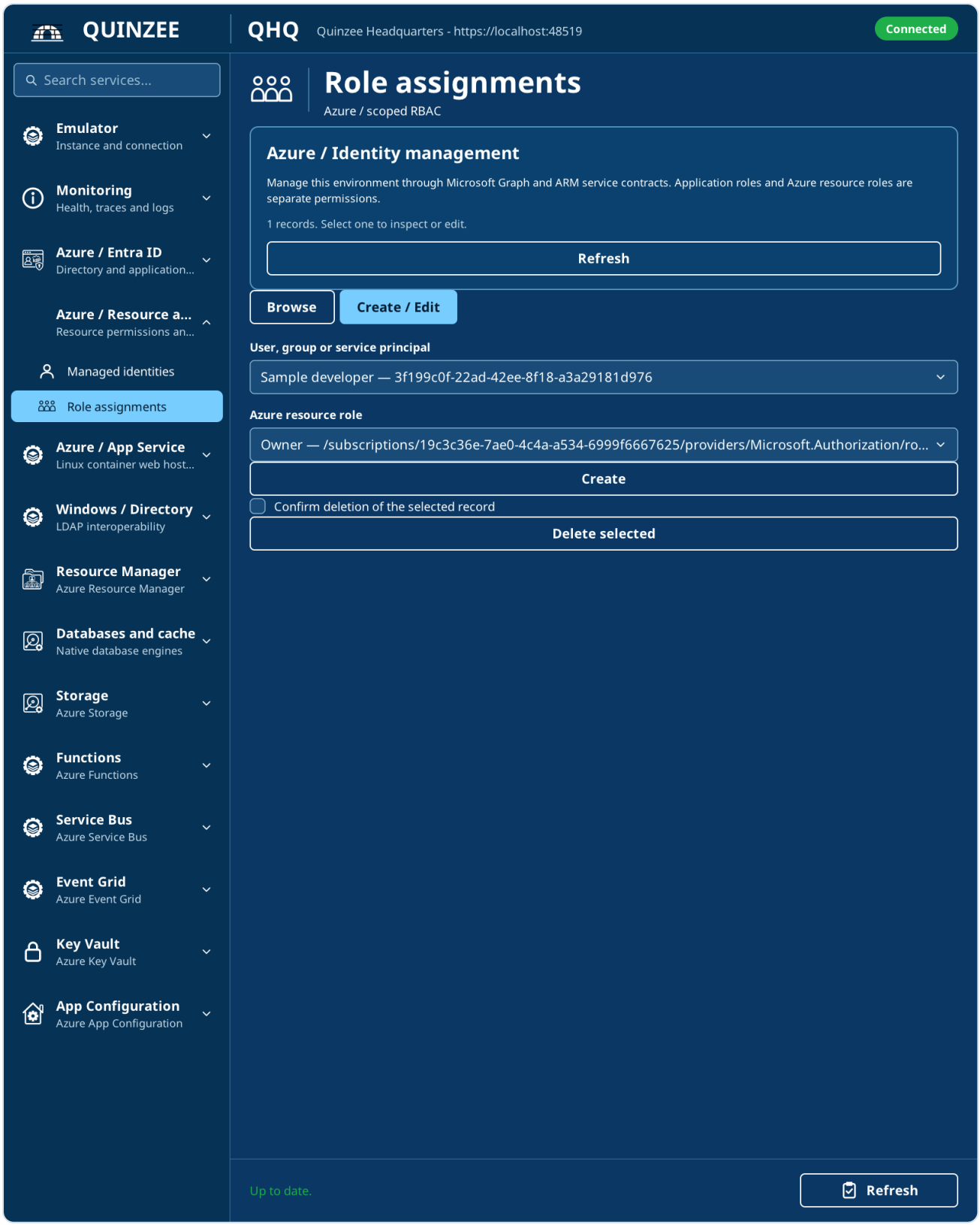


Rbac

Browse

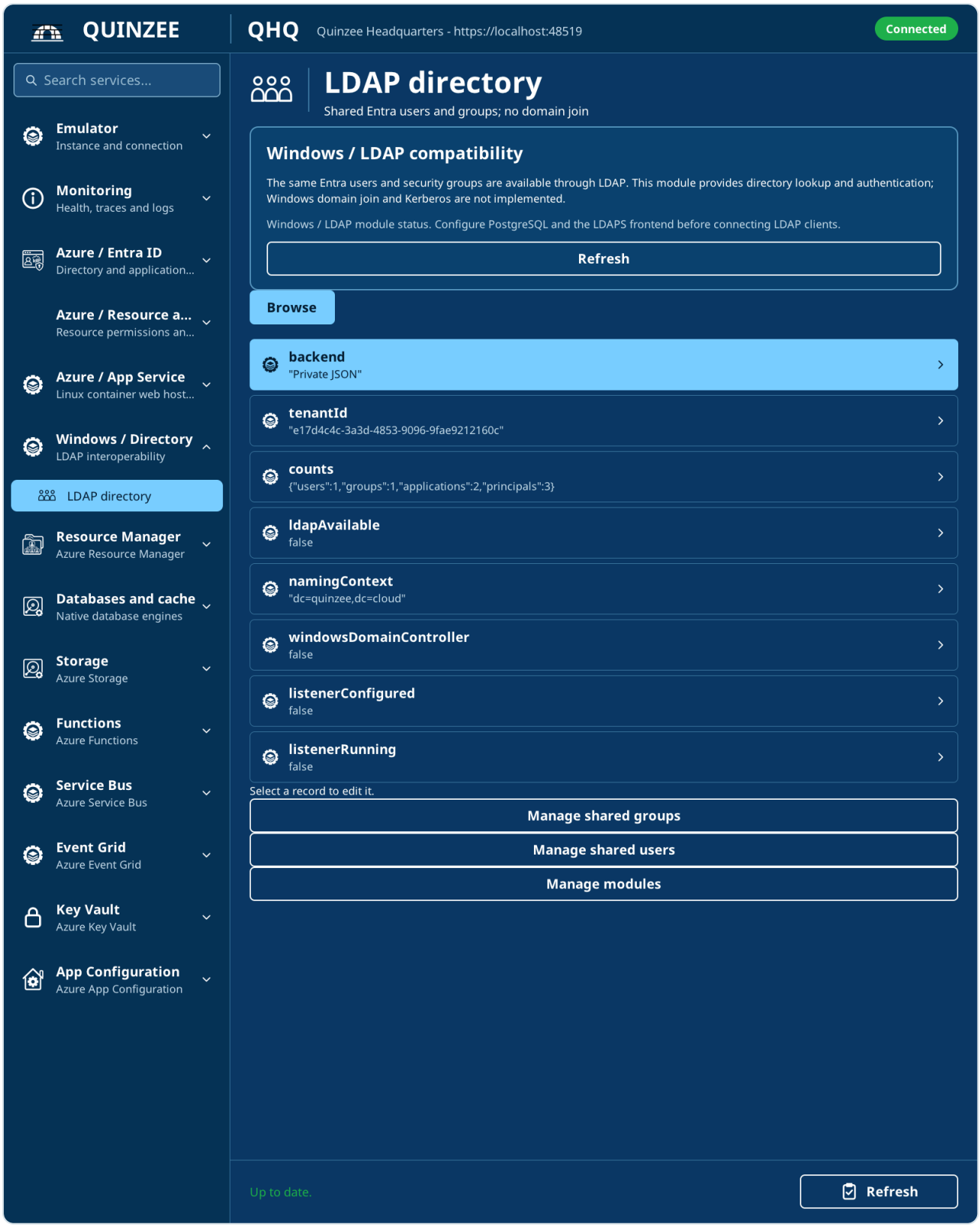

QUINZEE
QHQ
Quinzee Headquarters - https://localhost:48519
Connected
Search services...
Emulator
Instance and connection
Monitoring
Health, traces and logs
Azure / Entra ID
Directory and application...
Azure / Resource a...
Resource permissions an...
Managed identities
Role assignments
Azure / App Service
Linux container web host...
Windows / Directory
LDAP interoperability
Resource Manager
Azure Resource Manager
Databases and cache
Native database engines
Storage
Azure Storage
Functions
Azure Functions
Service Bus
Azure Service Bus
Event Grid
Azure Event Grid
Key Vault
Azure Key Vault
App Configuration
Azure App Configuration
Role assignments
Azure / scoped RBAC
Azure / Identity management
Manage this environment through Microsoft Graph and ARM service contracts. Application roles and Azure resource roles are separate permissions.
1 records. Select one to inspect or edit.
Refresh
Browse
Create / Edit
Scope
/subscriptions/19c3c36e-7ae0-4c4a-a534-6999f6667625
e1ba93dc-1ea0-49fb-8eeb-72ddbccb9ebd
/subscriptions/19c3c36e-7ae0-4c4a-a534-6999f6667625/resourcegroups/book-lab/providers/microsoft.authorization/role...
Select a record to edit it.
Up to date.
Refresh

Create and edit



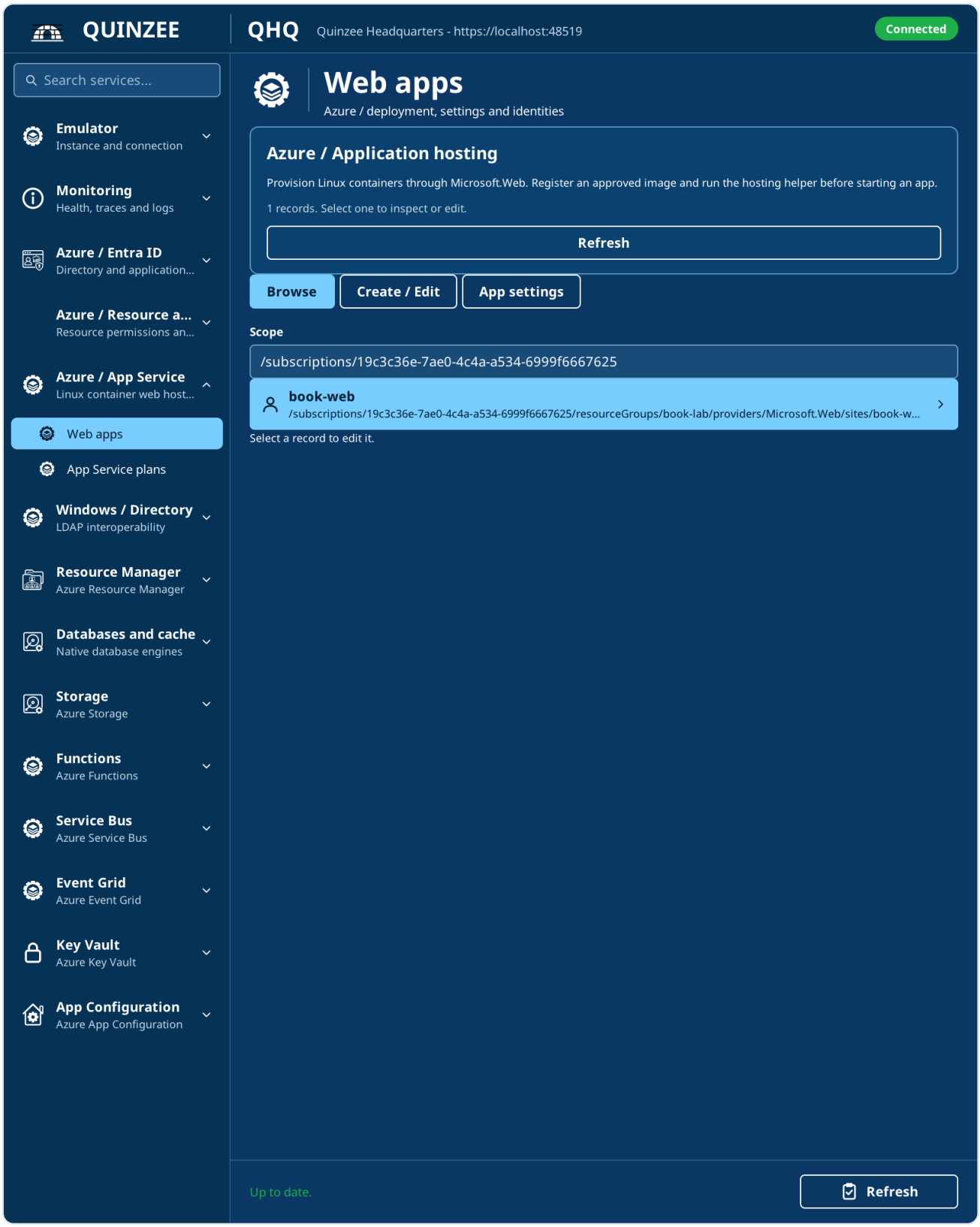
Windows Ldap

Directory status

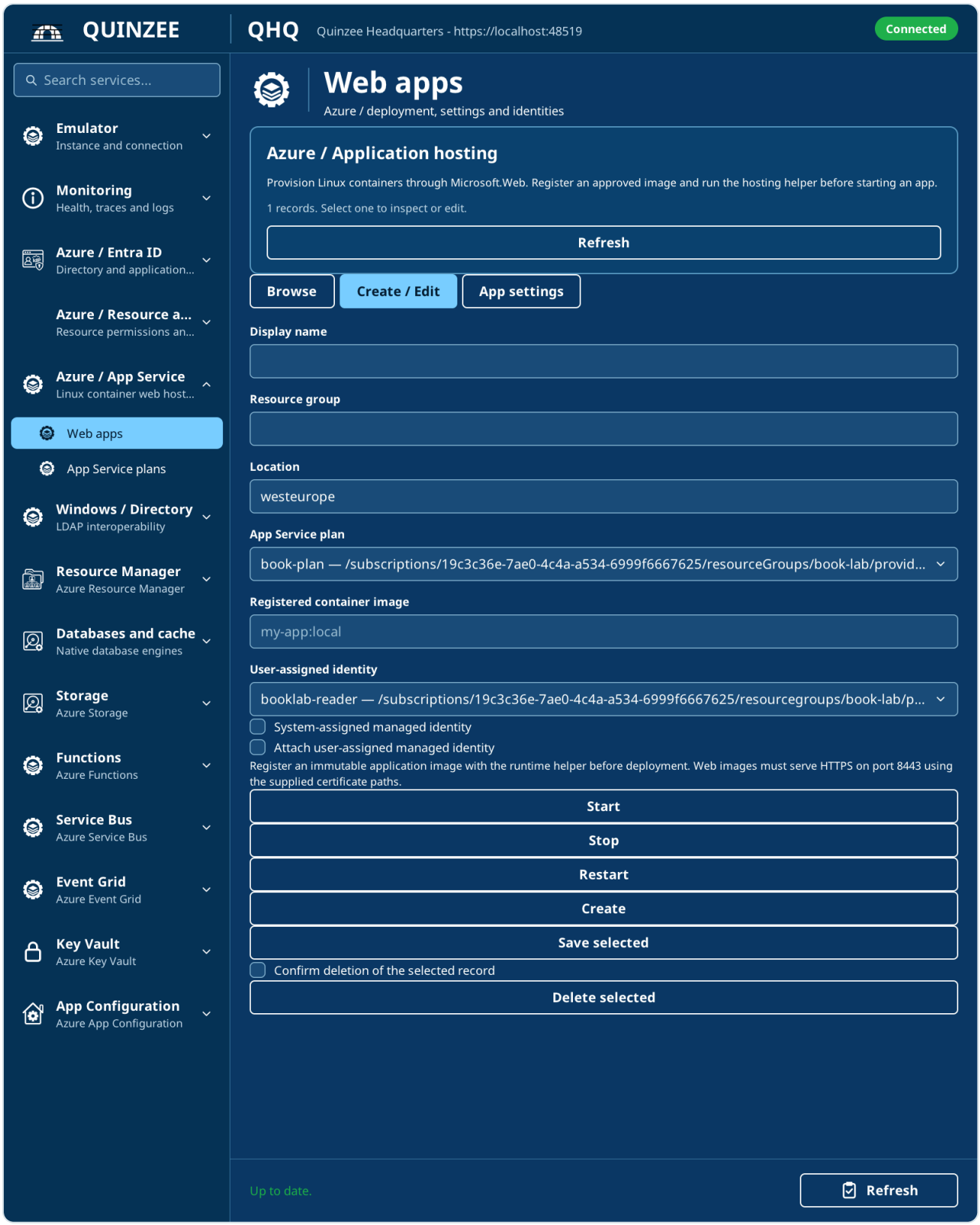


Web Apps

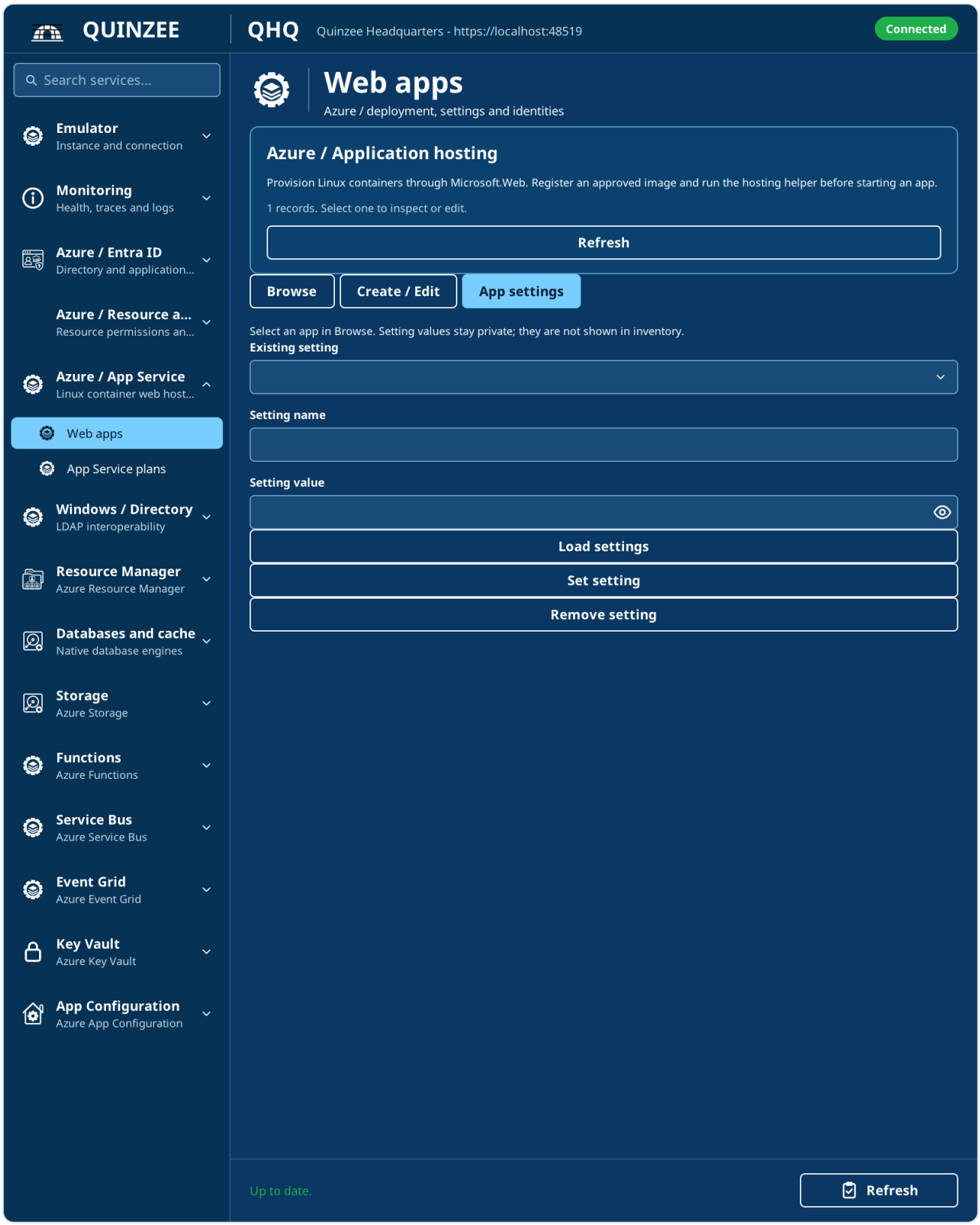
Browse



Create and edit

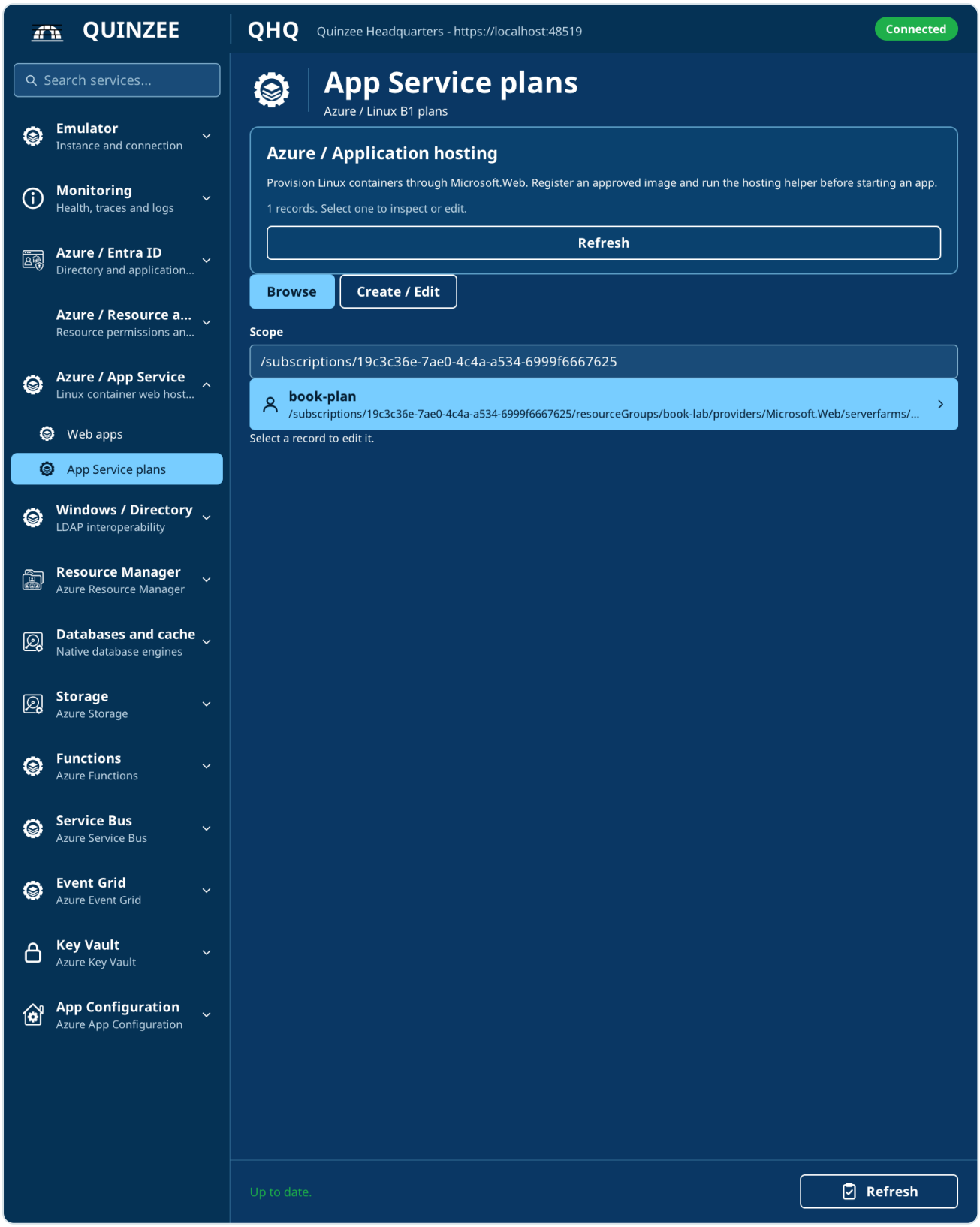


App settings

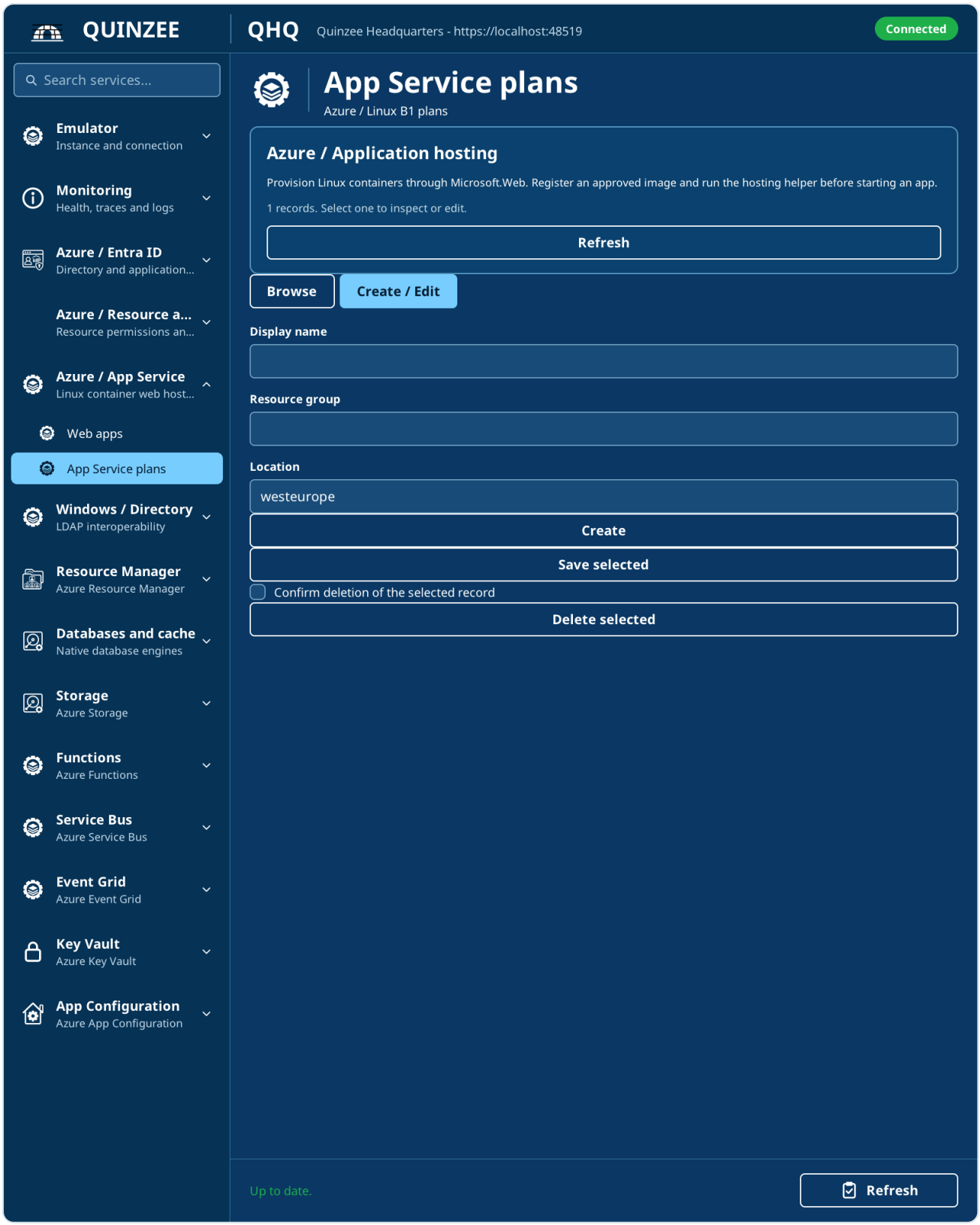


App Plans

Browse

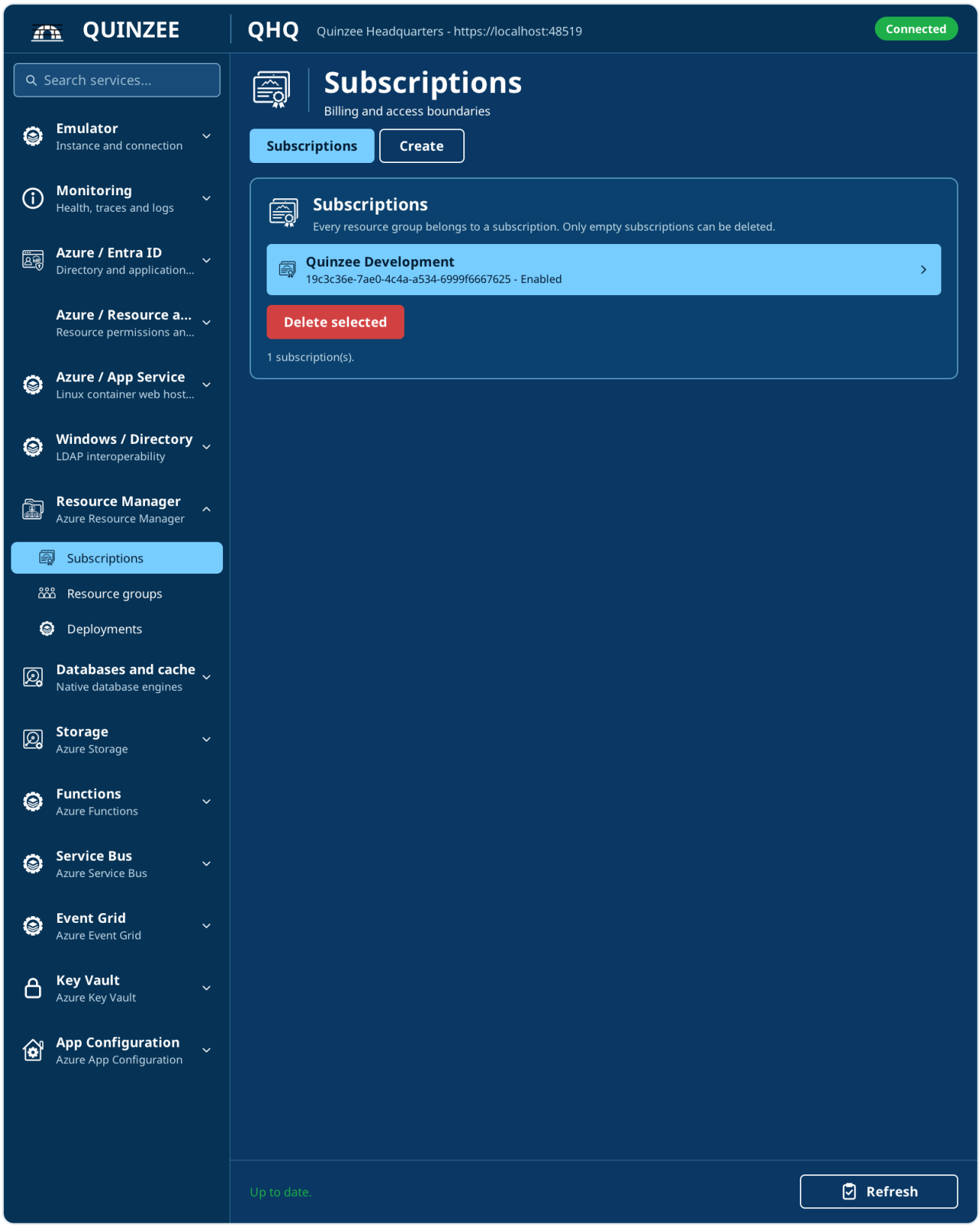


Create and edit

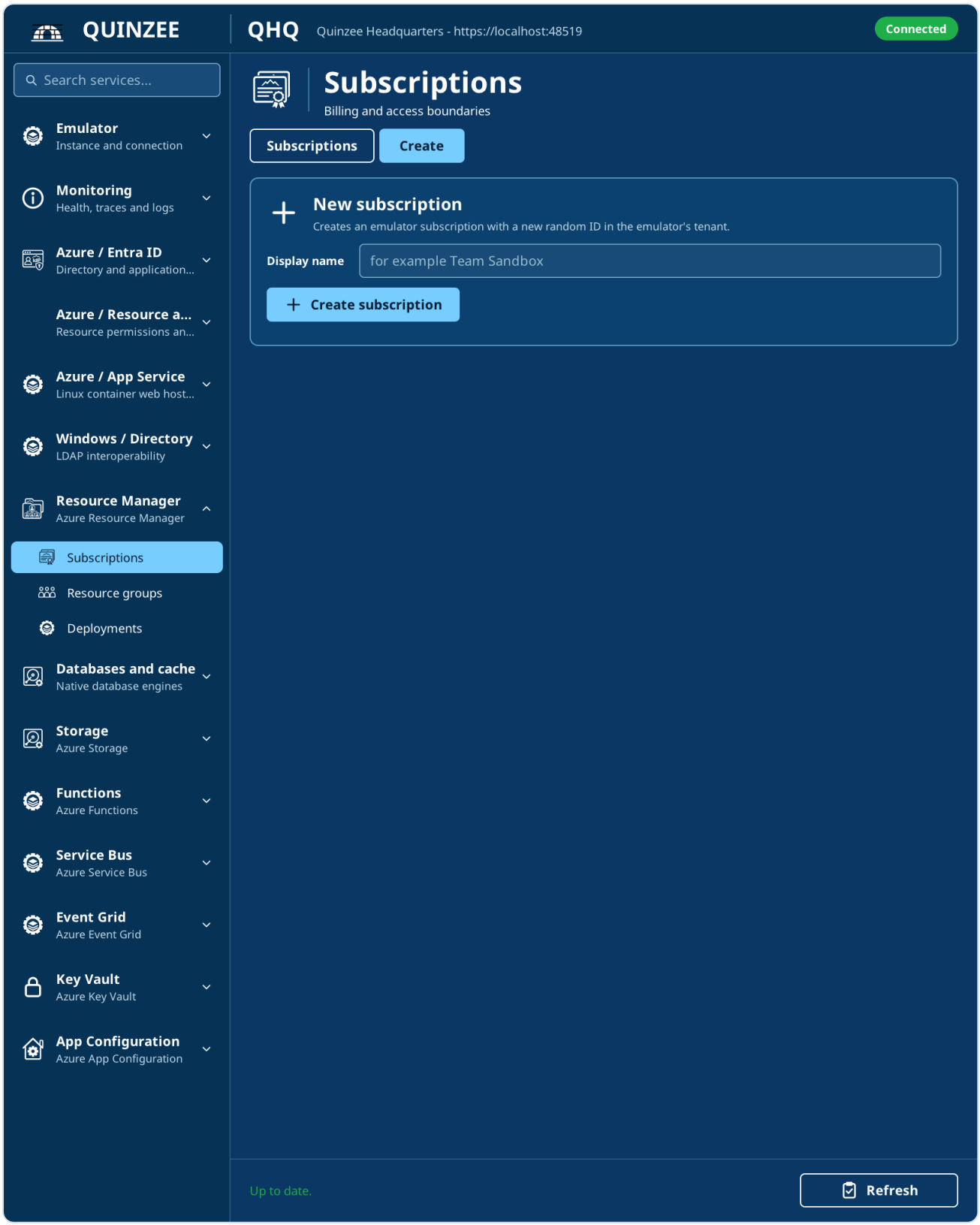


Subscriptions

Subscriptions

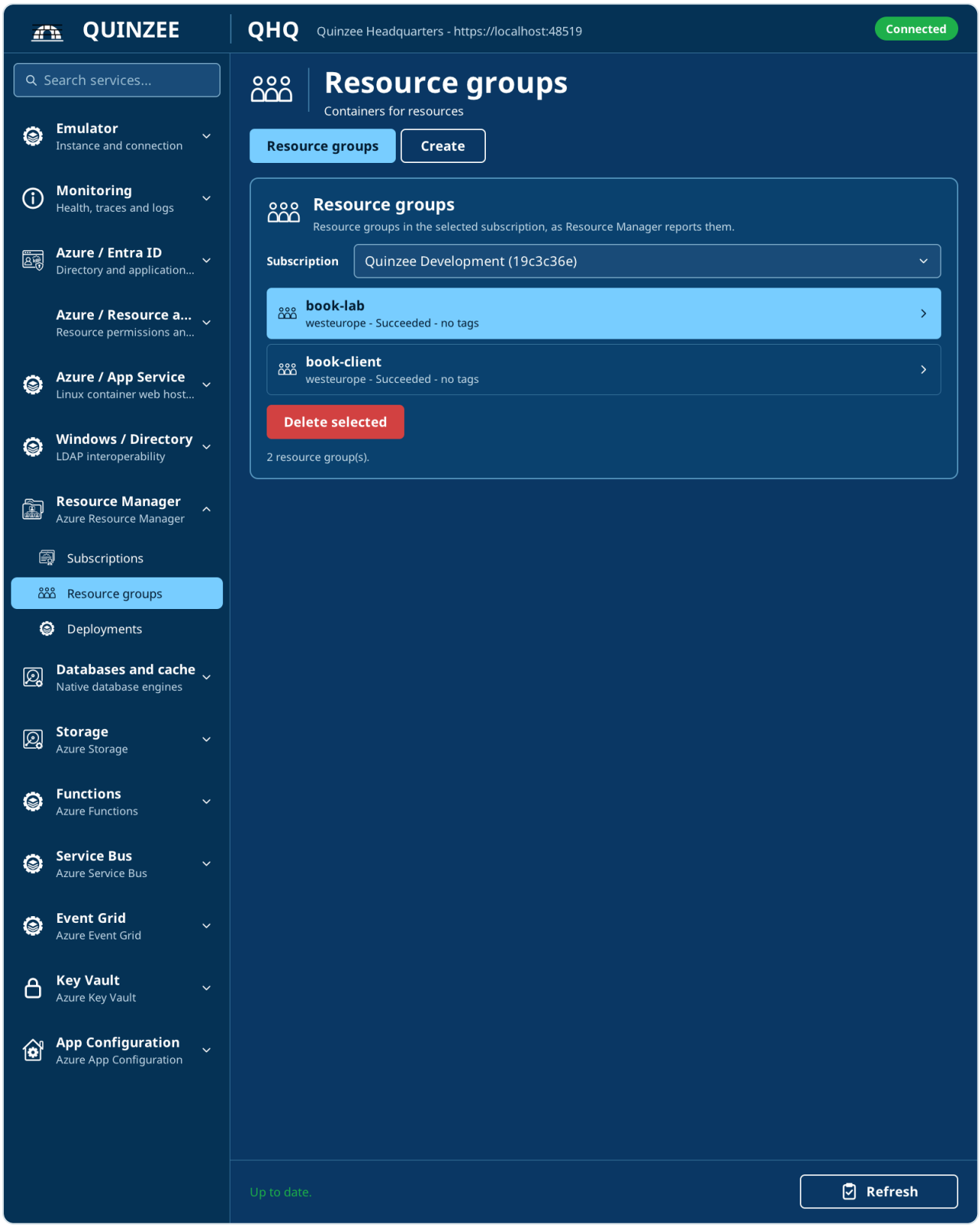


Create

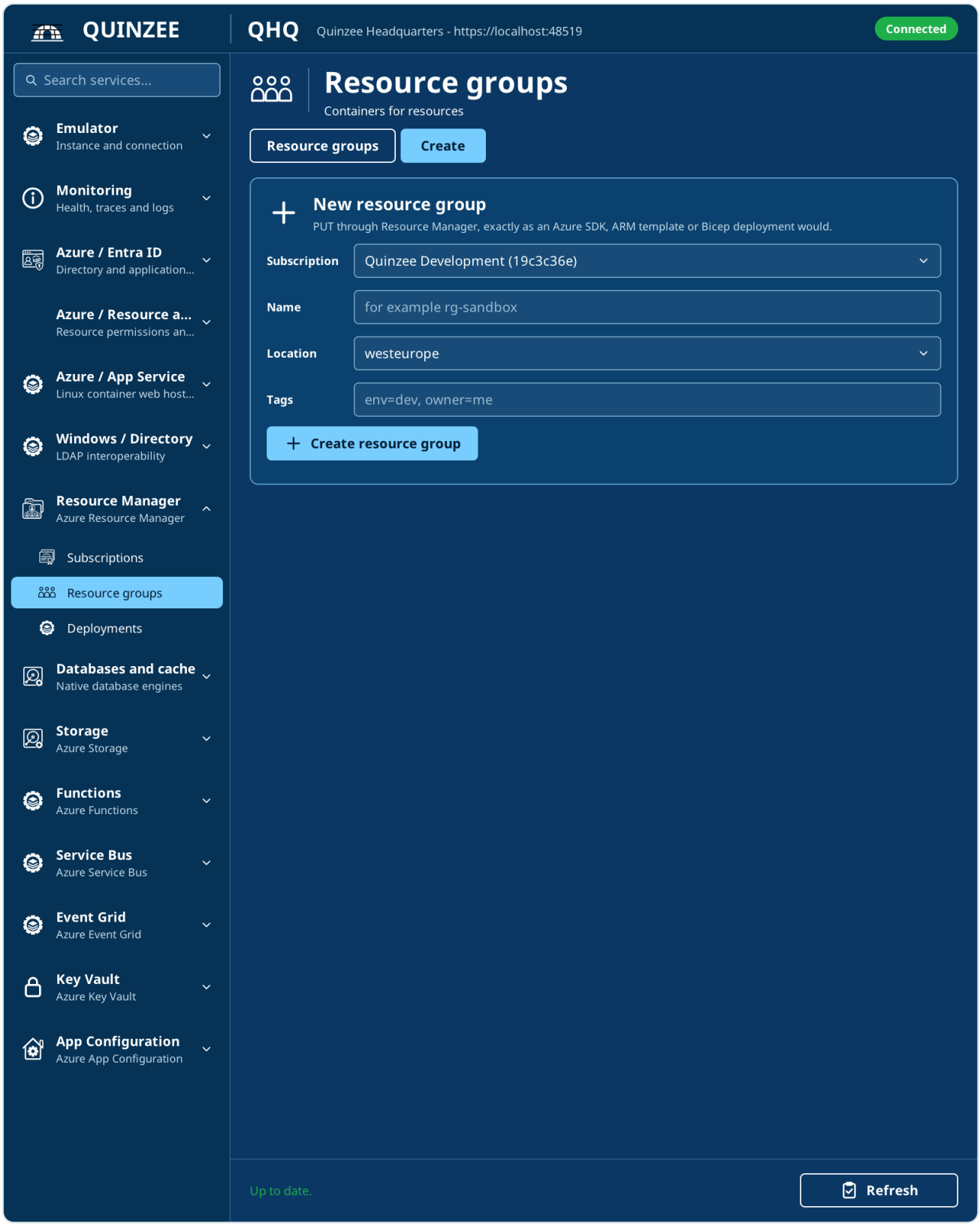

QUINZEE
QHQ
Quinzee Headquarters - https://localhost:48519
Connected
Search services...
Emulator
Instance and connection
Monitoring
Health, traces and logs
Azure / Entra ID
Directory and application...
Azure / Resource a...
Resource permissions an...
Azure / App Service
Linux container web host...
Windows / Directory
LDAP interoperability
Resource Manager
Azure Resource Manager
Subscriptions
Resource groups
Deployments
Databases and cache
Native database engines
Storage
Azure Storage
Functions
Azure Functions
Service Bus
Azure Service Bus
Event Grid
Azure Event Grid
Key Vault
Azure Key Vault
App Configuration
Azure App Configuration
Subscriptions
Billing and access boundaries
Subscriptions
Create
New subscription
Creates an emulator subscription with a new random ID in the emulator's tenant.
Display name
for example Team Sandbox
Create subscription
Up to date.
Refresh

Resource Groups

Resource groups

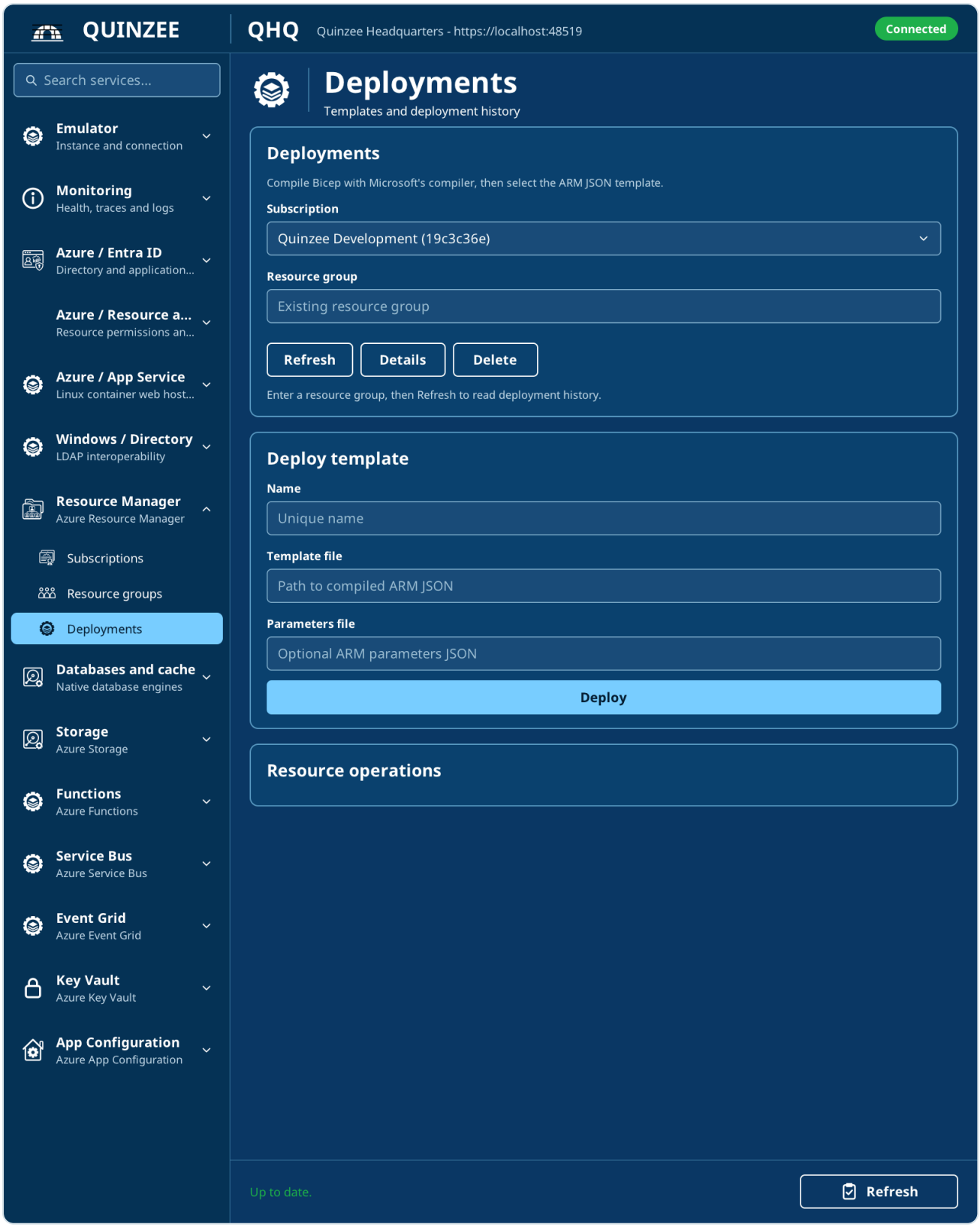


Create


QUINZEE
QHQ
Quinzee Headquarters - https://localhost:48519
Connected
Search services...
Emulator
Instance and connection
Monitoring
Health, traces and logs
Azure / Entra ID
Directory and application...
Azure / Resource a...
Resource permissions an...
Azure / App Service
Linux container web host...
Windows / Directory
LDAP interoperability
Resource Manager
Azure Resource Manager
Subscriptions
Resource groups
Deployments
Databases and cache
Native database engines
Storage
Azure Storage
Functions
Azure Functions
Service Bus
Azure Service Bus
Event Grid
Azure Event Grid
Key Vault
Azure Key Vault
App Configuration
Azure App Configuration
Resource groups
Containers for resources
Resource groups
Create
New resource group
PUT through Resource Manager, exactly as an Azure SDK, ARM template or Bicep deployment would.
Subscription
Quinzee Development (19c3c36e)
Name
for example rg-sandbox
Location
westeurope
Tags
env=dev, owner=me
Create resource group
Up to date.
Refresh

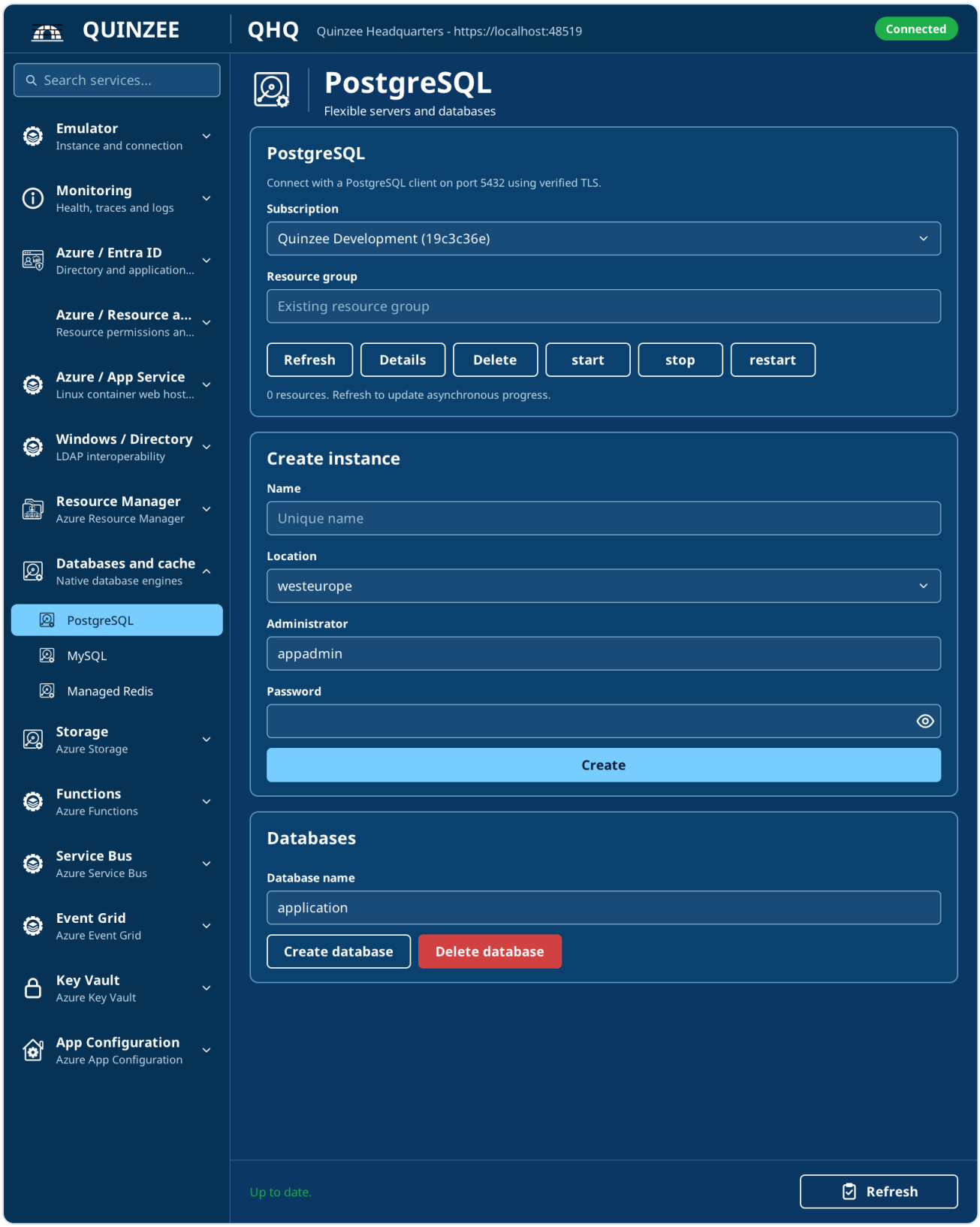
Deployments

Deployment operations



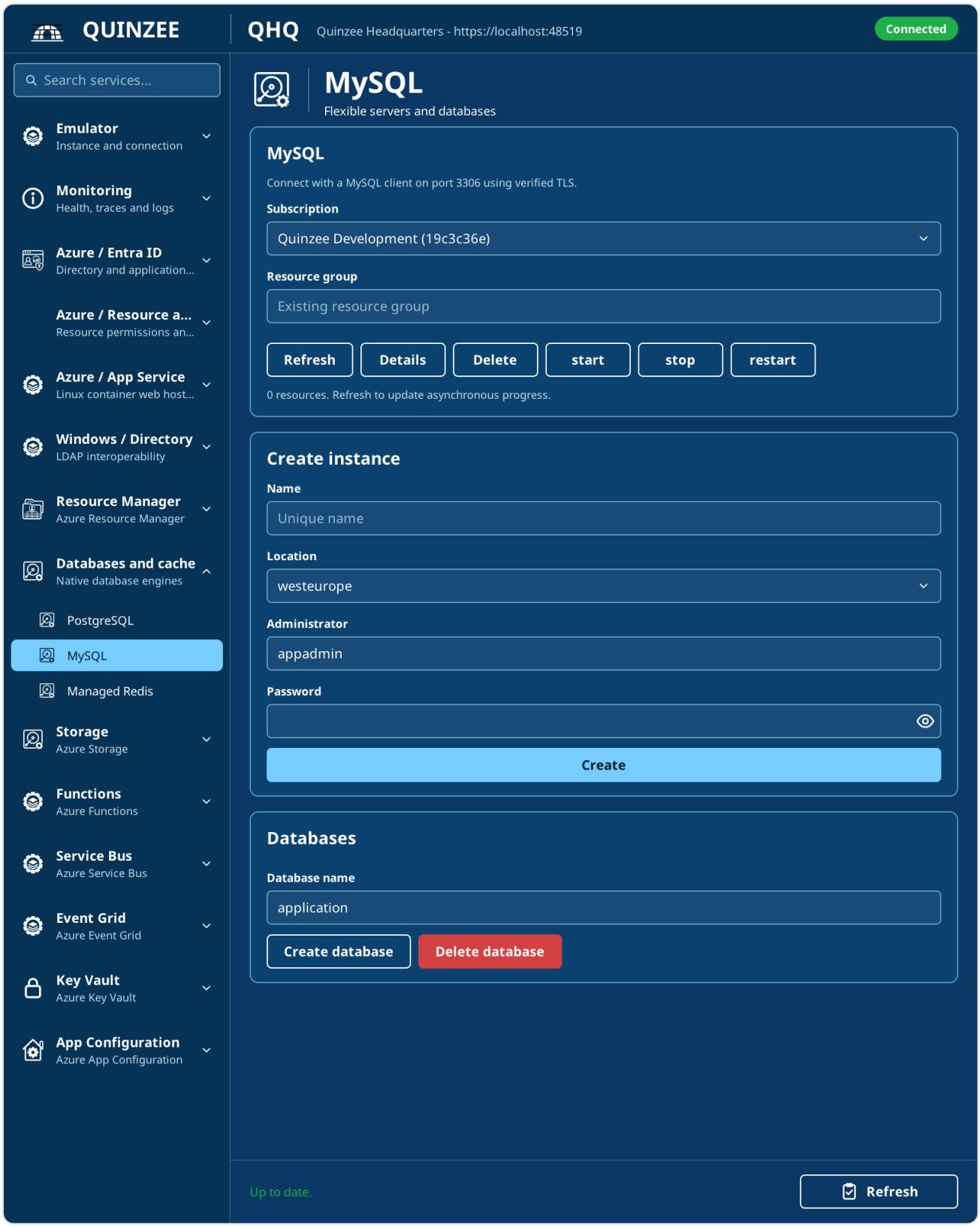
Postgresql

Servers and databases



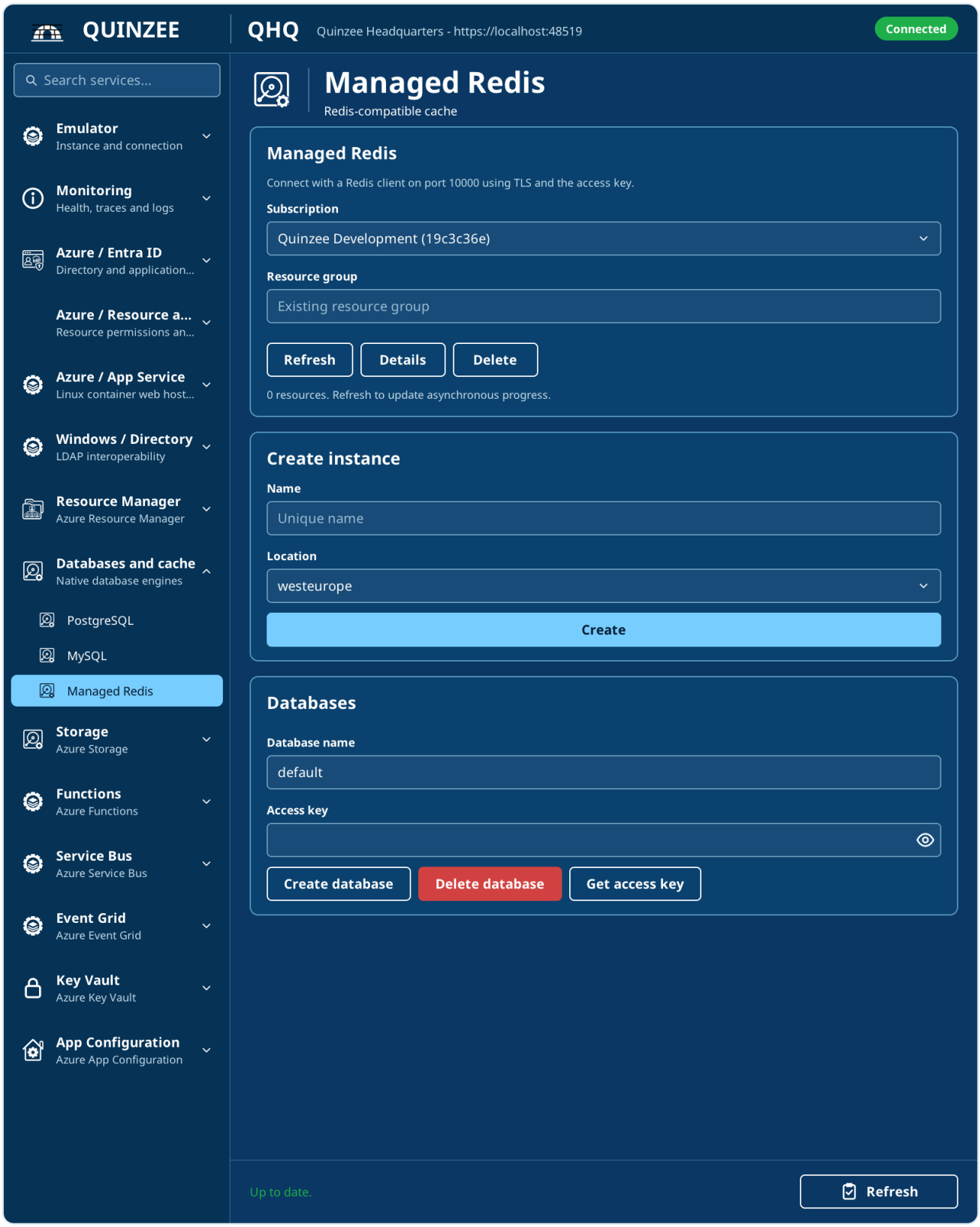
Mysql

Servers and databases



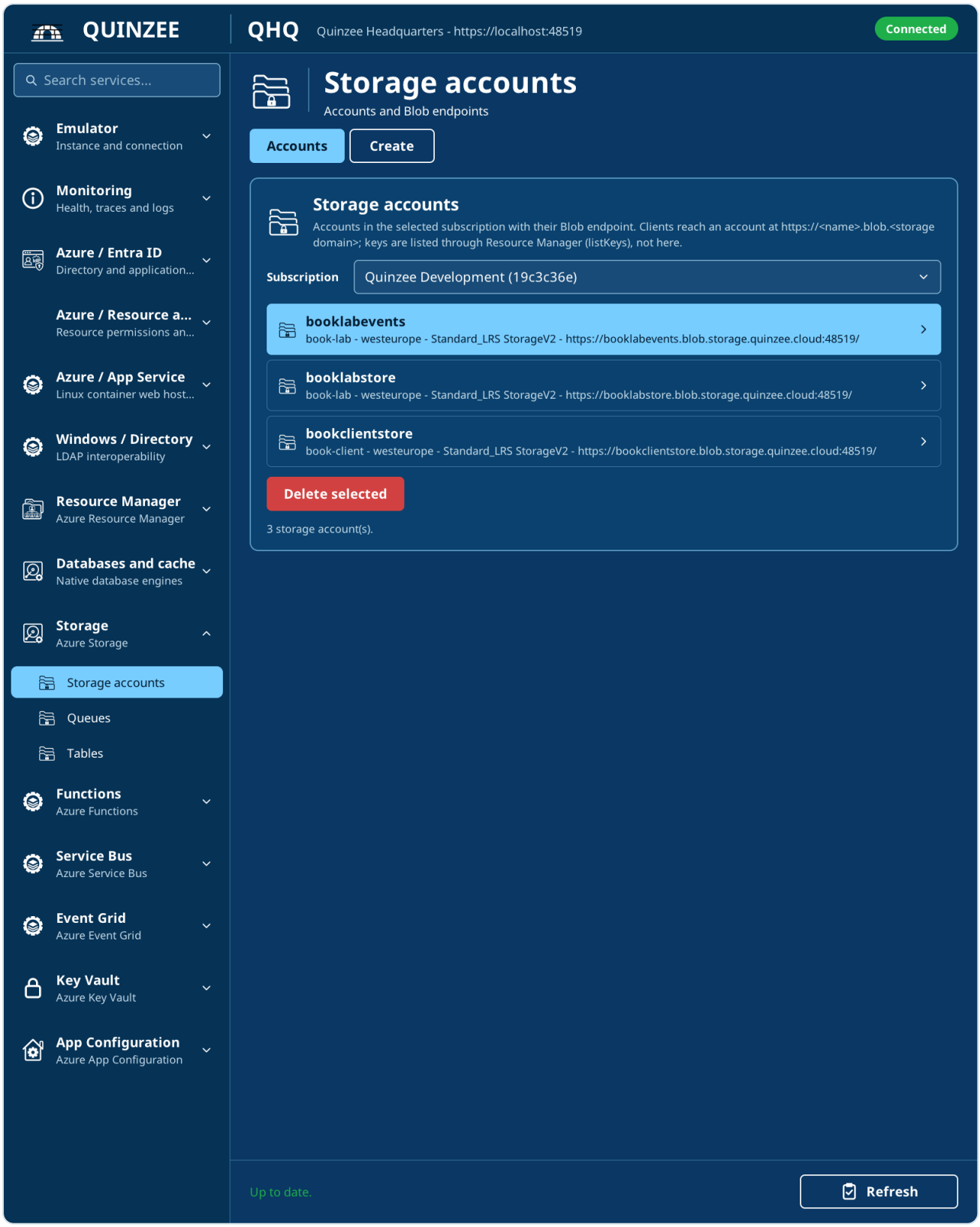
Managed Redis

Caches and database

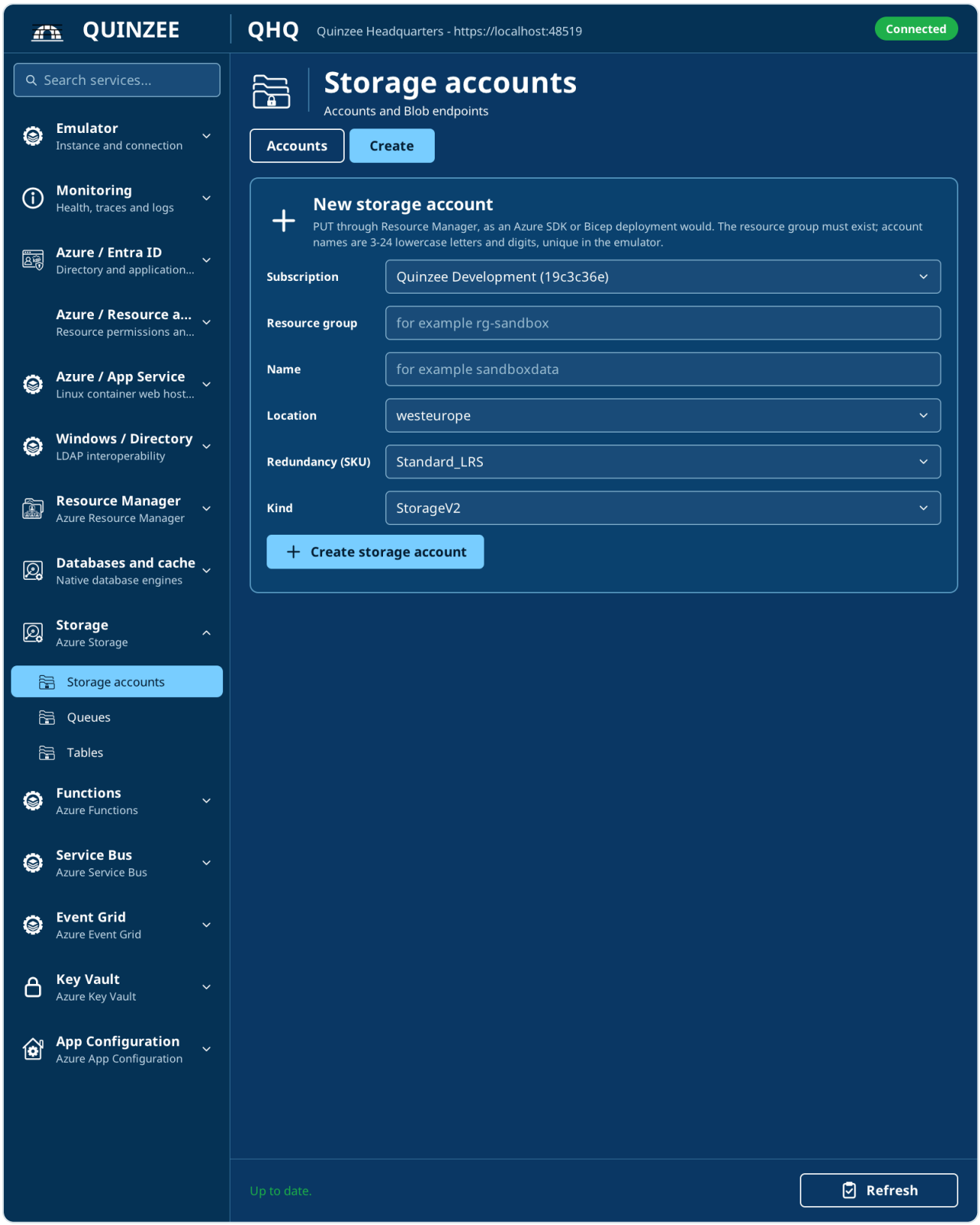


Storage Accounts

Accounts

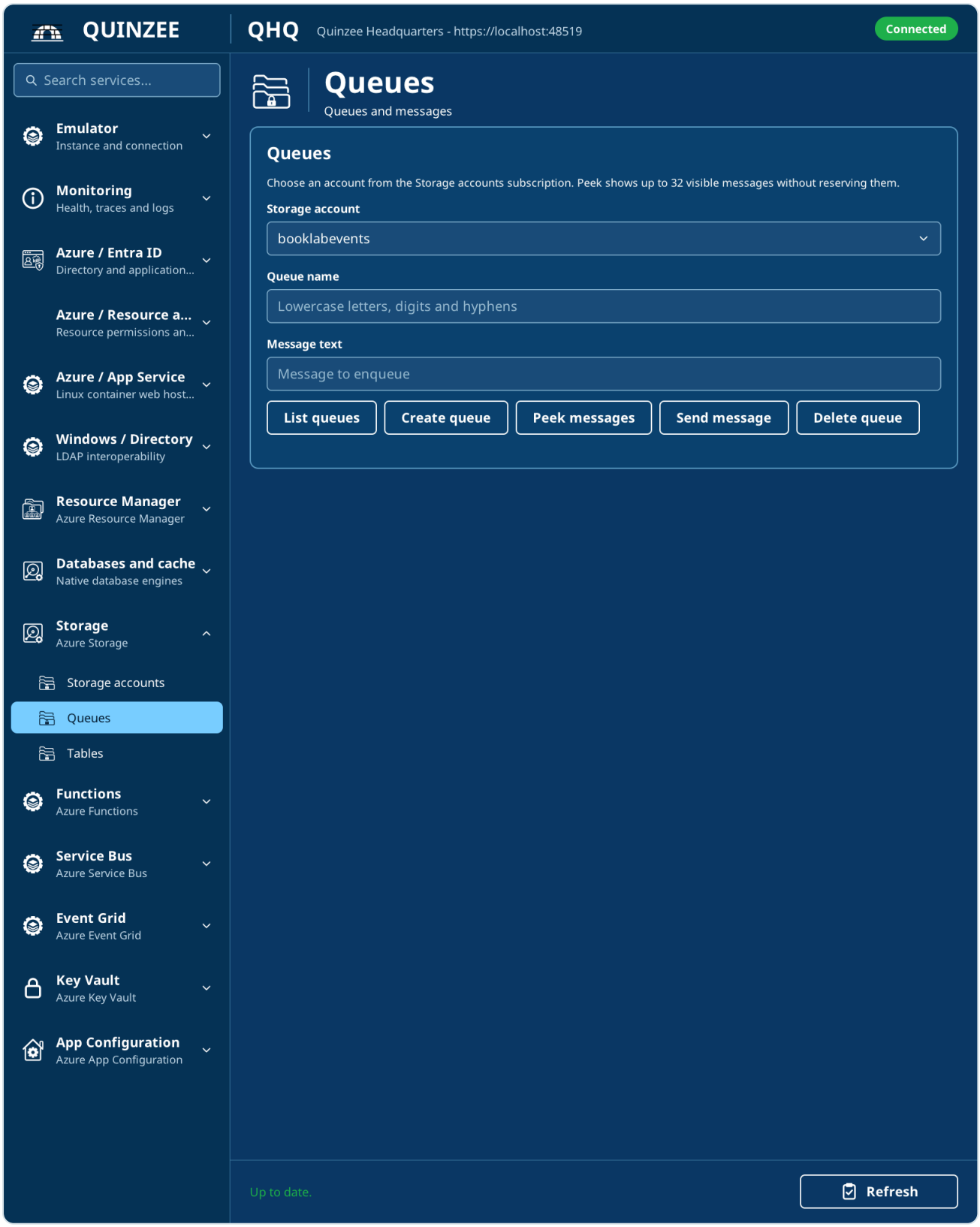


Create


QUINZEE
QHQ
Quinzee Headquarters - https://localhost:48519
Connected
Search services...
Emulator
Instance and connection
Monitoring
Health, traces and logs
Azure / Entra ID
Directory and application...
Azure / Resource a...
Resource permissions an...
Azure / App Service
Linux container web host...
Windows / Directory
LDAP interoperability
Resource Manager
Azure Resource Manager
Databases and cache
Native database engines
Storage
Azure Storage
Storage accounts
Queues
Tables
Functions
Azure Functions
Service Bus
Azure Service Bus
Event Grid
Azure Event Grid
Key Vault
Azure Key Vault
App Configuration
Azure App Configuration
Storage accounts
Accounts and Blob endpoints
Accounts
Create
New storage account
PUT through Resource Manager, as an Azure SDK or Bicep deployment would. The resource group must exist; account names are 3-24 lowercase letters and digits, unique in the emulator.
Subscription
Quinzee Development (19c3c36e)
Resource group
for example rg-sandbox
Name
for example sandboxdata
Location
westeurope
Redundancy (SKU)
Standard_LRS
Kind
StorageV2
Create storage account
Up to date.
Refresh

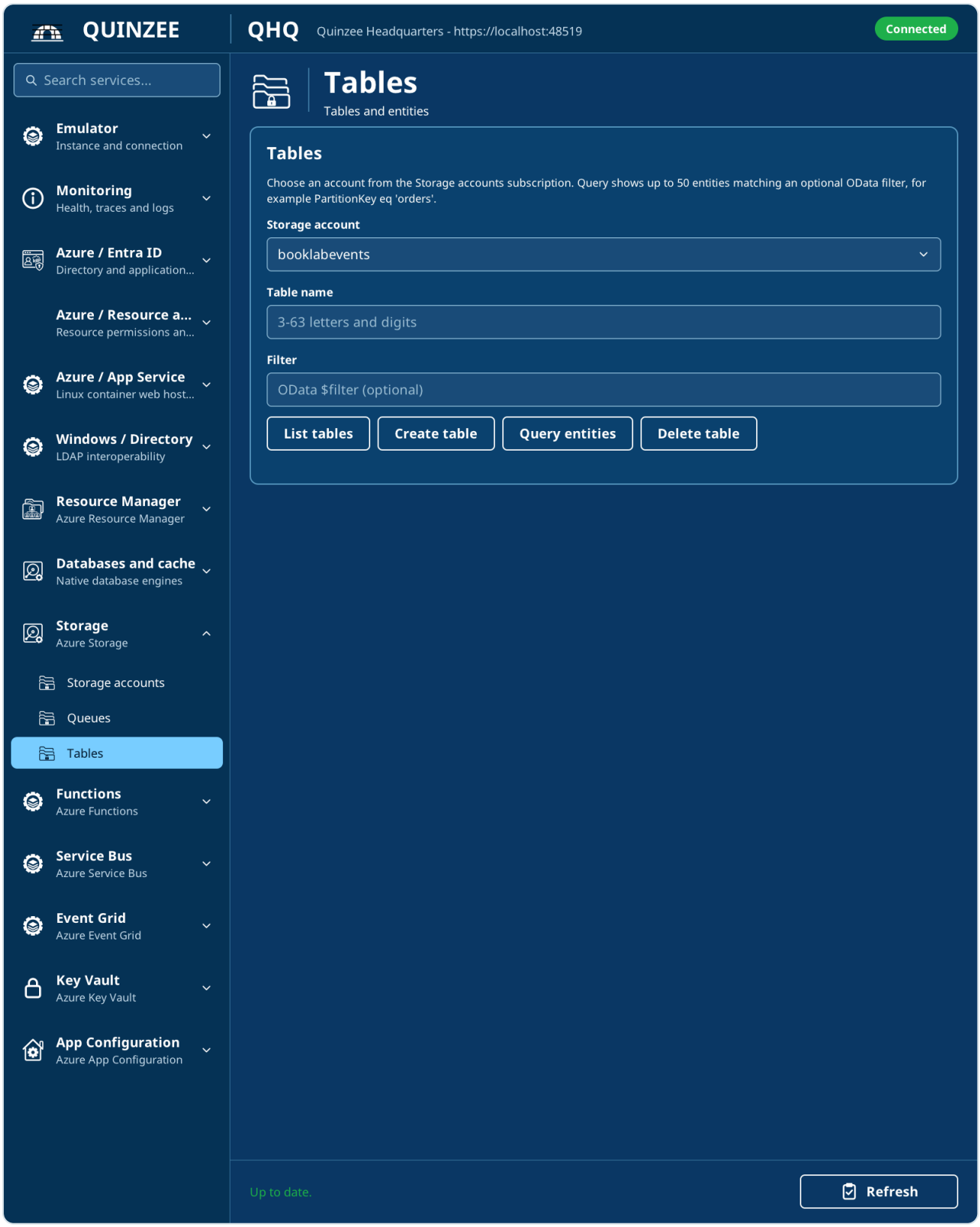
Queues

Queues and messages



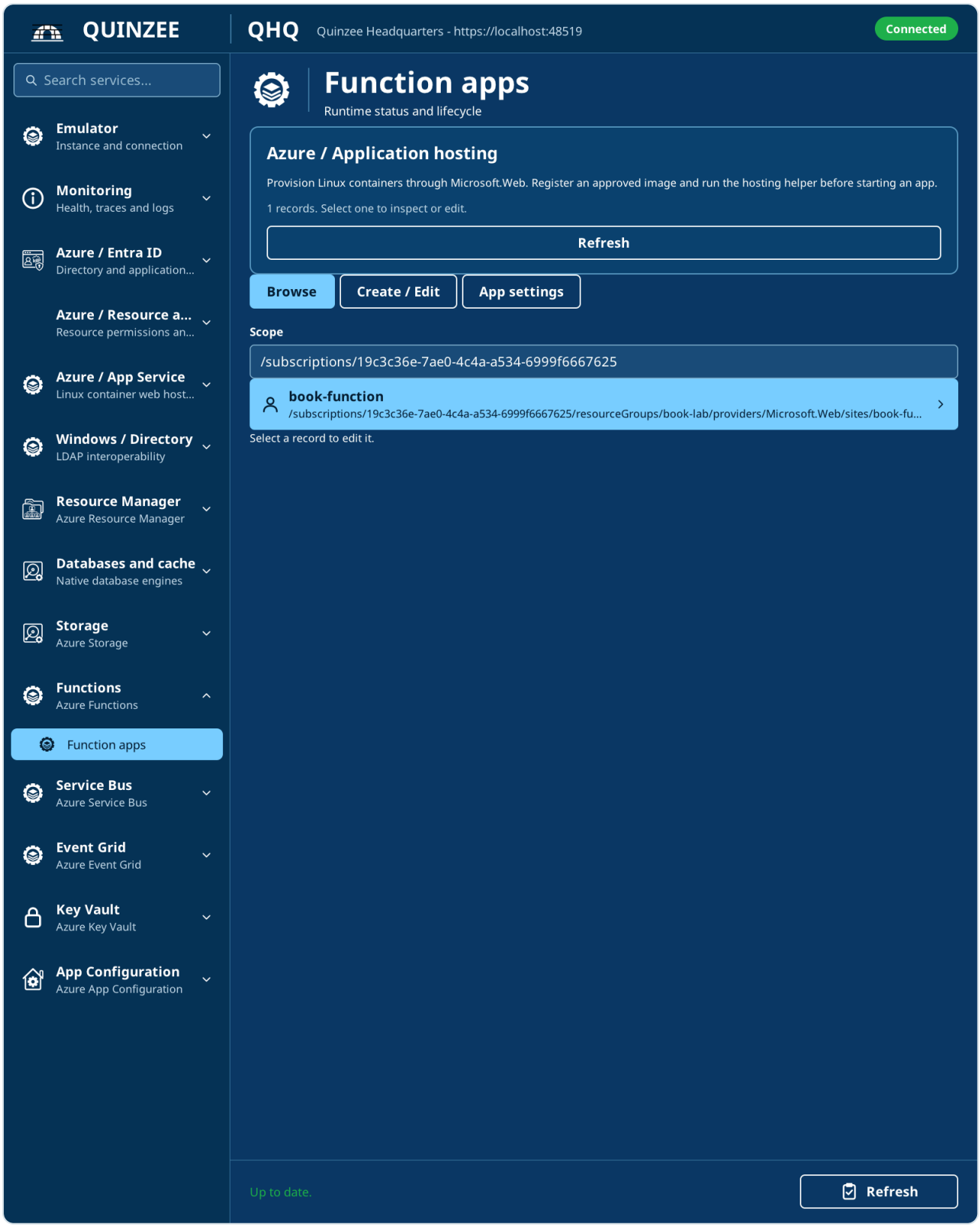
Tables

Tables and entities

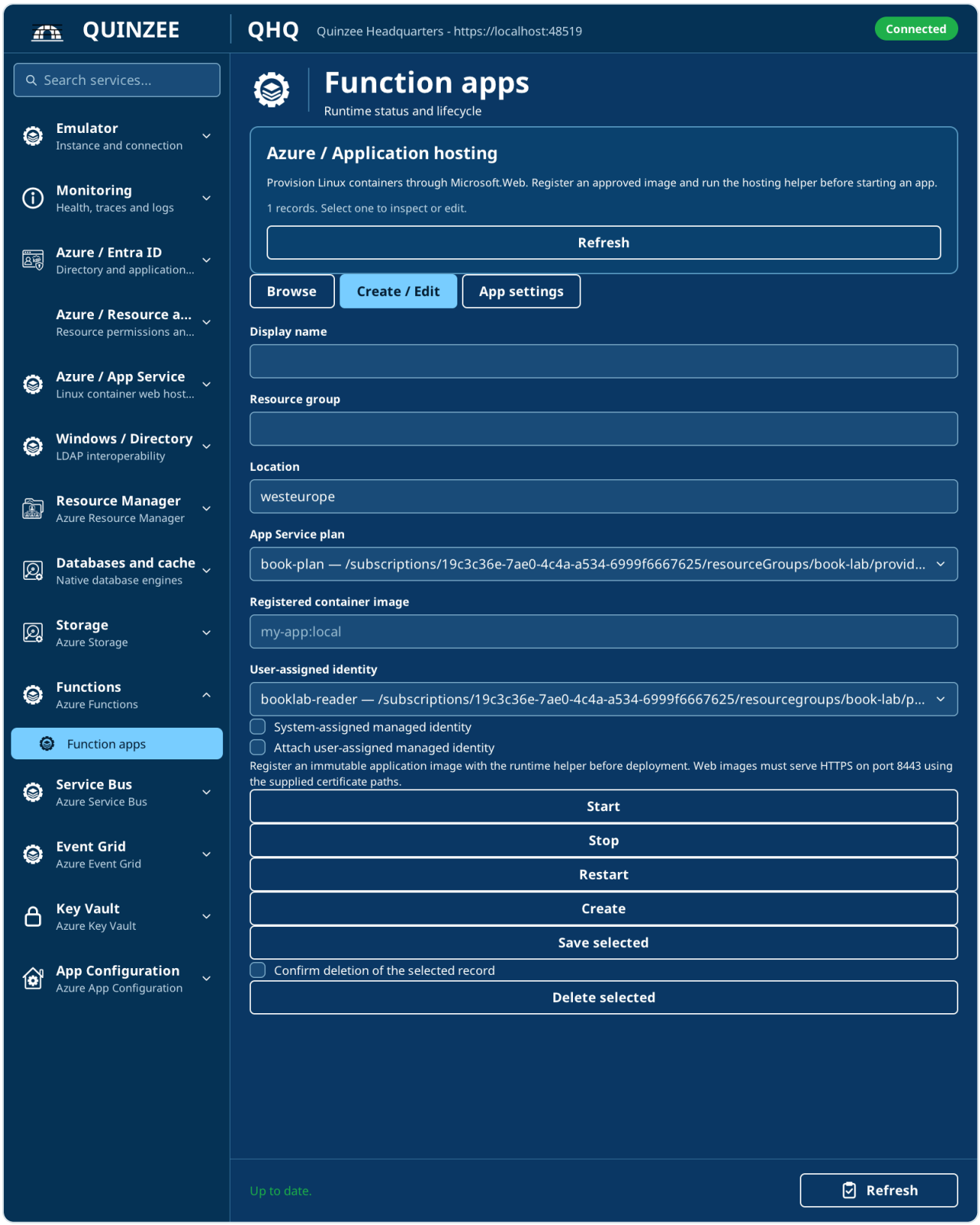


Function Apps

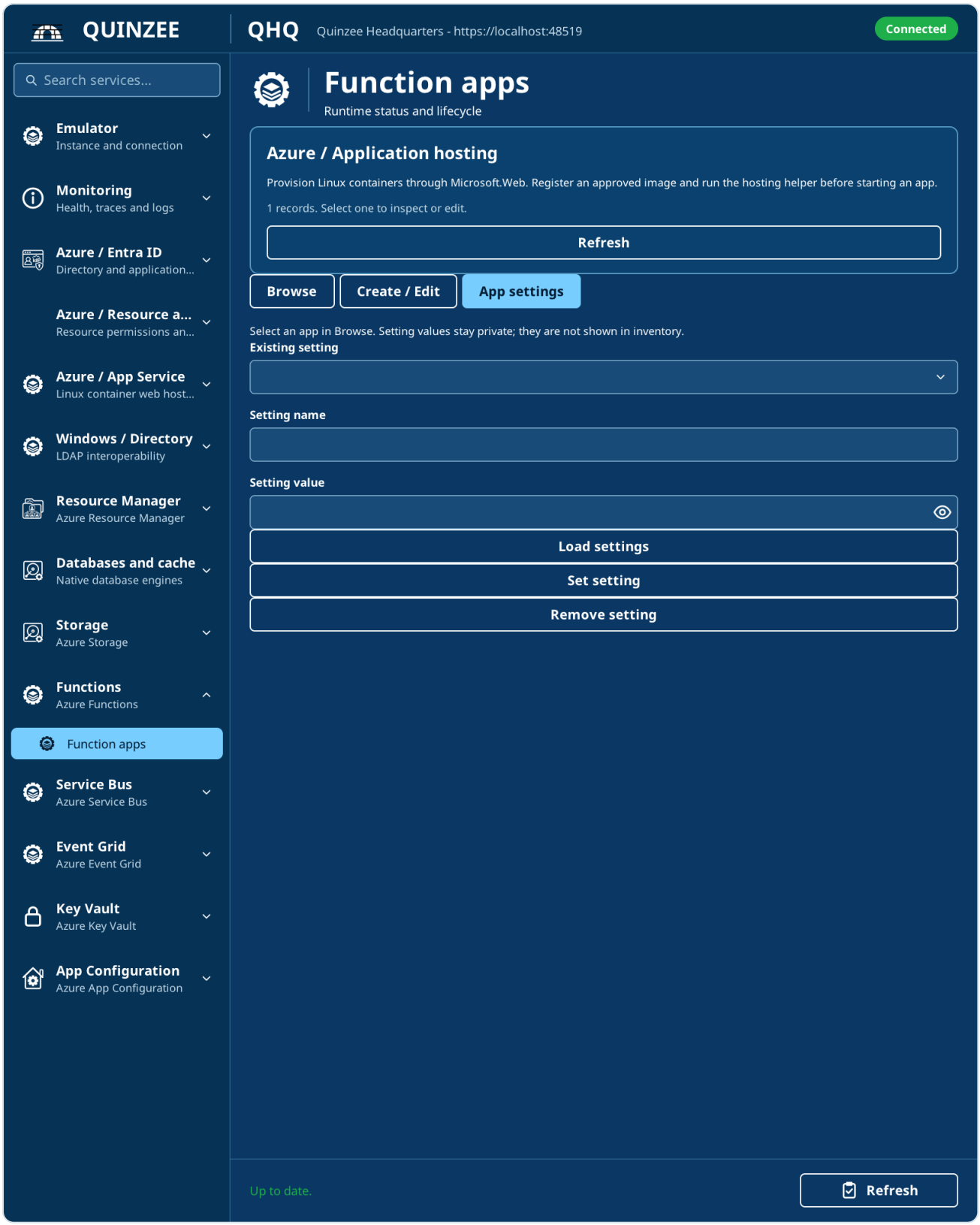
Browse



Create and edit

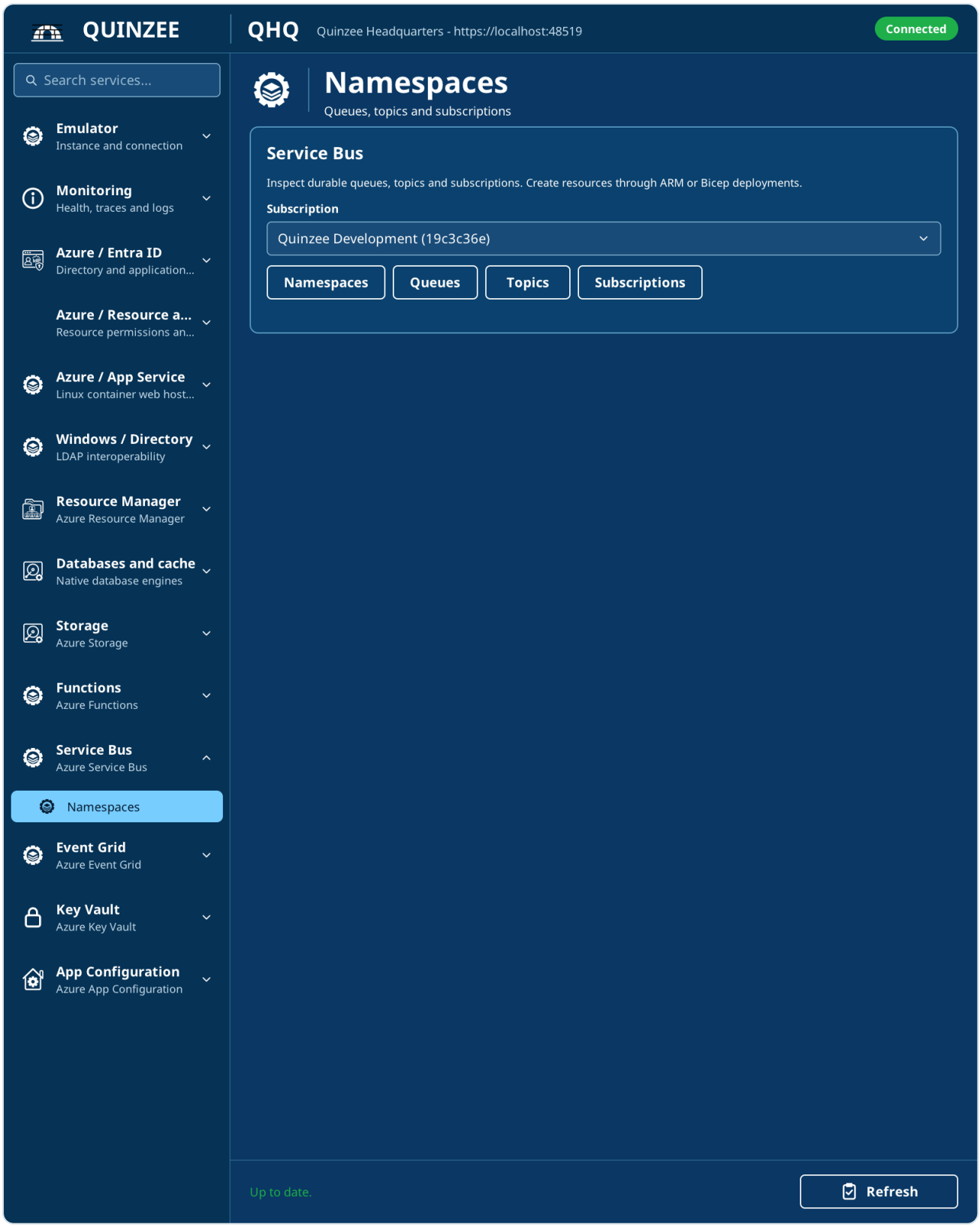


App settings



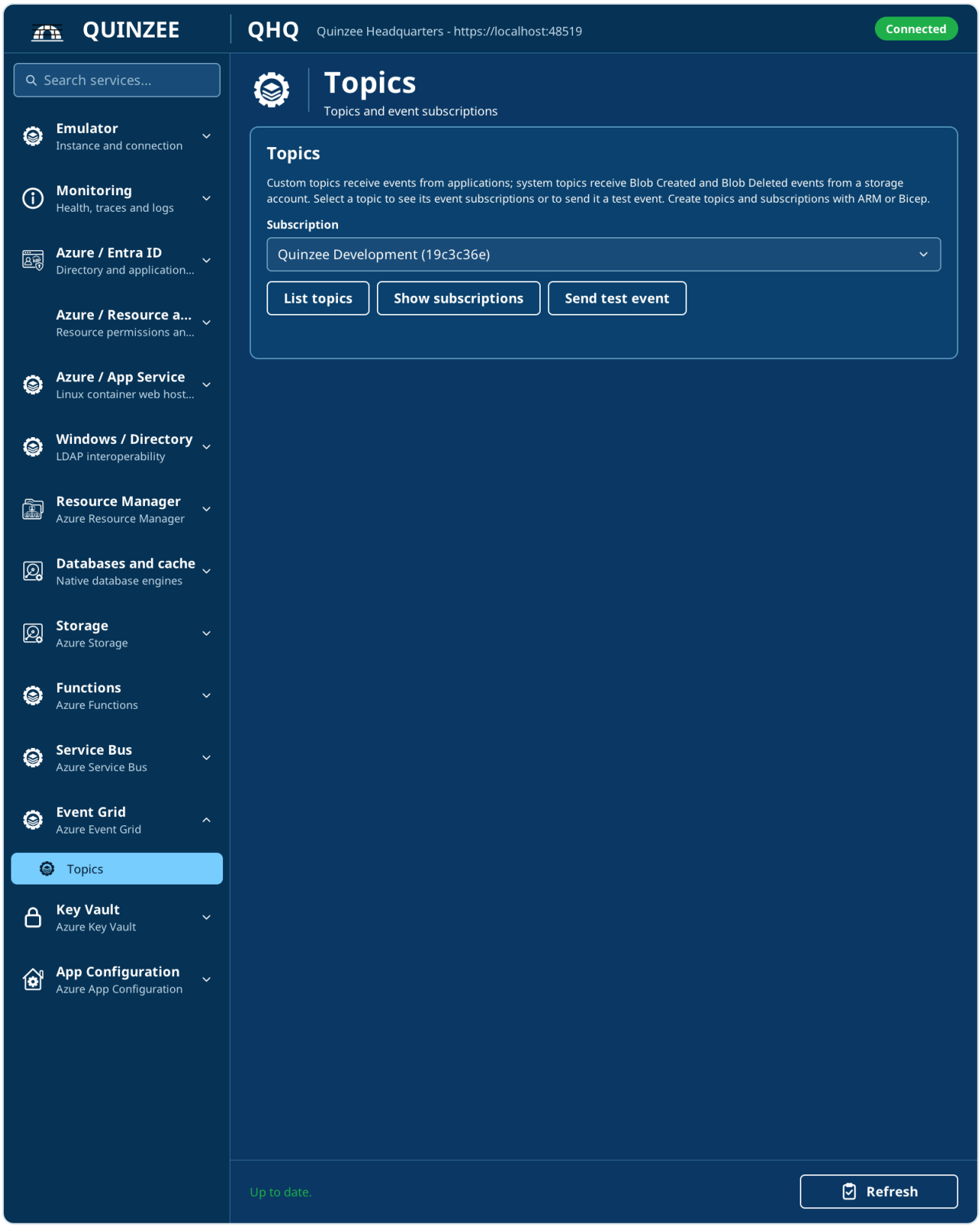
Bus Namespaces

Namespaces and entities



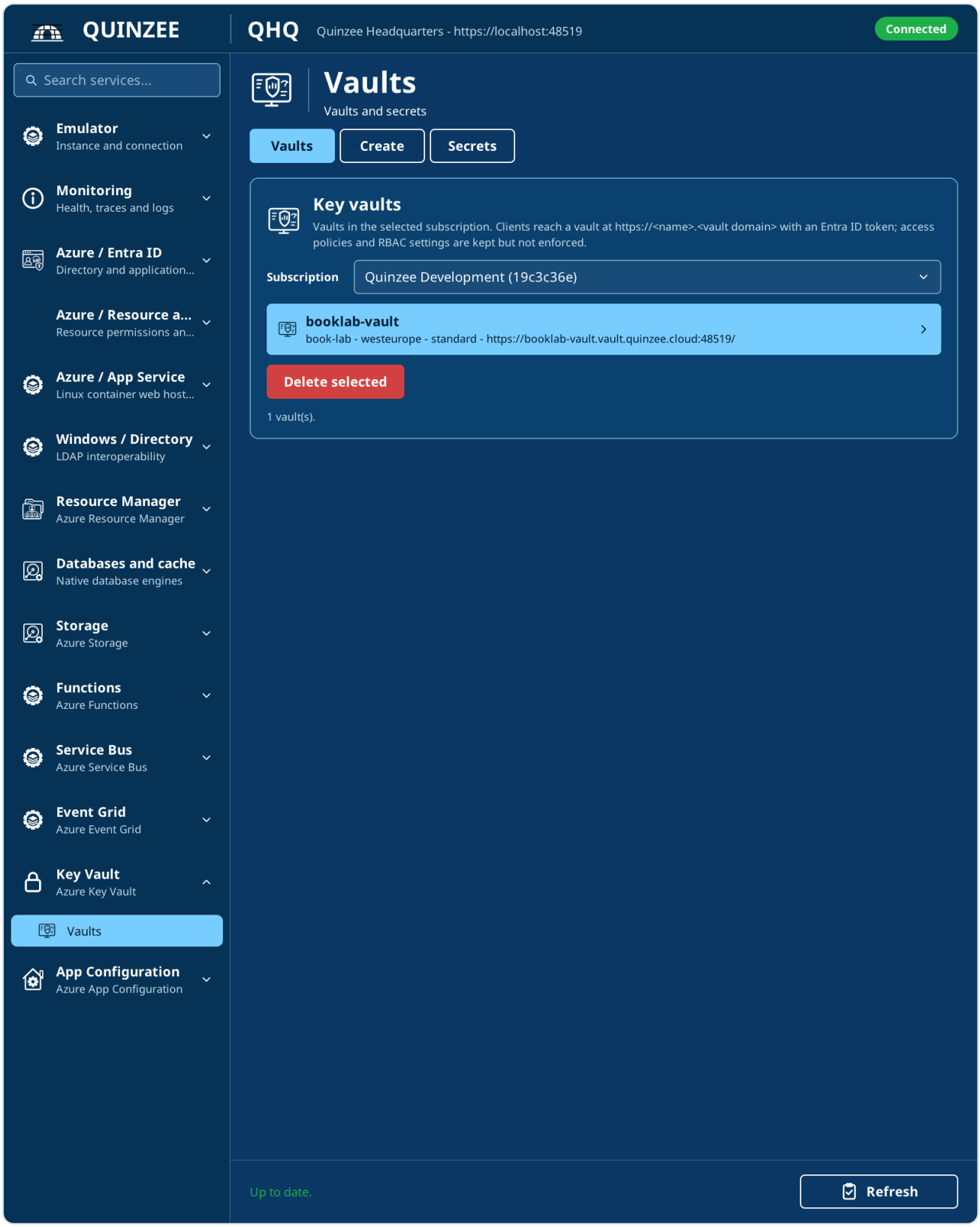
Topics

Topics and subscriptions

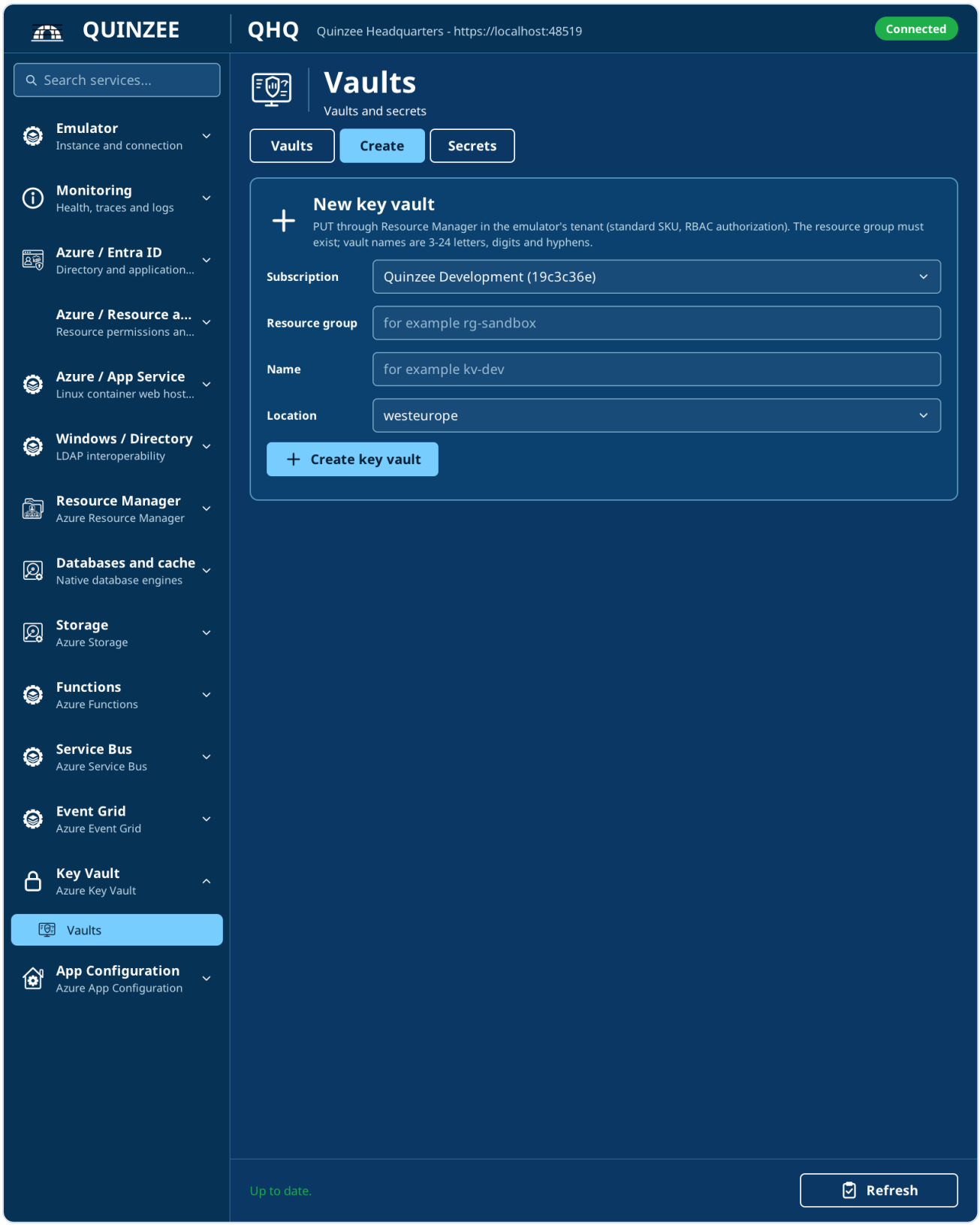


Vaults

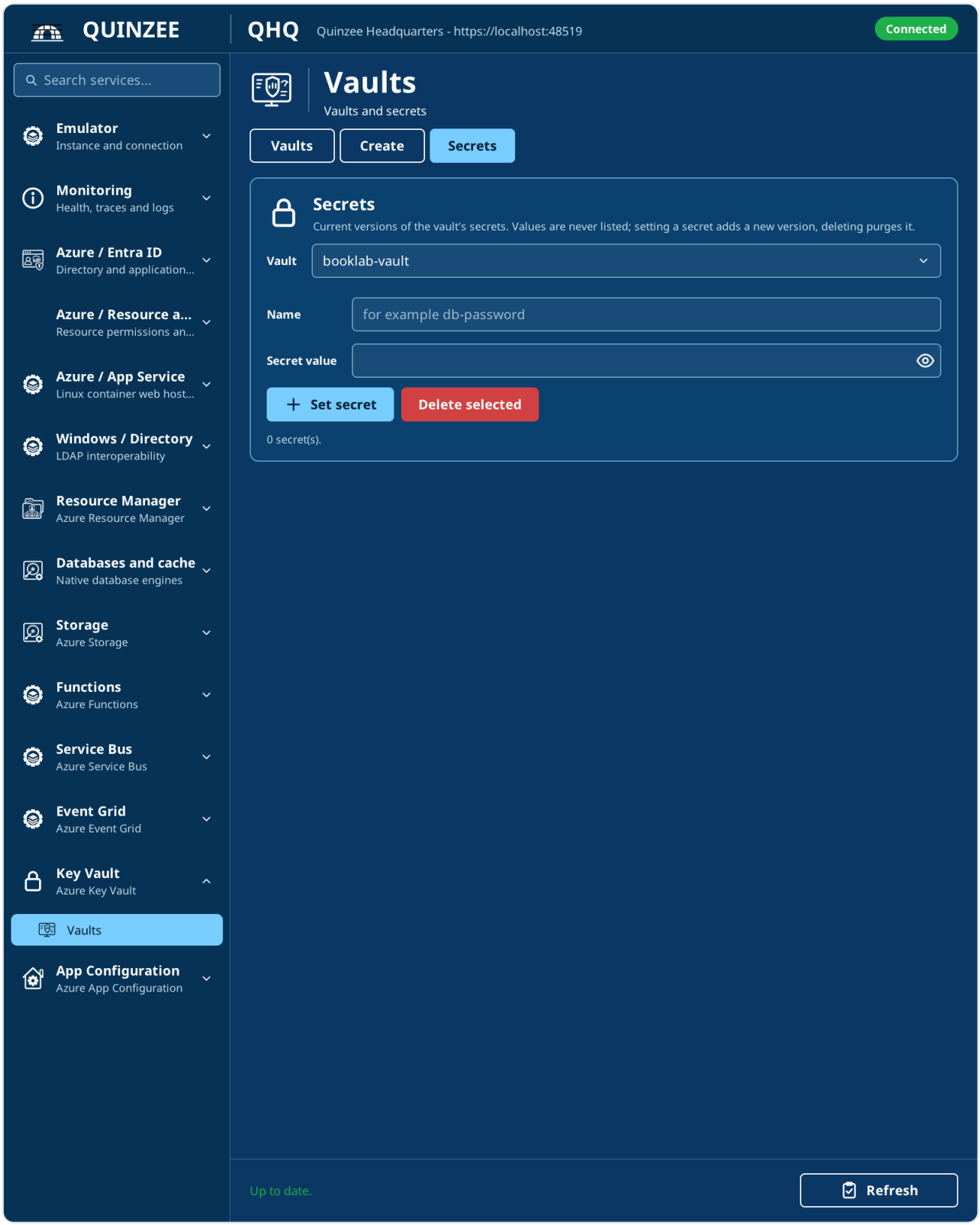
Vaults



Create

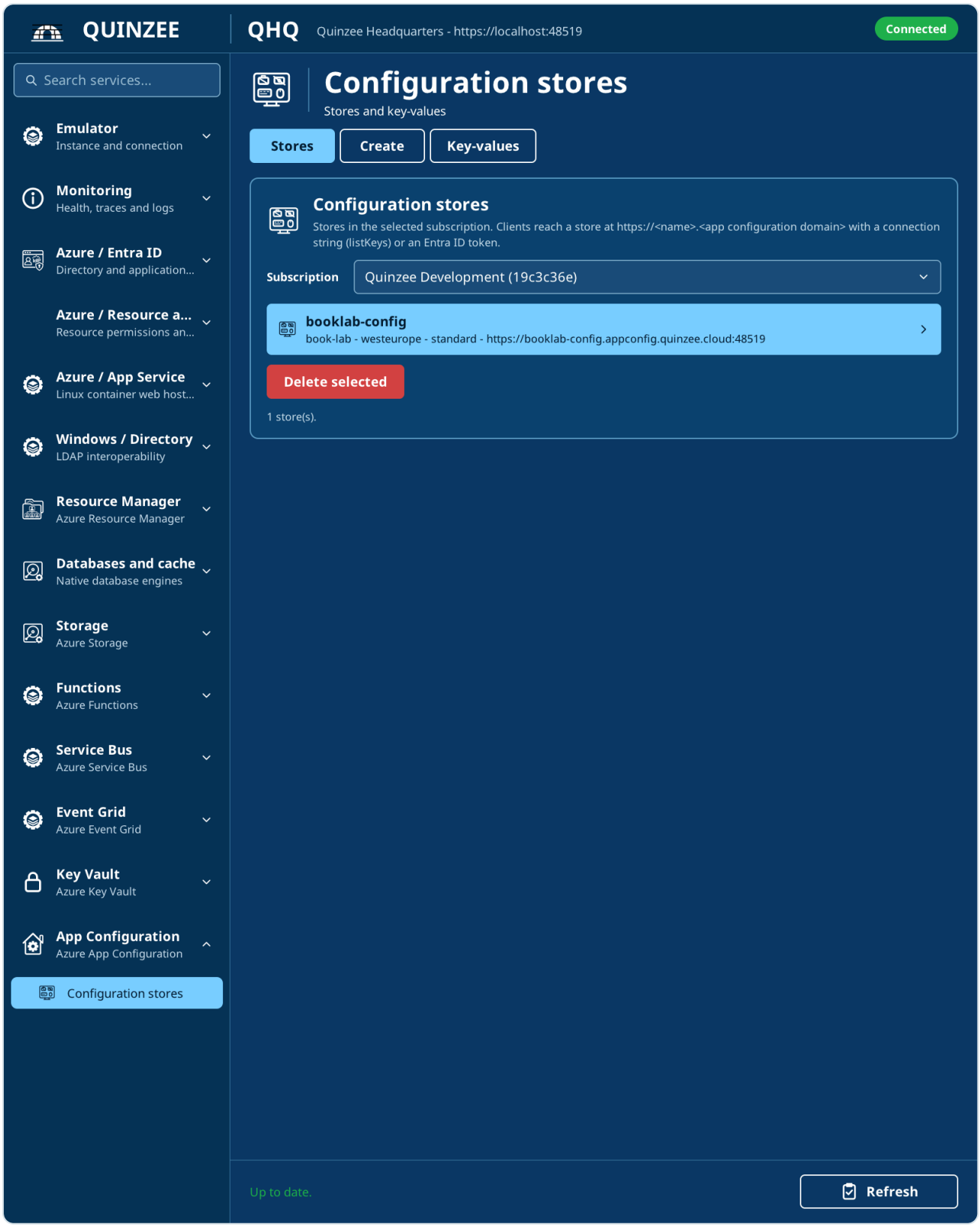

QUINZEE
QHQ
Quinzee Headquarters - https://localhost:48519
Connected
Search services...
Emulator
Instance and connection
Monitoring
Health, traces and logs
Azure / Entra ID
Directory and application...
Azure / Resource a...
Resource permissions an...
Azure / App Service
Linux container web host...
Windows / Directory
LDAP interoperability
Resource Manager
Azure Resource Manager
Databases and cache
Native database engines
Storage
Azure Storage
Functions
Azure Functions
Service Bus
Azure Service Bus
Event Grid
Azure Event Grid
Key Vault
Azure Key Vault
Vaults
App Configuration
Azure App Configuration
Vaults
Vaults and secrets
Vaults
Create
Secrets
New key vault
PUT through Resource Manager in the emulator's tenant (standard SKU, RBAC authorization). The resource group must exist; vault names are 3-24 letters, digits and hyphens.
Subscription
Quinzee Development (19c3c36e)
Resource group
for example rg-sandbox
Name
for example kv-dev
Location
westeurope
Create key vault
Up to date.
Refresh

Secrets

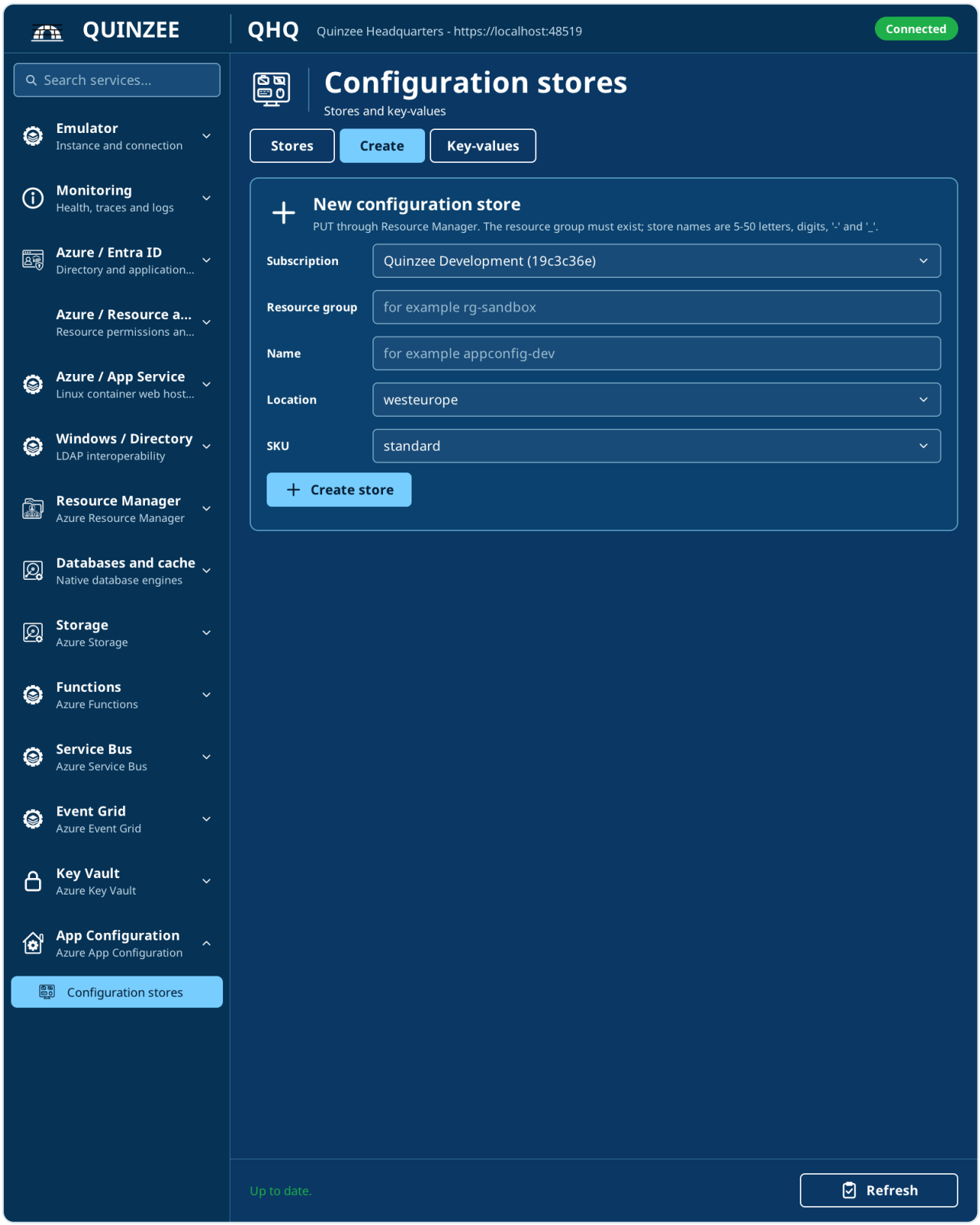

QUINZEE
QHQ
Quinzee Headquarters - https://localhost:48519
Connected
Search services...
Emulator
Instance and connection
Monitoring
Health, traces and logs
Azure / Entra ID
Directory and application...
Azure / Resource a...
Resource permissions an...
Azure / App Service
Linux container web host...
Windows / Directory
LDAP interoperability
Resource Manager
Azure Resource Manager
Databases and cache
Native database engines
Storage
Azure Storage
Functions
Azure Functions
Service Bus
Azure Service Bus
Event Grid
Azure Event Grid
Key Vault
Azure Key Vault
Vaults
App Configuration
Azure App Configuration
Vaults
Vaults and secrets
Vaults
Create
Secrets
Secrets
Current versions of the vault's secrets. Values are never listed; setting a secret adds a new version, deleting purges it.
Vault
booklab-vault
Name
for example db-password
Secret value
Set secret
Delete selected
0 secret(s).
Up to date.
Refresh

Configuration Stores

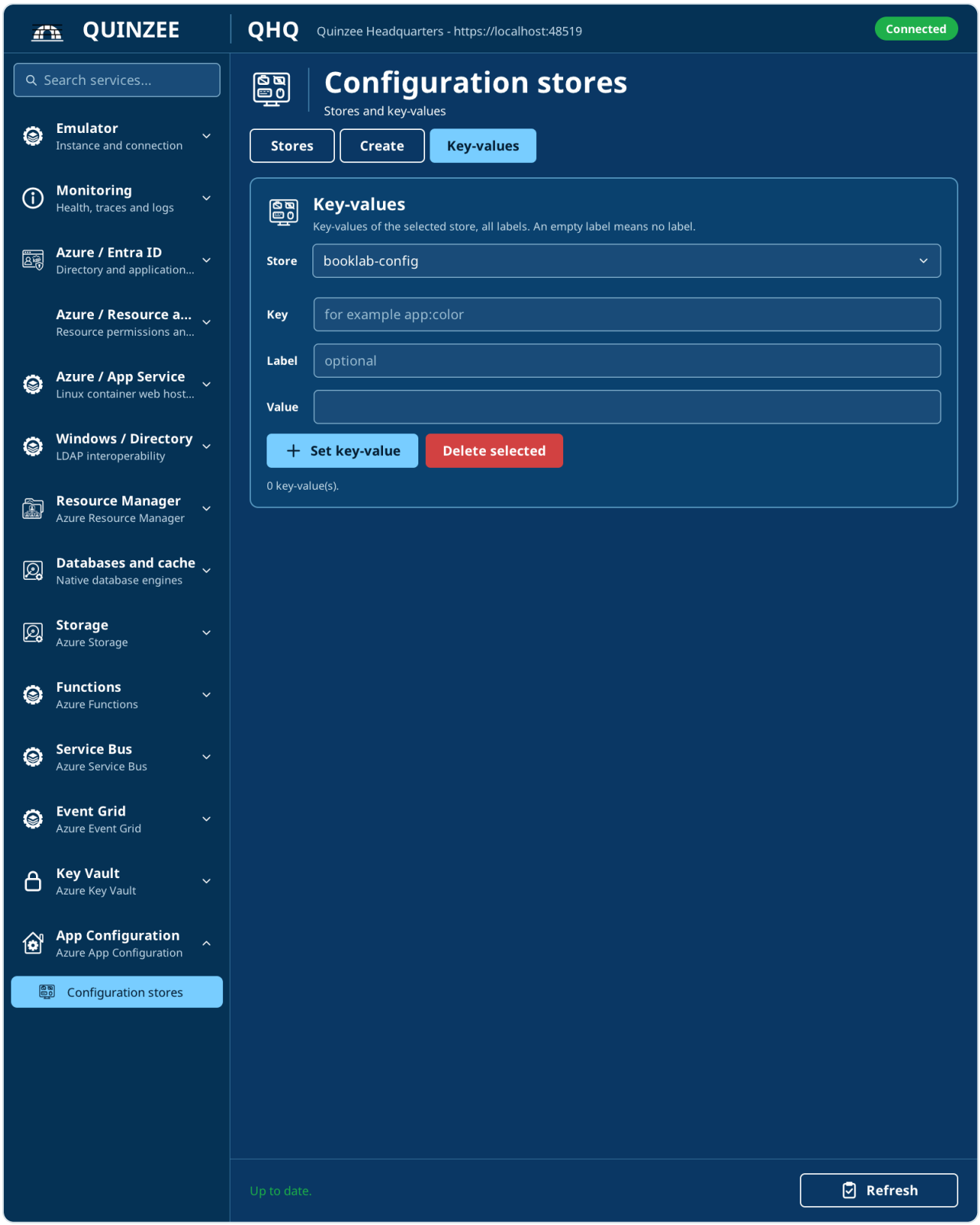
Stores



Create


QUINZEE
QHQ
Quinzee Headquarters - https://localhost:48519
Connected
Search services...
Emulator
Instance and connection
Monitoring
Health, traces and logs
Azure / Entra ID
Directory and application...
Azure / Resource a...
Resource permissions an...
Azure / App Service
Linux container web host...
Windows / Directory
LDAP interoperability
Resource Manager
Azure Resource Manager
Databases and cache
Native database engines
Storage
Azure Storage
Functions
Azure Functions
Service Bus
Azure Service Bus
Event Grid
Azure Event Grid
Key Vault
Azure Key Vault
App Configuration
Azure App Configuration
Configuration stores
Configuration stores
Stores and key-values
Stores
Create
Key-values
New configuration store
PUT through Resource Manager. The resource group must exist; store names are 5-50 letters, digits, '-' and '_'.
Subscription
Quinzee Development (19c3c36e)
Resource group
for example rg-sandbox
Name
for example appconfig-dev
Location
westeurope
SKU
standard
Create store
Up to date.
Refresh

Key-values


QUINZEE
QHQ
Quinzee Headquarters - https://localhost:48519
Connected
Search services...
Emulator
Instance and connection
Monitoring
Health, traces and logs
Azure / Entra ID
Directory and application...
Azure / Resource a...
Resource permissions an...
Azure / App Service
Linux container web host...
Windows / Directory
LDAP interoperability
Resource Manager
Azure Resource Manager
Databases and cache
Native database engines
Storage
Azure Storage
Functions
Azure Functions
Service Bus
Azure Service Bus
Event Grid
Azure Event Grid
Key Vault
Azure Key Vault
App Configuration
Azure App Configuration
Configuration stores
Configuration stores
Stores and key-values
Stores
Create
Key-values
Key-values
Key-values of the selected store, all labels. An empty label means no label.
Store
booklab-config
Key
for example app:color
Label
optional
Value
Set key-value
Delete selected
0 key-value(s).
Up to date.
Refresh

Refresh this atlas safely

The normal PDF build uses reviewed PNGs and starts no services. To recapture, run from the repository root after building the native binaries:

```
python3 tests/docs/service_examples.py --capture-dir build/admin-screenshots
```

This compiles the book templates, starts a temporary HTTPS cloud, validates them, deploys the in-process examples and captures QHQ through verified public APIs. It terminates its own cloud and removes temporary credentials/state. Engine and hosting templates are validated but their external workloads are not started by this capture. Review each image before copying the declared capture set into this book's `images` directory. Never point a publishing capture at a production or persistent private environment.

Related reading

- The Book of Quinzee: Manage the environment with QHQ
- Quinzee Internals: QHQ and libsimple
- Quinzee Internals: Inspect a disposable instance from Linux
- Scenario overview: Practising Azure Administration

PUBLICATION NOTES

About this edition

This book is generated from the Quinzee Markdown manuals. Screenshots show the real application and QHQ in an isolated local environment; names, identifiers and ports vary between runs.

The PDF includes DejaVu font subsets. Their copyright and permission notice follows.

Fonts are (c) Bitstream (see below). DejaVu changes are in public domain. Glyphs imported from Arev fonts are (c) Tavmjong Bah (see below)

Bitstream Vera Fonts Copyright ----------

Copyright (c) 2003 by Bitstream, Inc. All Rights Reserved. Bitstream Vera is a trademark of Bitstream, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of the fonts accompanying this license ("Fonts") and associated documentation files (the "Font Software"), to reproduce and distribute the Font Software, including without limitation the rights to use, copy, merge, publish, distribute, and/or sell copies of the Font Software, and to permit persons to whom the Font Software is furnished to do so, subject to the following conditions:

The above copyright and trademark notices and this permission notice shall be included in all copies of one or more of the Font Software typefaces.

The Font Software may be modified, altered, or added to, and in particular the designs of glyphs or characters in the Fonts may be modified and additional glyphs or characters may be added to the Fonts, only if the fonts are renamed to names not containing either the words "Bitstream" or the word "Vera".

This License becomes null and void to the extent applicable to Fonts or Font Software that has been modified and is distributed under the "Bitstream Vera" names.

The Font Software may be sold as part of a larger software package but no copy of one or more of the Font Software typefaces may be sold by itself.

THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL BITSTREAM OR THE GNOME FOUNDATION BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE.

Except as contained in this notice, the names of Gnome, the Gnome Foundation, and Bitstream Inc., shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Font Software without prior written authorization from the Gnome Foundation or Bitstream Inc., respectively. For further information, contact: fonts at gnome dot org.

Arev Fonts Copyright ----------

Copyright (c) 2006 by Tavmjong Bah. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of the fonts accompanying this license ("Fonts") and associated documentation files (the "Font Software"), to reproduce and distribute the modifications to the Bitstream Vera Font Software, including without limitation the rights to use, copy, merge, publish, distribute, and/or sell copies of the Font Software, and to permit persons to whom the Font Software is furnished to do so, subject to the following conditions:

The above copyright and trademark notices and this permission notice shall be included in all copies of one or more of the Font Software typefaces.

The Font Software may be modified, altered, or added to, and in particular the designs of glyphs or characters in the Fonts may be modified and additional glyphs or characters may be added to the Fonts, only if the fonts are renamed to names not containing either the words "Tavmjong Bah" or the word "Arev".

This License becomes null and void to the extent applicable to Fonts or Font Software that has been modified and is distributed under the "Tavmjong Bah Arev" names.

The Font Software may be sold as part of a larger software package but no copy of one or more of the Font Software typefaces may be sold by itself.

THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL TAVMJONG BAH BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE.

Except as contained in this notice, the name of Tavmjong Bah shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Font Software without prior written authorization from Tavmjong Bah. For further information, contact: tavmjong @ free . fr.

TeX Gyre DJV Math ---------- Fonts are (c) Bitstream (see below). DejaVu changes are in public domain.

Math extensions done by B. Jackowski, P. Strzelczyk and P. Pianowski (on behalf of TeX users groups) are in public domain.

Letters imported from Euler Fraktur from AMSfonts are (c) American Mathematical Society (see below). Bitstream Vera Fonts Copyright Copyright (c) 2003 by Bitstream, Inc. All Rights Reserved. Bitstream Vera is a trademark of Bitstream, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of the fonts accompanying this license (“Fonts”) and associated documentation files (the “Font Software”), to reproduce and distribute the Font Software, including without limitation the rights to use, copy, merge, publish, distribute, and/or sell copies of the Font Software, and to permit persons to whom the Font Software is furnished to do so, subject to the following conditions:

The above copyright and trademark notices and this permission notice shall be included in all copies of one or more of the Font Software typefaces.

The Font Software may be modified, altered, or added to, and in particular the designs of glyphs or characters in the Fonts may be modified and additional glyphs or characters may be added to the Fonts, only if the fonts are renamed to names not containing either the words “Bitstream” or the word “Vera”.

This License becomes null and void to the extent applicable to Fonts or Font Software that has been modified and is distributed under the “Bitstream Vera” names.

The Font Software may be sold as part of a larger software package but no copy of one or more of the Font Software typefaces may be sold by itself.

THE FONT SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL BITSTREAM OR THE GNOME FOUNDATION BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE. Except as contained in this notice, the names of GNOME, the GNOME Foundation, and Bitstream Inc., shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Font Software without prior written authorization from the GNOME Foundation or Bitstream Inc., respectively. For further information, contact: fonts at gnome dot org.

AMSFonts (v. 2.2) copyright

The PostScript Type 1 implementation of the AMSFonts produced by and previously distributed by Blue Sky Research and Y&Y, Inc. are now freely available for general use. This has been accomplished through the cooperation of a consortium of scientific publishers with Blue Sky Research and Y&Y. Members of this consortium include:

Elsevier Science IBM Corporation Society for Industrial and Applied Mathematics (SIAM) Springer-Verlag American Mathematical Society (AMS)

In order to assure the authenticity of these fonts, copyright will be held by the American Mathematical Society. This is not meant to restrict in any way the legitimate use of the fonts, such as (but not limited to) electronic distribution of documents containing these fonts, inclusion of these fonts into other public domain or commercial font collections or computer applications, use of the outline data to create derivative fonts and/or faces, etc. However, the AMS does require that the AMS copyright notice be removed from any derivative versions of the fonts which have been altered in any way. In addition, to ensure the fidelity of TeX documents using Computer Modern fonts, Professor Donald Knuth, creator of the Computer Modern faces, has requested that any alterations which yield different font metrics be given a different name.

Id