AUTHCHA LEARN

Foundations

What Authcha is, how a sign-in is decided, and how to get from a fresh install to a working two-factor login.

6 modules · about 145 minutes · Administrators new to Authcha

1. What Authcha is
2. How a device login is decided
3. Install and first sign-in
4. Console roles and securing the console
5. Your first two-factor device login
6. Login flows, settings and lockouts

Quizzes, the final assessment and live links are online at learn.authcha.cloud. Labs use the live demo at demo.authcha.cloud.

Module 1 of 6 · 20 minutes

What Authcha is

By the end you can

- Say in two sentences what Authcha does and who it is for
- Name the main services in an installation and what each one is for
- Tell the three kinds of sign-in Authcha handles apart

Why it exists

A network has two kinds of people to let in. **Engineers** sign in to routers, switches and firewalls to run the network. Everyone else, and everything else, **connects** to it: laptops, phones, printers, cameras, visitors.

Most networks buy a separate product for each job: a RADIUS server, a network access control (NAC) product, a guest portal, a tool for two-factor codes. Each has its own user list and its own renewal. Authcha puts them on **one server, with one user directory and one web console**. It runs entirely inside your network and needs no cloud service, so it works on a network with no internet access.

Words used here

RADIUS is the standard protocol a switch or access point uses to ask a central server "may this user in?". The switch or access point is called the **NAS** (network access server). Authcha is the server that answers.

The three kinds of sign-in

Every RADIUS request that reaches Authcha follows one of three paths, decided by what the request contains. All three run side by side.

What arrives	The path	Typical senders
A request carrying an EAP message	**802.1X**	Laptops and phones that can sign in with a username and password or a certificate
A user name that is a MAC address	**MAC bypass (MAB)**	IP phones, printers, cameras, ONTs and other devices that cannot do 802.1X
Anything else	**Password plus one-time code**	Engineers signing in to routers, switches and firewalls

You will meet each path in its own module. For now, remember that one server answers all of them.

What is inside

An installation is a handful of containers on one Linux host.

Service	What it does
FreeRADIUS	Answers RADIUS requests on UDP 1812 and 1813 and applies Authcha's policy
PostgreSQL	Stores users, devices, policies, accounting records and the audit trail
Redis	Holds one-time codes, retry counters and short-lived decisions
OTP delivery service	Sends one-time codes through your messaging provider
Admin API and console	The web console and the API. Also runs the background jobs: backups, quota checks, audit retention
Ops sidecar	The only part with access to Docker, so the console itself needs no special rights
Caddy	Serves the console over HTTPS on port 443

Only RADIUS (UDP 1812 and 1813, and TCP 2083 if you use RadSec) and HTTPS (TCP 443) must be reachable from your network.

What else is included

Beyond the three sign-in paths, the same installation can also do these. Each has its own track or module in this training.

- **Guest access:** vouchers and a captive portal for visitors.
- **Certificates:** a built-in certificate authority for EAP-TLS and RadSec.
- **Posture:** an optional agent that checks a device before it is let in.
- **Quotas and disconnects:** data limits and RFC 5176 disconnects.
- **DHCP:** subnets and reservations for a Kea DHCP server.
- **Identity:** Authcha can be the OpenID Connect provider your own applications sign in with.
- **Operations:** backups, a health view, an audit trail and event streaming to your SIEM.

Which optional modules you can configure depends on your license edition. The Licensing guide (https://demo.authcha.cloud/docs/licensing) explains editions.

> **Try it on the demo**
>
> Open the live demo (https://demo.authcha.cloud/) and sign in as `demo-viewer` with password `Authcha-View-2026`. The demo runs the real product with sample users, devices and traffic. This account is read-only, so you cannot break anything.

Lab HANDS-ON

1	Open the	live demo (https://demo.authcha.cloud/)	and sign in as	`demo-viewer`	(password	`Authcha-View-2026`	).	
2	Look at the left menu. Find	**Live Auth**	,	**Users**	,	**Devices**	and	**Network access**. These are the areas you will use most.
3	Open	Live Auth (https://demo.authcha.cloud/activity/live)	. Watch a few lines arrive. Each line is one RADIUS decision.					

4 Open Devices (https://demo.authcha.cloud/devices). Each row is a network device (a NAS) that is allowed to send requests.

5 Open System (https://demo.authcha.cloud/system) and find the list of services. Match each one to the table above.

6 Open the Overview guide (https://demo.authcha.cloud/docs/overview) and find the architecture diagram. Trace one request from a switch to the database and back.

Cheat sheet KEEP THIS

- One server answers three paths: **802.1X**, **MAC bypass**, **password plus code**.
- Core services: FreeRADIUS, PostgreSQL, Redis, OTP delivery, console, ops sidecar, Caddy.
- Reachable from your network: **UDP 1812/1813**, **TCP 443** (and TCP 2083 for RadSec).
- Console accounts (people who run Authcha) are separate from RADIUS users (people and devices on the network).
- Demo: `demo.authcha.cloud`, viewer login on the demo page (https://authcha.cloud/demo.html).

Module 2 of 6 · 25 minutes

How a device login is decided

By the end you can

- List the five things that decide a device login and say what each one does
- Explain how the same user gets different rights on different vendors
- Describe what a jump server rule can and cannot do

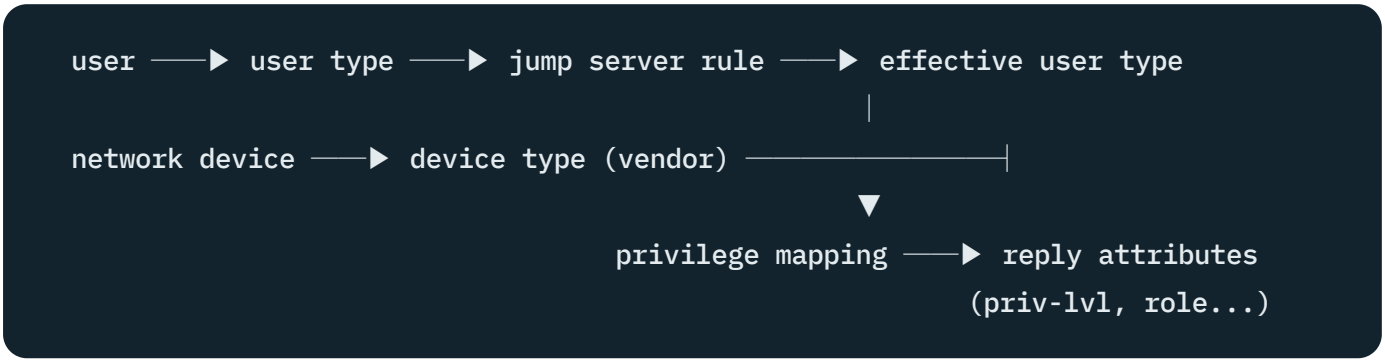
Start with the problem

An engineer types `ssh router1`, a username and a password. Two things must now be decided: **may this person in**, and **what may they do once inside**? A senior engineer should get full rights. A junior engineer should only be able to look. A script should get something in between.

Every router vendor writes this down differently. Cisco wants `shell:priv-lvl=15`. Juniper wants the name of a local template user. Authcha lets you describe the rule **once**, by kind of person, and translates it for each vendor.

The five pieces

Every device login is decided by these five pieces of configuration, in this order.



1. **User.** A person or an automation account, with a password, a mobile number and one **user type**. The seeded types are `human-write`, `human-read`, `automation` and `lookingglass`. You can add your own, such as `noc-l1`.
2. **Jump server rule (optional).** Which source addresses a user type may log in from. A rule can also **downgrade** a user: a `human-write` user connecting from a read-only bastion is treated as `human-read`. Rules never upgrade anyone.

3. **Network device.** Each router, switch or firewall is registered with its IP address and a **device type**, which is its vendor profile.
4. **RADIUS client.** The shared secret the device uses to talk to Authcha.
5. **Privilege mapping.** For each vendor, which privilege group each effective user type receives. The group holds the vendor's reply attributes.

Because the mapping is per vendor, one user account works across a mixed fleet. You do not create a separate account for every vendor.

> **Jump servers are optional**
>
> On a new installation, jump servers are set to **Optional**: a login from an address with no rule signs in with the user's own type. Set **Jump Servers → Are jump servers needed?** to **Required** and a source with no rule is refused. An installation that already had rules keeps **Required** until you change it. If your network has no bastion hosts, you can skip this piece.

A worked example

Take a user `asha` of type `human-write`, logging in to a Cisco switch from the read-only bastion `192.0.2.20`.

Step	What happens
Jump server rule	The rule for `192.0.2.20` says `human-write` becomes `human-read`
Effective user type	`human-read`
Device type	The switch is registered as `ios`
Privilege mapping	`human-read` on `ios` maps to `CISCO-OPER` (read-only)
Reply	The switch receives the read-only privilege for this login

If `asha` connects from the write bastion `192.0.2.30` instead, the rule keeps `human-write`, the mapping gives `CISCO-ADMIN`, and the switch receives `shell:priv-lvl=15`. Same person, same password, different rights, decided by where the login came from.

What Authcha ships with

Authcha ships privilege presets for 35 vendors, and mappings for the four seeded user types on the common ones. For example, `human-write` on Cisco IOS maps to `CISCO-ADMIN`, on Juniper to a local template user `raduser5`, and on Nokia SR OS to `Nokia-Admin`. Review these before go-live and adjust them to your own privilege model. Changes take effect on the next login with no restart.

> **Rules never upgrade**
>
> A jump server rule can lower a user's rights but never raise them. If a login from an unknown source is refused when jump servers are Required, that is the rule doing its job.

Lab HANDS-ON

1. Open the live demo (https://demo.authcha.cloud/) as `demo-viewer` (password `Authcha-View-2026`).
2. Open Users (https://demo.authcha.cloud/users). Pick any user and note its **user type**.
3. Open User Types (https://demo.authcha.cloud/user-types) and find the four seeded types.
4. Open Jump Servers (https://demo.authcha.cloud/jump-servers). Find the setting that says whether jump servers are needed, and note whether it is Required or Optional.
5. Open Devices (https://demo.authcha.cloud/devices). Pick a device and note its **device type**.
6. Open Auth Simulator (https://demo.authcha.cloud/activity/simulator). Enter a username, a device IP and a source IP from the lists you just viewed. Read the four steps it shows and the reply attributes at the end. The simulator never sends a code and never changes any counter.

7 Change only the source IP and run it again. Did the effective user type change?

Cheat sheet KEEP THIS

- The five pieces: **user**, **jump server rule**, **device**, **RADIUS client**, **privilege mapping**.
- Seeded user types: `human-write`, `human-read`, `automation`, `lookingglass`.
- Rules **lower** rights, never raise them.
- Jump servers: **Optional** on new installs, **Required** if rules already existed.
- Test without side effects: **Try It** → **Auth Simulator**.
- Read more: Getting started (https://demo.authcha.cloud/docs/getting-started).

Module 3 of 6 · 25 minutes

Install and first sign-in

By the end you can

- Check that a host and network meet the requirements before installing
- Install with the one-line installer, or with a bundle on a server with no internet
- Sign in for the first time and secure the admin account
- Confirm the stack is healthy

What you need first

The host. A Linux x86_64 server: 2 vCPU, 4 GB of memory and 20 GB of disk is the minimum; 4 vCPU, 8 GB and a 50 GB SSD is recommended. Ubuntu 22.04 or 24.04, Debian 12 or 13, and RHEL, Rocky Linux or AlmaLinux 9 are supported. You need Docker Engine 24 or later with the Compose v2 plugin. The installer installs Docker for you if it is missing.

The network. Give the host a static IP address or a stable DNS name, because your network devices will point at it. These ports must be free: 443/tcp, 80/tcp, 1812 and 1813/udp, 2083/tcp and 8444/tcp. A few loopback ports must also be free: 5432, 6380, 3000, 8443 and 50051. The installer checks all of them before it changes anything.

Check without installing

Run the installer with `--check` and it runs the preflight checks and stops without changing anything.

Install

On a fresh server, one command does everything: it checks the server, installs Docker if needed, downloads the release, verifies its signature, installs it and activates the license.

```bash
curl -fsSL https://get.authcha.cloud | sudo bash -s -- --license AUTHCHA-… --hostnam
```

To evaluate first, start a 30-day trial. A code is emailed to you.

```bash
curl -fsSL https://get.authcha.cloud | sudo bash -s -- --trial you@company.com --hos
```

Always pass `--hostname`. Passkeys, links in emails and the posture agent all use it.

Check the installer before you run it

The installer is a readable shell script, and it is signed. You can verify the signature before running it.

```bash
curl -fsSLO https://get.authcha.cloud/install.sh
curl -fsSLO https://get.authcha.cloud/install.sh.sig
curl -fsSLO https://get.authcha.cloud/release-public.pem
openssl pkeyutl -verify -pubin -inkey release-public.pem -rawin -in install.sh -sigf
```

The release it downloads is checked the same way, by SHA-256 and by an Ed25519 signature. Nothing is installed if either check fails.

A server with no internet

Download three files from your account team, or from the license server: `authcha-X.Y.Z.tar.gz`, `authcha-X.Y.Z.tar.gz.sig` and `install.sh`. Copy them to the server, then install from the bundle. Docker Engine 24 or later must already be installed.

```bash
sudo bash install.sh --bundle authcha-X.Y.Z.tar.gz --license-file license.json --hos
```

The license is then activated offline. The Licensing guide (https://demo.authcha.cloud/docs/licensing) shows how.

What the installer creates

It verifies every file in the bundle, loads the container images, creates `/opt/authcha`, writes `.env` and `otp.env` with a fresh random value for every secret, and starts the stack. `.env` and `otp.env` contain every bootstrap secret for your installation and are readable by root only. **Keep a copy somewhere safe.** The installer never overwrites them.

```
/opt/authcha/
  .env, otp.env       settings and secrets (mode 600)
  docker-compose.yml
  config/             FreeRADIUS policy, PostgreSQL, TLS proxy
  data/postgres       the database
  backups/            scheduled backups and audit archives
  scripts/            maintenance jobs run from /etc/cron.d/authcha
  integrations/       pfSense agent and helpers
```

First sign-in

The installer ends by printing the console address and the `admin` password. The password is shown once on the first install, and also saved in `/opt/authcha/run/initial-admin-password`. Every install generates a random password; none starts with a default one.

1. Open `https://<your-host>/`. Accept the browser's certificate warning, because the console starts with a self-signed certificate.
2. Choose **Use a password instead** and sign in as `admin`.
3. Open **Profile**, then **Change password** and set a strong one.
4. Under **Two-factor authentication**, choose **Begin enrollment**, scan the QR code with an authenticator app and confirm with a six-digit code.
5. Under **Profile** → **Passkeys**, add a passkey, then sign in with it from now on.
6. Open **Settings** → **License** and check the edition and expiry.

If you lose the password, run `sudo authcha reset-admin-password`.

> **Keep admin as your break-glass account**
>
> `admin` is a local account. It keeps working even after you connect single sign-on, so store its credentials safely.

Check that it is healthy

```bash
cd /opt/authcha
docker compose ps
curl -sk https://127.0.0.1/api/health
```

Every service should be `running`, and `db`, `redis` and `otp_service` should be `healthy`. The health call should answer `{"status":"ok","db_ok":true}`. In the console, **System** shows the same.

Open the right ports

Restrict UDP 1812 and 1813 to the subnets your network devices send RADIUS from, and TCP 443 to your management network. The shared secrets are the only protection on the RADIUS ports.

> **Codes need a provider**
>
> Until you set up a delivery provider, logins that need a one-time code are refused with "Service temporarily unavailable". 802.1X, MAC bypass and console sign-ins are not affected. You will set this up in the next modules.

Lab HANDS-ON

This lab needs your own server or a test VM. On the demo you can still do steps 6 and 7.

1. Check the host has Docker 24 or later (`docker --version`) and that nothing is listening on 443, 80 or 1812: `ss -lntu | grep -E ':(443|80|1812) '`.
2. Run the installer with `--check` first: download `install.sh` as shown above and run `sudo bash install.sh --check`.
3. Install with a trial, or with your license, using `--hostname`.
4. Sign in as `admin`, change the password and turn on two-factor.
5. Run `docker compose ps` and the health call above. Confirm everything is healthy.

6 Open Settings (https://demo.authcha.cloud/settings) on the demo and find **License**. Note the edition and expiry. (On the demo, sign in as `demo-viewer`, password `Authcha-View-2026`.)

7 Open System (https://demo.authcha.cloud/system) and match each running service to the table in the first module.

Cheat sheet KEEP THIS

- One command: `curl -fsSL https://get.authcha.cloud | sudo bash -s -- --license AUTHCHA-… --hostname FQDN`
- Trial: `--trial you@company.com`. Preflight only: `--check`. Offline: `--bundle`.
- Secrets live in `/opt/authcha/.env` and `otp.env` (mode 600). Back them up.
- First sign-in: `admin`, then change password, turn on two-factor, add a passkey.
- Lost password: `sudo authcha reset-admin-password`.
- Health: `docker compose ps` and `curl -sk https://127.0.0.1/api/health`.
- Full detail: Installation (https://demo.authcha.cloud/docs/installation).

Module 4 of 6 · 20 minutes

Console roles and securing the console

By the end you can

- Tell console accounts apart from RADIUS users
- Pick the right role for an administrator, a read-only reviewer and a front-desk sponsor
- Turn on two-factor, passkeys and single sign-on for administrators
- Limit an administrator to part of the estate

Two kinds of account

This is the first thing to get straight. **RADIUS users** are the people and devices that sign in to your *network*. **Console accounts** are the people who *administer Authcha*. They are separate lists. A console account does not give network access, and a RADIUS user cannot sign in to the console.

The three roles

Manage console accounts under **Administrators** → **Accounts**.

Role	What it can do
admin	Everything: administrators, API tokens, integrations, settings, license and backups
viewer	Read-only access to configuration, monitoring and audit. No access to voucher codes, credentials or other secrets
sponsor	Guest vouchers only: can issue vouchers and see their own

Each console account counts against the administrator cap in your license, if your license has one.

Limit an administrator to part of the estate

Administrators → **Admin Scoping** limits an administrator to particular **device classes** or **user types**. A facilities team might manage only `iot` endpoints, and a

NOC lead only `noc-l1` users. An administrator with no scoping grants is unrestricted. Only admins can change scoping.

Protect the sign-in

Layer these. Each one makes a stolen password less useful.

Two-factor (TOTP). Under **Profile** → **Two-factor authentication** → **Begin enrollment**, scan the QR code and confirm with a six-digit code. After that every sign-in asks for a code. To make it mandatory for everyone, use **Settings** → **Admin Security**. Accounts without it can still sign in but are held on their profile page until they enroll, so nobody is locked out when you switch the policy on. This is recommended for every production deployment.

Passkeys. A passkey lets an administrator sign in with a fingerprint, face or device PIN. It is tied to your console's address, so it cannot be phished, and it always needs the person to unlock it. A passkey sign-in is stronger than a password plus a code, so Authcha does not also ask for TOTP. It needs the console opened at an `https://` address with a DNS name, because browsers refuse passkeys for IP addresses.

Single sign-on (OIDC). Administrators can sign in through your identity provider (Okta, Microsoft Entra ID, Google, Keycloak). Access is granted only through group membership: a user in an admin group becomes an admin, in a viewer group a viewer, and in neither is refused. Every SSO sign-in is recorded in the audit trail. The local accounts, including `admin`, keep working alongside it.

Sessions and password resets

Administrators → **Console Sessions** lists everyone who is signed in, with their address and browser. End a session to make that person sign in again. Sessions last 8 hours by default. If email (SMTP) is configured, the sign-in page offers a password-reset link; without it, another admin can reset the password.

> **Least privilege**
>
> Give people the smallest role that does their job. A reviewer or auditor needs **viewer**. Front-desk staff who only issue guest vouchers need **sponsor**. Keep **admin** for the few people who change settings.

Lab HANDS-ON

Steps 1 to 3 work on the demo with the viewer login (`demo-viewer` , password `Authcha-View-2026`). Steps 4 to 6 need your own install.

1. Open Administrators (https://demo.authcha.cloud/admins) on the demo. See the roles listed.
2. Open Profile (https://demo.authcha.cloud/profile) and look at the **Two-factor authentication** and **Passkeys** sections.
3. Open Settings (https://demo.authcha.cloud/settings) and find **Admin Security** and **Single Sign-On**.
4. On your install, create an account with the **viewer** role and sign in with it. Confirm you can see the configuration but not secrets.
5. Turn on TOTP for your own account.
6. Under **Settings → Admin Security**, make TOTP mandatory, then check that an account without it is sent to its profile page.

Cheat sheet KEEP THIS

- **admin**, **viewer**, **sponsor**. Console accounts are not RADIUS users.
- Protect sign-in with **TOTP** (make it mandatory in Settings → Admin Security), **passkeys** and **SSO**.
- Passkeys need an `https://` DNS name, not an IP address.
- Scope an administrator under **Administrators → Admin Scoping**.
- End a session under **Administrators → Console Sessions**.
- Read more: Administration (https://demo.authcha.cloud/docs/administration), Passwordless sign-in (https://demo.authcha.cloud/docs/passwordless).

Module 5 of 6 · 35 minutes

Your first two-factor device login

By the end you can

- Register a network device and its RADIUS client with the right secret
- Create a user and map their type to privileges on that vendor
- Prove the setup with the Auth Simulator before touching the device
- Configure a Cisco, Juniper, Arista, MikroTik or FortiGate device and sign in with a code

The plan

You will go from an empty console to a working two-factor login on one real device, in eight steps. Follow them in order. The same checklist is in the console as **Guided Setup**, which ticks off each step for you.

You need: an installed Authcha, a delivery provider for codes (WhatsApp or SMS, under **Integrations → OTP Delivery**; the delivery channels guide (https://demo.authcha.cloud/docs/delivery-channels) shows how to send codes by other routes), and one network device you can configure.

Step 1. Register the device

Go to **Devices → Add device**.

Field	What to enter
IP address	The address the device sends RADIUS from. It must match the NAS-IP-Address in its requests. Use the RADIUS source interface you configure on the device; a loopback is best.
Device type	The vendor profile, such as `ios`, `iosxe`, `junos`, `arista`, `mikrotik` or `fortinet`
Hostname	Optional. Shown in logs and reports
Auth flow	Leave on **Inherit** for now. The next module explains the choices

You can import many devices at once from a CSV on the Devices page. Each registered device counts against the device cap in your license, if there is one.

Step 2. Add its RADIUS client

Go to **Authorization Policies** → **RADIUS Clients** → **Add client**. Enter the device's IP address (or a CIDR range that covers several devices with the same secret), a short name, and a **long random shared secret**. Use a different secret per device or per site.

FreeRADIUS reads its clients only at start, so the console applies your change by restarting the RADIUS service a few seconds after the last edit. The clients page shows **Applying…**, then **Applied**. The restart is brief and devices simply retry. Deleted clients are no longer loaded.

> **Always add devices in the console**
>
> The file `/opt/authcha/config/freeradius/clients.conf` admits only the Authcha host itself, for local diagnostics. Add real devices in the console.

Step 3 (optional). Define jump servers

Only if engineers connect through bastion hosts, or you want to limit where logins may come from. For each one, enter its IP address, the base user type it accepts and the effective user type users get from it. A login from a source with no matching rule is refused only when jump servers are set to **Required**. If you have no bastion hosts, skip this step.

Step 4. Create a user

Go to **Users** → **Add user**. The password is generated and shown once; pass it on securely. Enter the **mobile number** that will receive codes, and the **user type**, such as `human-write`. A user with no mobile number cannot complete a code login, unless the device is set to password-only.

Step 5. Check the privilege mapping

Go to **Authorization Policies** → **Privilege Mapping**. Authcha ships presets for 35 vendors and mappings for the four seeded user types on the common ones. Look at the row for your device type and your user type. Change the group or edit its reply attributes if your privilege model differs. Changes apply to the next login with no restart.

To add a vendor with no preset, register the device with a new device type name, create privilege groups with that vendor's attributes, and map each user type to a group.

Step 6. Test with Try It

Before touching the real device, open **Try It → Auth Simulator** (also at Auth Simulator (https://demo.authcha.cloud/activity/simulator) on the demo). Enter the username, the device IP and the source IP. It runs the same four steps as a real login (user and device, jump server rule, effective user type, privilege mapping) and shows the verdict and the exact reply attributes the device would receive. It never sends a code.

Then open **Try It → OTP Delivery** and send a real test code to a mobile number. This proves your provider works end to end.

If the simulator accepts the login and the code arrives, you are ready.

Step 7. Configure the device

Point it at Authcha's address, with the shared secret from step 2, sending RADIUS from the interface whose address you registered in step 1.

> **Set the timeout to at least 10 seconds**
>
> The first request of a login waits for the provider to accept the code. A short device timeout makes it retry or fail over too early.

Cisco IOS and IOS-XE:

```
aaa new-model
radius server AUTHCHA
 address ipv4 192.0.2.10 auth-port 1812 acct-port 1813
 timeout 10
 retransmit 1
 key <shared-secret>
aaa group server radius AUTHCHA-GRP
 server name AUTHCHA
 ip radius source-interface Loopback0
aaa authentication login default group AUTHCHA-GRP local
aaa authorization exec default group AUTHCHA-GRP local if-authenticated
aaa accounting exec default start-stop group AUTHCHA-GRP
line vty 0 15
 login authentication default
 authorization exec default
```

Keep `local` in the list so you can still reach the device if Authcha is unreachable. Junos, Arista EOS, MikroTik RouterOS and FortiGate have their own short recipes in the Getting started guide (https://demo.authcha.cloud/docs/getting-started). Every vendor needs the same three things: the server address, the shared secret and a source address that matches the device you registered.

> **Keep a way back in**
>
> Before you change authentication on a device, keep a local fallback account and an open session. Test in a second session before you close the first.

Step 8. Log in for real

1. SSH to the device (from a jump server if you use them).
2. Enter the username and password.
3. Enter the code that arrives by WhatsApp or SMS.

Watch it happen in **Live Auth → Logs**. Afterwards, find it in **Audit Trail → Auth History**, and the code delivery in **Audit Trail → OTP Funnel**. If the login is refused, the Troubleshooting guide (https://demo.authcha.cloud/docs/troubleshooting) walks through the device-login causes.

Lab HANDS-ON

Steps 1 to 4 work on the demo as `demo-viewer` (password `Authcha-View-2026`). Steps 5 to 9 need your own install and a device or a lab VM.

1. Open Devices (https://demo.authcha.cloud/devices) on the demo and find a Cisco device. Note its device type and IP address.
2. Open Auth Simulator (https://demo.authcha.cloud/activity/simulator) and enter a demo username, that device's IP and a source IP. Read the verdict and the reply attributes.
3. Open Live Auth (https://demo.authcha.cloud/activity/live) and find a recent accepted login. Open its detail.
4. Open Users (https://demo.authcha.cloud/users) and find the user from the log entry. What is their user type?
5. On your install, register a device and its RADIUS client with a new long secret. Wait for the clients page to show **Applied**.
6. Create a user with a mobile number and the `human-write` type.
7. Run the simulator for that user and device. You should see an accept and the privilege attributes.
8. Send a test code from **Try It → OTP Delivery**.
9. Configure the device and sign in. Find your login in **Live Auth** and in **Auth History**.

Cheat sheet KEEP THIS

1. **Devices → Add device** (IP, device type)
2. **Authorization Policies → RADIUS Clients** (address, long secret), wait for **Applied**

3. Jump servers (optional)
4. **Users** → **Add user** (mobile number, user type)
5. **Privilege Mapping** (review per vendor)
6. **Try It** → **Auth Simulator**, then **OTP Delivery** test
7. Configure the device: server, secret, source interface, timeout of 10 s or more, local fallback
8. Sign in, then check **Live Auth**, **Auth History**, **OTP Funnel**
9. Read more: Getting started (https://demo.authcha.cloud/docs/getting-started).

Module 6 of 6 · 20 minutes

Login flows, settings and lockouts

By the end you can

- Choose the right login flow for a device
- Predict which setting wins when it is set at several levels
- Explain the lockout rule and clear a lockout
- Name the risk of password-only and where its use is recorded

Why there is more than one flow

Not every device can show a second prompt. Authcha supports three ways to log in, and you choose per device.

Flow	How the user logs in	Use it for
Challenge (default)	Password, then a second prompt for the code	Devices that support the RADIUS Access-Challenge, which is most modern routers, switches and firewalls over SSH
Trigger phrase	Type the trigger phrase (default `givemeotp`) as the password. This sends a code and that login fails once. Then log in again with the code as the password	Devices or clients that cannot show a second prompt
Password only	Password alone. No code is sent	Only the rare device that supports neither of the above

Password-only removes the second factor

You can set it only on an individual device, never on a group, a vendor default or the global default. Every login it allows is recorded as `bypassed` in the OTP Funnel, so you can audit its use.

Which setting wins

The auth flow, the code lifetime and the trigger phrase are each resolved **independently**. Authcha uses the most specific value that is set, in this order:

1. **The device**, on the device dialog
2. **The device group**, under **Device Groups**
3. **The vendor default**, under **Settings** → **Authentication Policy**, per device type
4. **The global default**, under **Settings** → **Authentication Policy**

The device dialog shows the effective value for each setting and which level it came from. Changes apply to the next login with no restart. This is the same idea as CSS: the more specific rule beats the general one.

Code lifetime and lockouts

A code stays valid for the code lifetime, 1 hour by default. While it is valid, repeated logins **reuse the same code** instead of sending a new one, so a user who retries does not get a flood of messages.

After **three wrong passwords or three wrong codes**, the user is locked out for the rest of the code lifetime. To clear a lockout sooner, run this on the Authcha host:

```bash
cd /opt/authcha
docker compose exec redis redis-cli DEL retry:<username> otp_retry:<username>
```

Captive portals use a simpler path

Visitors and staff signing in to a captive portal (pfSense and similar) use a different, simpler path. Add the gateway as a device of type **Captive portal / hotspot (PAP)** and create users of type `network-user`. No code, no mobile number, no jump server and no privilege mapping are involved. The Guest and portals track covers this.

Lab HANDS-ON

1. Open Devices (https://demo.authcha.cloud/devices) on the demo (viewer login `demo-viewer`, password `Authcha-View-2026`). Open a device and look at the **Auth flow** field and the effective values it shows.
2. Open Settings (https://demo.authcha.cloud/settings) and find **Authentication Policy**. Note the global default and any vendor defaults.
3. Open Device Groups (https://demo.authcha.cloud/device-groups) and see whether a group overrides anything.
4. On your own install, pick a test device. Set its auth flow to **Trigger phrase** and its trigger phrase to something else. Confirm that the device dialog now shows the value as coming from the device.
5. Clear the device's value and confirm it falls back to the next level up.
6. Deliberately type a wrong password three times for a test user. Confirm the lockout, then clear it with the command above.

Cheat sheet KEEP THIS

- Flows: **Challenge** (default), **Trigger phrase** (`givemeotp`), **Password only** (device-level only).
- Resolution order: **device, group, vendor default, global default**. Each setting is resolved separately.
- A code is valid 1 hour by default and reused while valid.
- Lockout: **3** wrong passwords or codes. Clear with `redis-cli DEL retry:<user> otp_retry:<user>` .
- Captive portals: device type **Captive portal / hotspot (PAP)**, users of type `network-user` .

- Read more: Getting started (https://demo.authcha.cloud/docs/getting-started).