ngrok Public tunnels

START HERE

`ngrok config add-authtoken <token>`
Save your token

`ngrok tls <port> --crt <f> --key <f>`
Terminate TLS yourself

`ngrok http <port>`
Expose a local web app

`ngrok start --all`
Run all configured endpoints

`ngrok http <port> --url <domain>`
Use a fixed domain

`ngrok config check`
Validate the config file

`ngrok tcp <port>`
Expose a TCP service

`ngrok diagnose`
Test connectivity

Internet client public URL → HTTPS → **ngrok cloud** traffic policy → tunnel → **ngrok agent** your machine → forward → **Local service** localhost:port

CONCEPTS

Free accounts get one random public URL per session; reserve a domain in the dashboard to keep it stable.

Anyone with the URL can reach your service: add a traffic policy or auth before sharing it.

The agent logs requests in the inspector at http://127.0.0.1:4040, where you can replay them.

SETUP

`ngrok config add-authtoken <token>`
Store your authtoken
Written to the config file; NGROK_AUTHTOKEN overrides it.

`ngrok version`
Print the agent version

`ngrok update`
Update the agent
--channel picks stable, beta or unstable.

`ngrok config check`
Check the config file

`ngrok config edit`
Open the config in an editor

`ngrok config upgrade`
Migrate config to the newest format
--dry-run previews the change.

`ngrok completion`
Print shell completion code

TUNNELS

`ngrok http <port>`
Expose a local HTTP port

`ngrok http <host:port>`
Forward to another host

`ngrok http <port> --url <domain>`
Serve on a specific domain

`ngrok tcp <port>`
Expose a raw TCP port
Use for SSH, databases and other non-HTTP services.

`ngrok tls <port> --crt <file> --key <file>`
TLS endpoint with your cert

`ngrok tcp <port> --cidr-allow <cidr>`
Only allow an IP range

`ngrok tcp <port> --cidr-deny <cidr>`
Block an IP range

`ngrok http <port> --inspect=false`
Disable the traffic inspector

`ngrok http <port> --pooling-enabled`
Load-balance across agents

`ngrok http <port> --upstream-tls-verify=false`
Skip upstream cert check

`ngrok tls <port> --mutual-tls-cas <file>`
Require client certificates

TRAFFIC POLICY

`ngrok http <port> --traffic-policy-file <file>`
Apply a policy file

TRAFFIC POLICY (cont.)

`ngrok http <port> --traffic-policy-url <url>`
Fetch the policy from a URL

`ngrok http <port> --description <text>`
Label the endpoint

`ngrok http <port> --metadata <text>`
Attach custom metadata

Policy phases

Phase	When it runs
`on_tcp_connect`	a TCP connection opens
`on_http_request`	a request arrives
`on_http_response`	a response returns

CONFIG & SERVICES

`ngrok start <name>`
Start one configured endpoint

`ngrok start --all`
Start every configured endpoint

`ngrok start --none`
Connect without endpoints

`ngrok http <port> --config <file>`
Use an extra config file
Several --config files are merged.

`ngrok service install --config <file>`
Run the agent as a service

`ngrok service start`
Start the installed service

`ngrok service stop`
Stop the service

CONFIG & SERVICES (cont.)

`ngrok service restart`
Restart the service

`ngrok service uninstall`
Remove the service

`ngrok diagnose`
Test connectivity to ngrok
Exits 0 when OK, 1 otherwise.

`ngrok diagnose --write-report <file>`
Save a diagnostic report

RECIPES

Share a dev server

```
ngrok config add-authtoken
<token>
ngrok http 3000
# inspector:
http://127.0.0.1:4040
```

Basic auth via policy

```
# policy.yml
on_http_request:
  - actions:
      - type: basic-auth
        config:
          credentials:
            - user:<password>

ngrok http 3000 --traffic-
policy-file policy.yml
```

Config file with endpoints

```
# ~/.config/ngrok/ngrok.yml
version: 3
endpoints:
  - name: web
    upstream:
      url: 3000

ngrok start web
```

Expose SSH

```
ngrok tcp 22 --cidr-allow <cidr>
```