# The ngrok cheatbook



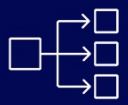
Installation 1

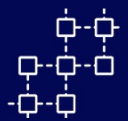

Exposing different kinds of servers 3


Troubleshooting 3


Add Authentication w/ Traffic Policy 4


Verify your webhooks 5


Do even more w/ internal endpoints 6


Manage traffic in other ways 7


CLI Flags 11

# Installation

macOS

```
# Install via Homebrew
brew install ngrok

# Add your authtoken
ngrok config add-authtoken <token>

# Start an endpoint
ngrok http 80
```

Linux

```
# Install via Apt
curl -sSL https://ngrok-agent.s3.amazonaws.com/
ngrok.asc \
  | sudo tee /etc/apt/trusted.gpg.d/ngrok.asc >/
dev/null \
  && echo "deb https://ngrok-
agent.s3.amazonaws.com bookworm main" \
  | sudo tee /etc/apt/sources.list.d/ngrok.list \
  && sudo apt update \
  && sudo apt install ngrok
# OR
# Install via Snap
snap install ngrok

# Add your authtoken
ngrok config add-authtoken <token>

# Start an endpoint
ngrok http 80
```

Windows

```
# Install via WinGet
winget install ngrok -s msstore
# OR
# Install via Scoop
scoop install ngrok

# Add your authtoken
ngrok config add-authtoken <token>

# Start an endpoint
ngrok http 80
```

# Installation

## Kubernetes

```
# Add ngrok Kubernetes Operator to Helm
helm repo add ngrok https://charts.ngrok.com

# Add ngrok API key and authtoken
export
NGROK_AUTHTOKEN=YOUR_NGROK_AUTHTOKEN_HERE

helm install ngrok-operator ngrok/ngrok-
operator \
  --namespace ngrok-operator \
  --create-namespace \
  --set credentials.apiKey=$NGROK_API_KEY \
  --set credentials.authtoken=$NGROK_AUTHTOKEN
```

## Docker

```
# Install via Docker
docker pull ngrok/ngrok

# Run ngrok via Docker
docker run --net=host -it -e
NGROK_AUTHTOKEN=xyz ngrok/ngrok:latest http 80
```

## SDKs

```
# Node.js: https://ngrok.com/downloads/node-js
npm install @ngrok/ngrok

# Go: https://ngrok.com/downloads/go
go get golang.ngrok.com/ngrok/v2

# Python: https://ngrok.com/downloads/python
python3 -m pip install ngrok

# Rust: https://ngrok.com/downloads/rust
# Install ngrok-rust package and the required
dependencies
cargo add ngrok -F axum && cargo add axum &&
cargo add tokio -F rt-multi-thread -F macros

# Java: https://ngrok.com/downloads/java
```

# Exposing different kinds of servers

#2

### API Service

```
# Example: API service on localhost:8080
ngrok http 8080
```

### Web App

```
# Example: On localhost:3000
ngrok http 3000
```

### SSH Server

```
# Example: On Port 22
ngrok tcp 22
```

### Postgres Server

```
ngrok tcp 5432
```

### Any service or server on a different machine

```
ngrok http http://192.168.1.50:8080
```

# Troubleshooting

#3

```
ngrok diagnose

# To test IPv6 connectivity
ngrok diagnose --ipv6 true

# To test connectivity between the ngrok agent and all ngrok points of presence
ngrok diagnose --region all

# For a verbose report
ngrok diagnose -w out.txt #OR
ngrok diagnose --write-report out.txt
```

# Add Authentication w/ Traffic Policy

#4

## Create a Traffic Policy file `policy.yaml`

```
nano policy.yaml
```

## Add the OAuth Action with Google

```
# policy.yaml
on_http_request:
  - actions:
      - type: oauth
        config:
          provider: google # OAuth available with Amazon, Facebook,
GitHub, GitLab, Google, LinkedIn, Microsoft, Twitch
```

## Run your endpoint with the Traffic Policy file

```
ngrok http 8080 --traffic-policy-file=policy.yaml
```

## Restrict OAuth to specific emails

```
# policy.yaml
on_http_request:
  - actions:
      - type: oauth
        config:
          provider: google
  - expressions:
      - "!(actions.ngrok.oauth.identity.email in
['alice@example.com','bob@example.com'])"
    actions:
      - type: deny
```

# Add Authentication w/ Traffic Policy

#4

### Restrict OAuth to specific domains

```yaml
# policy.yaml
on_http_request:
  - actions:
      - type: oauth
        config:
          provider: google
  - expressions:
      - "!(actions.ngrok.oauth.identity.email.endsWith('@example.com'))"
    actions:
      - type: deny
```

Read more: https://ngrok.com/docs/traffic-policy/actions/oauth/

# Verify your webhooks

#5

### Add the `verify-webhook` action for Slack

```yaml
# policy.yaml
on_http_request:
  - actions:
      - type: verify-webhook
        config:
          provider: slack
          secret: $SLACK_TOKEN
```

### CLI Alternative

```
ngrok http 3000 \
  --verify-webhook=slack \
  --verify-webhook-secret=$SLACK_TOKEN
```

# Verify your webhooks

#5

Replace **`provider`** for any supported provider

```
# policy.yaml
on_http_request:
  - actions:
      - type: verify-webhook
        config:
          provider: $PROVIDER # Example: GitHub
          secret: $PROVIDER_TOKEN
```

List of Supported Providers:
https://ngrok.com/docs/traffic-policy/actions/verify-webhook/

# Do even more w/ internal endpoints

#6

Create a Cloud Endpoint

```
# Create a private, internal agent endpoint only reachable via forward-internal
ngrok http 8080 --binding=internal --url https://api.internal
```

Example Traffic Policy File

```
# policy.yaml
# Forward to an internal endpoint from a public endpoint
on_http_request:
  - actions:
      - type: forward-internal
        config:
          url: https://api.internal
```

## Do even more w/ internal endpoints

#6

Start Public Endpoint with `forward-internal` action

```
ngrok http 8080 --url forward-internal-example.ngrok.app --traffic-policy-file policy.yml
```

Read more: https://ngrok.com/docs/traffic-policy/actions/forward-internal/

## Manage traffic in other ways

#7

Add path-based routing

```yaml
# policy.yaml
# Route /api/* to api.internal, /app/* to app.internal
on_http_request:
  - expressions:
      - "req.path.startsWith('/api/')"
    actions:
      - type: forward-internal
        config:
          url: https://api.internal
  - expressions:
      - "req.path.startsWith('/app/')"
    actions:
      - type: forward-internal
        config:
          url: https://app.internal
```

# Manage traffic in other ways

#7

## Route traffic by anything

```yaml
# policy.yaml
# Host & header-based dynamic forwarding to internal endpoints
on_http_request:
  - expressions:
      - "req.host == 'api.example.com'"
    actions:
      - type: forward-internal
        config: { url: https://api.internal }
  - expressions:
      - "getReqHeader('X-Tenant') != ''"
    actions:
      - type: forward-internal
        config: { url: https://tenant.internal }
```

## Multiplex to Internal Services from a Single Domain

```yaml
# policy.yaml
on_http_request:
  - actions:
      - type: forward-internal
        config:
          url: https://${req.host.split(".$NGROK_DOMAIN")[0]}.internal
```

## Add rate limiting

```yaml
# policy.yaml
# 10 requests per 60s window per client IP -> 429 on limit
on_http_request:
  - actions:
      - type: rate-limit
        config:
          name: per-ip-60s
          algorithm: sliding_window
          capacity: 10
          rate: "60s"
          bucket_key:
            - conn.client_ip
```

# Manage traffic in other ways

#7

## Block search and AI bots

```yaml
# policy.yaml
# Send a robots.txt denying crawlers
on_http_request:
  - expressions:
      - "req.path == '/robots.txt'"
    actions:
      - type: custom-response
        config:
          status_code: 200
          headers:
            Content-Type: "text/plain"
          body: |
            User-agent: *
            Disallow: /
```

```yaml
# policy.yaml
# Deny common bot/AI user agents
on_http_request:
  - expressions:
      - "req.user_agent.raw.matches('(?i)(gptbot|chatgpt-user|ccbot|
bingbot|googlebot)')"
    actions:
      - type: deny
```

```yaml
# policy.yaml
# Also add X-Robots-Tag to all responses
on_http_response:
  - actions:
      - type: add-headers
        config:
          headers:
            X-Robots-Tag: "noindex, nofollow, noai, noimageai"
```

## Add headers

**Via CLI**

```
# Common security headers
ngrok http 8080 \
  --request-header-add "X-Frame-Options: DENY" \
  --response-header-add "Referrer-Policy: no-referrer"
```

# Manage traffic in other ways

#7

## Add headers

**Via Traffic Policy**

```yaml
# policy.yaml
# Add headers on request/response
on_http_request:
  - actions:
      - type: add-headers
        config:
          headers:
            X-Frame-Options: "DENY"
on_http_response:
  - actions:
      - type: add-headers
        config:
          headers:
            Referrer-Policy: "no-referrer"
```

## Restrict access by IPs

```
# Allow only 203.0.113.0/24; deny others
ngrok http 8080 --cidr-allow 203.0.113.0/24

# Or explicitly deny CIDRs
ngrok http 8080 --cidr-deny 0.0.0.0/0
```

## Block all the potentially bad things

```yaml
# policy.yaml
# Apply OWASP Core Rule Set on requests/responses
on_http_request:
  - actions:
      - type: owasp-crs-request
on_http_response:
  - actions:
      - type: owasp-crs-response
```

# CLI Flags



### URL

```
# Choose a URL instead of random assignment
ngrok http 8080 --url https://baz.ngrok.dev
```

### traffic-policy-file

```
# Manipulate traffic to your endpoint with a traffic policy file
ngrok http 8080 --url https://baz.ngrok.dev --traffic-policy-file
policy.yaml
```

### traffic-policy-url

```
# Manipulate traffic to your endpoint with a traffic policy file
ngrok http 8080 --url https://baz.ngrok.dev policy --traffic-policy-url
https://example.com/policy.yml
```

### pooling-enabled

```
# Load Balance (different ports)
ngrok http 8080 --url https://api.example.com --pooling-enabled
ngrok http 8081 --url https://api.example.com --pooling-enabled
```