Cookbook

Small snippets and pieces of knowledge.

- Local v5 Nuts node with Ngrok
- V6 configuration without V5 features

Local v5 Nuts node with Ngrok

> This documentation only applies to use cases requiring the Nuts gRPC network (did:nuts DIDs). If your use cases uses did:web DIDs (v6 functionality), you can use the development Docker image. Refer to the Docker documentation for more information.

Ideally a Nuts node runs in a data center or cloud environment with a fixed IP address behind a ngress proxy. During development this is not always practical and there exists a need for running a Nuts node locally on for instance a laptop behind a NAT.

In order for the node to receive requests, private transactions and synchronize with other nodes, it must be reachable by external nodes. For this you can use the handy service ngrok. Ngrok describes itself as follows:

> “ ngrok is a simplified API-first ingress-as-a-service that adds connectivity, security, and observability to your apps in one line

1. Setup ngrok

To get started with ngrok, you can follow this tutorial

Once you have an account and access token setup, add the following lines to your ngrok configuration:

```
tunnels:
  grpc:
    proto: tcp
    addr: 5555
  n2n:
    proto: tcp
    addr: 1323
```

This informs the ngrok application to create 2 tunnels, each forwarding tcp traffic to respectively port `5555` for `grpc` traffic and `1323` for `http` traffic.

When you start ngrok with this config, the output will look something like this:

```
Session Status                online
Account                       OrgName (Plan: Free)
Version                       3.2.2
Region                        Europe (eu)
Latency                       15ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    tcp://0.tcp.eu.ngrok.io:16077 -> localhost:5555
Forwarding                    tcp://7.tcp.eu.ngrok.io:18593 -> localhost:1323

Connections                   ttl     opn     rt1     rt5     p50     p90
```

2. Certificates

Your node needs certificates for these endpoints to join a network.

For more information, read the official documentation.

Since ngrok uses the `tcp.eu.ngrok.io` path, you can generate a certificate for the ngrok domains and `stable` network with the following command:

```
$ ./issue-cert.sh stable '*.tcp.eu.ngrok.io'
```

Now, copy the resulting files to a location accessable for your node and add the following section to your `nuts.yaml` config file:

```
tls:
  certfile: "*.tcp.eu.ngrok.io-stable.pem"
  certkeyfile: "*.tcp.eu.ngrok.io-stable.key"
  truststorefile: ./truststore-stable.pem
```

3. Endpoint configuration

Your node needs to broadcast these endpoints to other nodes in order for them to connect to you.

You need to setup a vendor DID document. See step 3 in the official docs. For the `NutsComm` service you will use the tunnel url which maps to port `5555`. The url will become `grpc://0.tcp.eu.ngrok.io:16077` in our example.

When a service uses oauth you must configure an oauth endpoint. Following our example, that would become: `http://7.tcp.eu.ngrok.io:18593/n2n/auth/v1/accesstoken`.

An example of the vendor DID document service section with a `test-service` would look like this:

```
{
    "id": "did:nuts:FJ2WaKNaf6jKZ9tgJoNCmQKh3xoe9vFEBzj59WSkmSEV",
    ...
    "service": [
        {
            "id":
"did:nuts:FJ2WaKNaf6jKZ9tgJoNCmQKh3xoe9vFEBzj59WSkmSEV#AYQhYmNeYdoNthAiyAZ9ufAM54DF2ZqJmzGrTi4dm6nT",
            "serviceEndpoint": "grpc://0.tcp.eu.ngrok.io:16077",
            "type": "NutsComm"
        },
        {
            "id":
"did:nuts:FJ2WaKNaf6jKZ9tgJoNCmQKh3xoe9vFEBzj59WSkmSEV#DtUY29HzYR1H2UofLXVJ3ru6TAxYY59NiBqSyfTAeM5Y",
            "serviceEndpoint": "http://7.tcp.eu.ngrok.io:18593/n2n/auth/v1/accesstoken",
            "type": "oauth"
        },
        {
            "id":
"did:nuts:FJ2WaKNaf6jKZ9tgJoNCmQKh3xoe9vFEBzj59WSkmSEV#7jbTfYq8vjVwvhJt6Nayxzk1ki4xx1Bw1MwhebLoDxau",
            "serviceEndpoint": {
                "oauth":
"did:nuts:FJ2WaKNaf6jKZ9tgJoNCmQKh3xoe9vFEBzj59WSkmSEV/serviceEndpoint?type=oauth"
            },
            "type": "test-service"
        }
    ]
}
```

V6 configuration without V5 features

The configuration below shows a minimal `nuts.yaml` config file:

- `did:nuts` disabled, `did:web` enabled
- discovery refresh of 1 minute for faster discovery during development
- default mappings for discovery and policy files (mounts required)
- SQLite will be used as DB
- gRPC network is not started

```
url: https://<your-domain.com>
verbosity: debug
strictmode: false
internalratelimiter: false
didmethods:
  - web
http:
  log: metadata-and-body
  internal:
    address: :8081
auth:
  contractvalidators:
    - dummy
  irma:
    autoupdateschemas: false
policy:
  directory: /nuts/config/policy
discovery:
  definitions:
    directory: /nuts/config/discovery
  client:
    refreshinterval: 1m
network:
  grpcaddr: ""
```