Docker Compose Installation & Setup Guide.

NS1 Private DNS 1.1 • Revised October 31, 2018

TABLE OF CONTENTS

1. Overview

NS1's Private DNS is a solution that can be installed on customers' networks. It is the software version of NS1's industry-leading SaaS platform, bringing DNS and traffic management capabilities into customers' networks.

1.1. Quick Start Checklist

1.1.1. Environment

- ❏ Confirm the host machine(s) meets minimum requirements
- ❏ Confirm the firewall rules for the host machine(s):
  - ❏ All containers can connect to `data` on port `5353`
  - ❏ Inbound/outbound rules allow SSH on ports `22` / `23`
  - ❏ Inbound/outbound rules allow HTTP/HTTPS on ports `80` / `443`
  - ❏ Inbound/outbound rules allows DNS queries on port `53`
  - ❏ (Optionally) inbound rule allows zone transfers (`xfr`) on port `5400`

1.1.2. Run Containers

- ❏ Download container images to host machines
  - ❏ `sh <(curl -Ls https://raw.githubusercontent.com/ns1/ns1-privatedns/master/get-privatedns.sh -o -) -k APIKEY`
- ❏ Download and (optionally) modify the docker-compose.yml example as needed
  - ❏ `wget https://raw.githubusercontent.com/ns1/ns1-privatedns/master/docker-compose.yml`
- ❏ Execute docker-compose or docker run commands to start the images
  - ❏ `TAG=1.1.0 docker-compose -p privatedns up -d`
- ❏ After approx. 60 seconds, confirm health checks are passing (all checks equal `0`)
  - ❏ `docker exec <container-name> supd health | jq`
  - ❏ All 18 checks for `data` are healthy
  - ❏ All 10 checks for `dns` are healthy
  - ❏ All 4 checks for `web` are healthy
  - ❏ (Optional) All 4 checks for `xfr` are healthy
  - ❏ (Optional) All 6 checks for `cache` are healthy

1.1.3. Bootstrap to Create Orgs / Users / API keys

- ❏ Call bootstrap endpoint to add the first operator
- ❏ Create the first organization
- ❏ Create the first application user in that organization (capture the invite token in the response)
- ❏ The first user (above) visits `https://{{web-host}}/#/invite/{{invite-token>}}` to create a password

2. Before You Begin

This guide assumes the reader has familiarity with the principles of DNS, networking, and working knowledge of Docker and container-based systems.

2.1. System Requirements

The following sections contain both hard requirements and general guidance (e.g. hardware) for the system. For questions or concerns about target environments, please contact support@ns1.com for more information.

2.1.1. Hardware

The following is the minimum hardware recommended to run a single container image:

- 2 CPUs
- 2 GB RAM
- 20 GB of free disk space

Important note: NS1 highly recommends monitoring disk usage on each host to prevent service disruption.

2.1.2. Operating Systems

- Linux Ubuntu 18.04 / 17.04 / 16.04 / 14.04 x64
- CentOS 7 x86
- Red Hat x86_64

Important notes:

1. If running multiple containers on the same host machine, a Linux kernel must support the SO_REUSEPORT socket option to allow multiple sockets on the same host to bind to the same port.
2. Running Private DNS on Mac OS or Windows 10 Pro / Enterprise 64-bit is not supported by NS1.
3. Ubuntu 18.04 installs may require systemd-resolved to be disabled and stopped. To do this, run the following commands:
    a. Disable service.

    ```
    sudo systemctl disable systemd-resolved.service
    ```

    b. Stop the service.

    ```
    sudo service systemd-resolved stop
    ```

2.1.3. Docker

To run multi-container applications with Docker Compose, Private DNS **requires** Docker Version **17.03.x or 17.06.x (CE or EE)** or higher. More information about Docker requirements along with installation steps is found at the following locations:

- Docker: https://docs.docker.com/engine/installation/
- Docker Compose: https://docs.docker.com/compose/install/.

2.1.4. Software

Web interface configuration was designed to work with current Evergreen browser versions. Note that newer versions of each browser may be available since this release and therefore may not be tested yet by NS1. The following browser versions were tested prior to this release:

- Chrome 67.0.3396.87 (Official Build) (64-bit)
- Firefox 58.0
- Edge 41.16299.15
- EdgeHTML 16.16299
- Safari 11.0.2

2.1.5. Network

While configuring the system, IPv4 addresses are recommended. Access to the public internet is required to pull the latest container images from my.nsone.net.

Important notes:

- For private networks without access to the public internet, NS1 recommends pulling the container images on a separate machine first before transferring contents to a hard disk or volume which is accessible by the private network.
- If available, it is highly recommended to add each image version to a private container registry.

The following network protocols are used by the system: HTTP/HTTPS, UDP, TCP, TLS.

2.2. Securing System Access

Host Access. Access to host machines on which each container is running should be secured to system administrators only. NS1 *highly* recommends that administrators and operators should be required to access this machine via secure shell (SSH) connection.

Basic Auth. To further secure containers' web configuration interface, operators must sign in using "Basic Auth" credentials. By default, Private DNS initializes Username to `ns1` and Password to `private`.

Limiting Port Visibility. The web container should only be viewable on ports 80 and 443 from the localhost.

2.3. Transport Layer Security

By default all network communication between the various Private DNS containers, management portal and API endpoints, as well as the main portal and API endpoints is encrypted using self-signed certificates that are generated at initial runtime of each container. It is expected behavior for a browser certificate trust warning to appear when navigating to the web configuration interfaces of each container for the first time. Proceed following the on screen browser instructions.

There are up to three different TLS configurations depending on the container:

TLS Configuration	Description	Availability
`Transport TLS`	`This configuration manages how communication between the various containers is encrypted and verified.`	● Data ● Web ● Xfr ● Dns ● Cache
`Management TLS`	`This configuration manages how communication between clients and the management interface is encrypted and verified.`	● Data ● Web ● Xfr ● Dns ● Cache
`Web TLS`	`This configuration manages how communication between clients and the main NS1 portal and API is encrypted and verified.`	● Web

2.3.1. Certificate Files

In each of the above listed TLS configurations Transport, Management, Web, there are two available certificates that can be uploaded. The certificate bundle and the certificate revocation list (CRL) certificate.

The certificate bundle is a combination of standard TLS files used to configure the Private DNS networking components. It contains the following components in PEM format:

- The public root certificate authority (CA) certificate.

- Any intermediary certificate authority (CA) certificates used by the root CA.
- The public server certificate.
- The private server key.

The bundle itself is created by concatenating the above files into a single file which would end up looking something like this example bundle. Also please see this script file for an example on creating custom certificates via OpenSSL and BASH.

The certificate revocation list (CRL) certificate is an advanced configuration option allowing the ability to manually supply a means of revoking server and client certificates from being trusted. Please see your CA about how to generate and use this file. Note that if your CA certificate has a CRL location specified it will be used rendering manual upload of this file unnecessary.

2.3.2. Configuration Options

There are a few configuration options for each of the different TLS configuration sections detailed below:

2.3.2.1. Certificate Bundle

This is the certificate bundle to utilize for the specified TLS configuration. And is a combined PEM format certificate detailed above in section 2.3.1.

Available: Transport, Management, Web TLS configurations

2.3.2.2. CRL Certificate

This is the certificate bundle to utilize for the specified TLS configuration. And is a combined PEM format certificate detailed above.

Available: Transport, Management, Web TLS configurations

2.3.2.3. Web portal/api force TLS redirect

This option allows for either forcing a redirect to https from http or allowing http plain text traffic to the primary NS1 portal and api.

Available: Web TLS configurations

2.3.2.4. TLS Certificate Zone

This is the zone name used in the certificate used for transport networking. This is explained below in section 2.3.3.

Available: Transport TLS configurations

2.3.2.5. Strict Certificate Verification

This enforces full verification of client and server certificates. Note that depending on the configuration it means different things:

Transport TLS

Setting this to true means the “TLS Certificate Zone” setting **must** be properly configured.

Management and Web TLS

Setting this to true means that all interactions with the portal/api will require a client certificate trusted by the “Certificate Bundle” to be supplied.

Available: Transport, Management, Web TLS configurations

2.3.2.6. Minimum TLS Version

This enforces a minimum TLS version that can be used. Note that we do not support anything bellow “TLSv1.0” or above “TLSv1.2”.

Available: Transport, Management, Web TLS configurations

2.3.2.7. Allowed Cipher List

This is an advanced configuration option allowing to manually specify the TLS ciphers that are allowed to be used.

Available: Transport, Management, Web TLS configurations

2.3.3. Transport TLS Configuration

The transport TLS configuration is unique in Private DNS, in that it has some specific requirements for the certificate that is used and the configuration supplied.

Important notes:

- The certificate bundle used for transport TLS **must** be a wildcard certificate. Meaning that the common name or one of the subject alternate names must be in the form of `*.ns1.private`.
- If “Strict Certificate Verification” is enabled, the zone name used in the wildcard can be anything but it **must** be specified in the “TLS Certificate Zone” configuration when the certificate is uploaded. Failing to do this will mean that none of the containers participating in the Private DNS cluster will be able to communicate.

A second requirement is that all containers have certificate bundles utilizing the **same** certificate authority as the signer. If different certificate authorities are used to sign the various transport certificate bundles, verification will fail and none of the containers participating in the Private DNS cluster will be able to communicate.

3. Installation

This section provides instructions on installation using docker-compose.

3.1. Accessing Files & Images

Additional utilities and resources are found in the following Github repository: https://github.com/ns1/ns1-privatedns

When onboarding, NS1's Customer Success team will send an email invitation to Private DNS customers directing them to https://my.nsone.net/. Once a password is created, navigate to the Downloads page.

Container images can now be downloaded locally, unzipped, and pushed to a private container registry.

Alternatively, use your NS1 Managed DNS API key to run the following command or download and run the command here:

```
sh <(curl -Ls
https://raw.githubusercontent.com/ns1/ns1-privatedns/master/get-privatedns.sh -o -) -k
APIKEY
```

This script will download the images locally and load them into the docker volume store. Once this is complete, download the desired compose file here. For example:

```
wget
https://raw.githubusercontent.com/ns1/ns1-privatedns/master/docker-compose/docker-comp
ose.yml
```

Important note: It is a violation of the NS1 license agreement to push container images to a repository which is *not* for exclusive use by the licensee. Pushing images to a public registry is unlawful redistribution of the software.

3.2. Using docker-compose.yml

The example `docker-compose.yml` is used to start all container images on a single host: data, dns, web, xfr, and cache. This compose file is best used to get familiar with the system. It is not advisable to use this file in production.

For example:

```
 sudo TAG=1.1.0 POP_ID=local SERVER_ID=localvm docker-compose -p mytest -f
 docker-compose.yml up -d
```

Options	Description
TAG	The image tag, or version number, of the container images; defaults to 1.1.0
POP_ID	Specifies the location (datacenter/pop) of the server where the data container is running
SERVER_ID:	Identifies a specific server in a location where the data container is running

Important note: Upgrading the host machine's version of Docker Compose is recommended. See https://docs.docker.com/compose/install/#upgrading for more information.

3.3. Using core-compose.yml and edge-compose.yml

Production and production-like deployment topologies generally follow a "hub and spoke" pattern where certain services can be grouped in the "hub" referred to here as core; other services on the "spoke", which we referred to as edge. These compose files are meant to assist operators in deploying to production or production-like environments more easily.

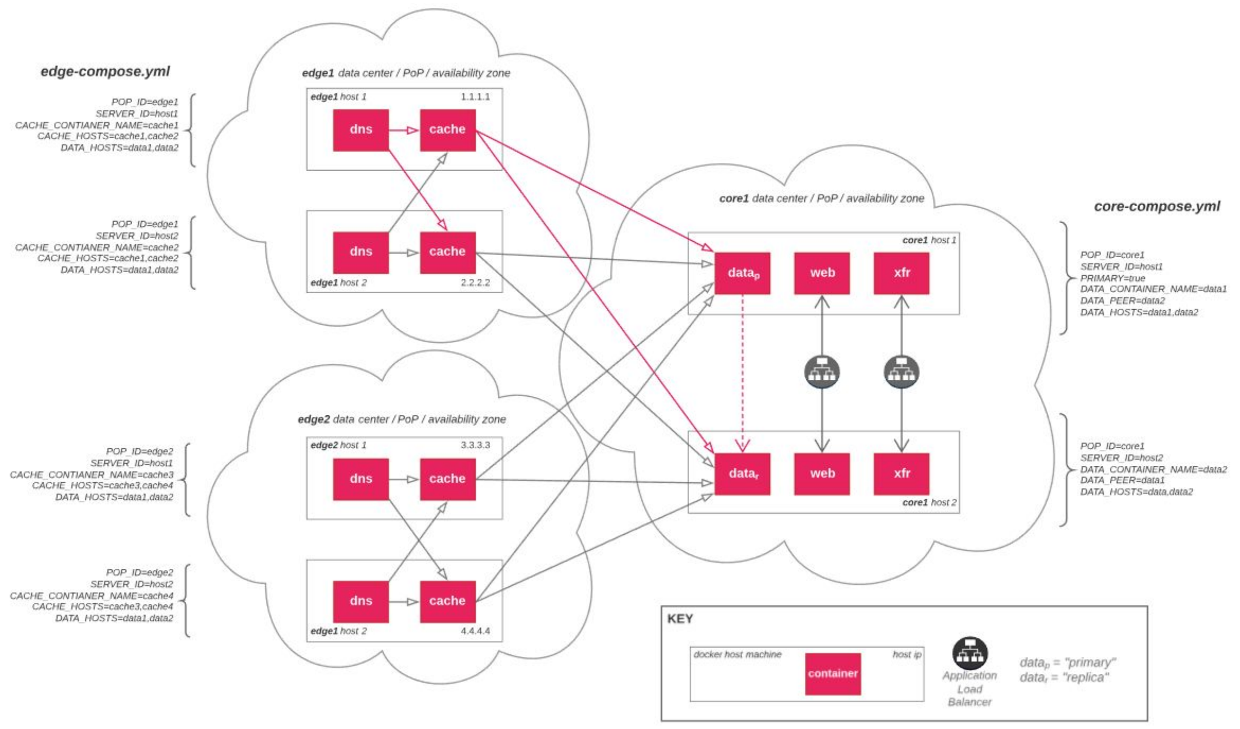


Figure 1. Example topology with core-compose.yml and edge-compose.yml command line variables for reference.

3.3.1. core-compose.yml

Used to start core services on a single host: data, web, xfr. For example, starting a core services host with primary data:

```
sudo TAG=1.1.0 POP_ID=nyc SERVER_ID=core1 PRIMARY=true DATA_CONTAINER_NAME=data1
DATA_PEERS=data2 DATA_HOSTS=data1,data2 docker-compose -p myproject -f
core-compose.yml up -d
```

And alternatively, starting a core services host with replica data:

```
sudo TAG=1.1.0 POP_ID=nyc SERVER_ID=core2 DATA_CONTAINER_NAME=data2 DATA_PEERS=data1
DATA_HOSTS=data1,data2 docker-compose -p myproject -f core-compose.yml up -d
```

Options	Description
TAG	The image tag, or version number, of the container images; defaults to 1.1.0
PRIMARY	Skip if the host's data container will operate as a Replica or set this variable to true if the host's data container will operate as Primary in a Primary-Replica configuration; defaults to null
POP_ID	Specifies the location (datacenter/pop) of the server where the data container is running; defaults to mypop
SERVER_ID	Identifies a specific server in a location where the data container is running; defaults to myserver
DATA_PEERS	Identifies the peer(s) of this host's data container with one operating as primary and the other replica
DATA_HOSTS	Series of comma delimited hostnames of data containers e.g. data1,data2; defaults to data
DATA_CONTAINER_NAME	Sets the container's name; defaults to data
WEB_CONTAINER_NAME	Sets the container's name; defaults to web
XFR_CONTAINER_NAME	Sets the container's name; defaults to xfr
API_HOSTNAME	Hostname for the api and feed URLs; defaults to localhost
PORTAL_HOSTNAME	Hostname for the portal; defaults to localhost
NAMESERVERS	Nameservers used in SOA records; defaults to ns1.mycompany.net
HOSTMASTER_EMAIL	Hostmaster email address for SOA records; defaults to hostmaster@mycompany.net

3.3.2. edge-compose.yml

Used to start edge services on a single host: dns and cache. For example:

```
sudo TAG=1.1.0 POP_ID=nyc SERVER_ID=edge1 docker-compose -p myproject -f
edge-compose.yml up -d
```

Options	Description
TAG	The image tag, or version number, of the container images; defaults to 1.1.0
POP_ID	Specifies the location (datacenter/pop) of the server

	where the data container is running; defaults to mypop
SERVER_ID	Identifies a specific server in a location where the data container is running; defaults to myserver
DNS_CONTAINER_NAME	Sets the container's name; defaults to dns
CACHE_CONTAINER_NAME	Sets the container's name; defaults to cache
CACHE_HOSTS	Series of comma delimited hostnames of cache containers e.g. cache1,cache2; defaults to cache
DATA_HOSTS	Series of comma delimited hostnames of data containers e.g. data1,data2; defaults to data
DNS_OP_MODE	authoritative/recursive; the mode of operation for this dns container; defaults to authoritative

4. User Setup

Once the web and data containers are available and healthy, users can be added to the system. Every end user is generally classified as one of two types: “operators” or “application users” or just “users” for short. Operators act as administrators on behalf of any tenant in the system where users within a tenant organization only have access to that organization’s resources (e.g. zones, records, API keys, etc.).

4.1. Bootstrap an Operator

When no operator users exist in the main database, the bootstrap endpoint is enabled. Below is an example cURL command. Read more in Appendix A: Managing Operators.

```
curl -X POST \
  https://{{web-host}}/v1/ops/bootstrap \
  -H 'Content-Type: application/json' \
  -d '{
      "user": "root",
      "name": "Root Operator",
      "email": "ops@example.com",
      "password": "rootpassword"
}'
```

Important Notes:

- Bootstrap can only be performed once.
- Save the API Key (`key`) and 2FA Secret (`secret`) of the first operator created during bootstrap. If the API Key is lost and no other operators are created yet, the system is rendered unmanageable and a new instance must be deployed.
- The options `-k` or `--insecure` are needed on a cURL command if a valid SSL/TLS certificate has not been uploaded for use by the web container. See Transport Layer Security for more information about adding custom certificates.
- Password length should be between 10 and 255 characters.

If the operator was successfully added, a JSON response will provide information such as the 2FA Secret and API Key. For example,

```
{
"two_factor_auth": { "secret": "{{2fasecret}}", "type": "totp" },
"name": "Operator Name",
"id": "5b11a509d4f9fa0e5ec1373d",
"user": "operator",
"key": "{{operator_api_key}}",
"last_access": "2018-06-01T19:56:56.604228",
"password": "L0R3M1PSUm",
"email": "operator@example.com"
```

```
      }
```

4.2. Create an Organization

An existing organization is required to manage resources like zones and records. Read more in Appendix B: Managing Organizations.

To create the first organization, an operator can make the following cURL command:

```
curl -X PUT \
'https://{{web-host}}/v1/ops/orgs' \
-H 'X-NSONE-Key: {{operator_api_key}}' \
-d '{"name": "Organization Name"}'
```

In response, an `org_id` is created along with a unique object `id`. The first organization in the Private DNS instance is set to 2000.

```
{
    "org_id": 2000,
    "name": "Organization Name",
    "id": "5b16cb703ca56f"
}
```

4.3. Create an Application User

An operator can add application users to a tenant organization with the following cURL command:

```
curl -X PUT \
'https://{{web-host}}/v1/account/users/newuser' \
-H 'X-NSONE-Key: {{operator_api_key}}!{{org_id}}' \
-d '{"username":"newuser", "name":"New User","email":"newuser@exmple.com"}'
```

In response an invite token is generated for this user to set a password:

```
{
    "username": "newuser",
    "notify": {
        "billing": false
    },
    "permissions": {},
    "invite_token": "{{new_invite_token}}",
    "ip_whitelist_strict": false,
    "name": "New User",
    "teams": [],
    "ip_whitelist": [],
    "last_access": null,
    "2fa_enabled": false,
    "email": "newuser@example.com"
}
```

An operator can then send an invitation URL to the application user for setting an initial password.

```
https://{{web-host}}/#/invite/{{new_invite_token}}
```

5. Container Configuration

When first setting up the system, the data container(s) should be configured first. Configuration of other containers can be done in any order.

5.1. Configuration with CLI

Using the command line, the system is set up using the supd application. For usage information of supd, add the `--help` or `-h` option. For example, the xfr container's supd help is accessed with the following command:

```
docker exec privatedns_xfr_1 supd --help
```

5.1.1. Common CLI Configuration Options

Certain fields are common and shared among all containers. To view the usage information of the run command for the specific container. For example,

```
docker exec privatedns_data_1 supd run --help
```

The following table describes common options for the run command.

Options	Description
`--recovery`	Used by the system to recover from a failed configuration
`--bootstrap`	Used by the system when first standing up with no configuration available
`-f, --force`	overwrite any existing config options with CLI parameters
`--pop_id [value]`	This specifies the location (datacenter/pop) of the server where the data container is running

--server_id [value]	This identifies a specific server in a location where the data container is running
--data_host	- Comma delimited list of hostnames or IP addresses of every data container in the cluster - For data containers, this should only list peers and exclude its own hostname/IP address - For dns containers, this list can be a combination of data and/or cache containers
--enable_ops_metrics [value]	If enabled this will export system metrics to the specified metrics database(s)
--transport_enable_tls [value]	Enable or disable transport TLS encryption, to use for communicating with other cluster members. (defaults to True)
--default_transport_tls_settings [value]	Use the predefined TLS settings for transport networking
--manual_transport_tls_settings [value]	Define the TLS settings for transport networking, when using manually mounted certificates
--manual_transport_tls_crl_file [value]	The CRL certificate to use for verifying client certificates revocation status, when communicating between cluster members
--manual_transport_tls_sni_zone [value]	The zone to use for TLS verification purposes, when communicating between cluster members
--manual_transport_tls_client_cert_verfiy [value]	Whether or not to enforce strict certificate verification, when communicating between cluster members. NOTE: You must set 'TLS certificate zone' for this to work with custom certificates
--manual_transport_tls_version [value]	The minimum TLS version to enforce, when communicating between cluster members
--manual_transport_tls_ciphers [value]	The set of TLS ciphers to use, when communicating between cluster members
--management_enable_tls [value]	Enable or disable management TLS encryption, to use for communicating with other cluster members. (defaults to True)
--default_management_tls_settings [value]	Use the predefined TLS settings for management networking
--manual_management_tls_settings [value]	Define the TLS settings for management networking, when using manually mounted certificates
--manual_management	The combined certificate/key PEM file to present, when

_tls_cert_bundle [value]	communicating between cluster members
--manual_management_tls_crl_file [value]	The CRL certificate to use for verifying client certificates revocation status, when communicating between cluster members
--manual_management_tls_client_cert_verfiy [value]	Whether or not to enforce strict certificate verification, when communicating between cluster members
--manual_management_tls_version [value]	The minimum TLS version to enforce, when communicating between cluster members
--manual_management_tls_ciphers [value]	The set of TLS ciphers to use, when communicating between cluster members
-h, --help	output usage information

5.1.2. Data CLI Configuration

To see the data container's unique configuration options, view the usage information of the run command.

```
docker exec privatedns_data_1 supd run --help
```

The following table describes options specific to the data container.

Options	Description
`--number_of_stats_processors [value]`	The number of processors to run for query metrics
`--query_metrics_ttl [value]`	Duration to keep query metrics.
`--telegraf_output_ns1_data [value]`	None
`--telegraf_output_ns1_data_enabled [value]`	Use the default metrics store
`--telegraf_output_influxdb [value]`	Enable InfluxDB output for operational metrics
`--telegraf_output_influxdb_data_host [value]`	Hostname of InfluxDB host
`--telegraf_output_influxdb_database [value]`	InfluxDB Database Name
`--telegraf_output_influxdb_enable_auth [value]`	Enable user/password authentication for InfluxDB
`--telegraf_output_influxdb_username [value]`	InfluxDB User
`--telegraf_output_influxdb_password [value]`	InfluxDB Password
`--telegraf_output_influxdb_enable_tls [value]`	Enable TLS for InfluxDB
`--telegraf_output_influxdb_server_pem_key [value]`	The PEM key for the InfluxDB server
`--telegraf_output_influxdb_ca_pem_key [value]`	The CA key used for validation if provided
`--telegraf_output_influxdb_crt [value]`	The CRT for the InfluxDB server
`--telegraf_output_elasticsearch [value]`	Enable Elasticsearch output for operational metrics
`--telegraf_output_elasticsearch_data_host [value]`	Elasticsearch Host

--telegraf_output_elasticsearch_sniff_cluster [value]	Enable sniffing Elasticsearch for additional hosts in cluster
--telegraf_output_elasticsearch_timeout [value]	Timeout for Connecting to Elasticsearch
--telegraf_output_elasticsearch_index [value]	Elasticsearch Index
--telegraf_output_elasticsearch_enable_auth [value]	Enable user/password authentication for Elasticsearch
--telegraf_output_elasticsearch_username [value]	Elasticsearch Username
--telegraf_output_elasticsearch_password [value]	Elasticsearch Password
--telegraf_output_elasticsearch_enable_tls [value]	Enable TLs for Elasticsearch
--telegraf_output_elasticsearch_server_pem_key [value]	The PEM key for the Elasticsearch server
--telegraf_output_elasticsearch_ca_pem_key [value]	The CA key used for validation of Elasticsearch
--telegraf_output_elasticsearch_crt [value]	The CRT for the Elasticsearch server
--telegraf_output_elasticsearch_enable_managed_template [value]	Whether to have the metrics process or Elasticsearch manage the template
--telegraf_output_elasticsearch_template_name [value]	The name of the elasticsearch template
--telegraf_output_elasticsearch_overwrite_template [value]	Whether to be able to overwrite an existing template
--telegraf_output_opentsdb [value]	Enable OpenTSDB output for operational metrics
--telegraf_output_opentsdb_host [value]	OpenTSDB Hostname
--telegraf_output_opentsdb_batch_size [value]	Batch Size of writes to OpenTSDB
--telegraf_output_graphite [value]	Enable graphite output for operational metrics
--telegraf_output_graphite_host [value]	Graphite server hostname
--telegraf_output_graphite_timeout [value]	Timeout for connecting to graphite

`--telegraf_output_graphite_template [value]`	`Template format for metrics in graphite`
`--telegraf_output_graphite_enable_tls [value]`	`Enable TLS for graphite`
`--telegraf_output_graphite_server_pem_key [value]`	`The PEM key for the graphite server`
`--telegraf_output_graphite_ca_pem_key [value]`	`The CA PEM key used for validation of graphite server`
`--telegraf_output_graphite_crt [value]`	`The CRT for the graphite server`
`--telegraf_output_prometheus [value]`	`Export metrics to Prometheus`
`--telegraf_output_prometheus_port [value]`	`Port to listen on for requests from a Prometheus Server.`
`--telegraf_output_prometheus_enable_tls [value]`	`Whether to use TLS on Prometheus Connection`
`--telegraf_output_prometheus_tls_crt [value]`	`Cert for Prometheus TLS.`
`--telegraf_output_prometheus_tls_key [value]`	`Cert Key for Prometheus TLS.`
`--telegraf_output_prometheus_tls_version [value]`	`TLS Version for Prometheus`
`--telegraf_output_prometheus_tls_ciphers [value]`	`Cipher suite for Prometheus Export.`
`--telegraf_output_prometheus_enable_auth [value]`	`Enable Basic Auth on Prometheus`
`--telegraf_output_prometheus_username [value]`	`Username for basic auth with Prometheus.`
`--telegraf_output_prometheus_password [value]`	`Password for basic auth with Prometheus.`

5.1.3. DNS CLI Configuration

To see the dns container's unique configuration options, view the usage information of the run command.

```
docker exec privatedns_dns_1 supd run --help
```

The following table describes options specific to the dns container.

Options	Description

--data_host [value]	Comma separated list or array of hostnames or IP addresses of data and/or cache containers
--num_metrics_procs [value]	The number of processors running to collect metrics for DNS queries
--metrics_flush_interval [value]	Interval for flushing metrics to the data container
--dns_recv_buffer [value]	The size in bytes of the UDP receive buffers used by the dns server. A setting of '0' uses the OS defined default value.
--num_trex_procs [value]	Number of Authoritative DNS server processes to run
--enable_ops_metrics [value]	Whether to send operational metrics to the data container
--operation_mode [value]	Controls whether the server will act as **authoritative** server or **recursive** resolver
--forward_zones_recurse [value]	List of zones and recursive server addresses. Any query for a name in the configured zone will be forwarded to the corresponding recursive server for full resolution. Use this option if you want queries for certain names to be answered by a different resolver
--forward_zones [value]	List of zones and authoritative name server addresses. Any query for a name in the configured zone will be sent to the corresponding authoritative server but the full query resolution will happen locally. Use this option to configure resolution of domains which are not accessible from public internet

5.1.4. Web CLI Configuration

To see the web container's unique configuration options, view the usage information of the run command.

```
docker exec privatedns_web_1 supd run --help
```

The following table describes options specific to the web container.

Options	Description
--data_host [value]	Comma separated list or array of hostnames or IP addresses of data containers
--api_hostname [value]	Hostname to use for API calls (bootstrap, organization management require this to be set)
--portal_hostname [value]	Hostname to use for portal access

`--web_enable_tls [value]`	Enable or disable web TLS encryption, to use for communicating with the portal interface and api. (defaults to True)
`--default_web_tls_settings [value]`	Use the predefined TLS settings for web networking.
`--manual_web_tls_settings [value]`	Define the TLS settings for web networking, when using manually mounted certificates.
`--manual_web_tls_cert_bundle [value]`	The combined certificate/key PEM file to present, when communicating between cluster members.
`--manual_web_tls_crl_file [value]`	The CRL certificate to use for verifying client certificates revocation status, when communicating between cluster members.
`--manual_web_tls_force_redir [value]`	Enable or disable forcing redirect of http traffic to https, for the web facing portal and api. (defaults to True)
`--manual_web_tls_client_cert_verfiy [value]`	Whether or not to enforce strict certificate verification, when communicating between cluster members.
`--manual_web_tls_version [value]`	The minimum TLS version to enforce, when communicating between cluster members.
`--manual_web_tls_ciphers [value]`	The set of TLS ciphers to use, when communicating between cluster members.
`--nameservers [value]`	List of default name servers to set as NS records for new zones. The first server in the list is also used as primary name server in the SOA record. For example, ['ns1.example.net', 'ns2.example.net'].
`--hostmaster_email [value]`	E-mail address to set as hostmaster in the SOA record for new zones.

5.1.5. XFR CLI Configuration

To see the xfr container's unique configuration options, view the usage information of the run command.

```
docker exec privatedns_xfr_1 supd run --help
```

The list of common CLI configuration options describes every option for the xfr container.

5.1.6. Cache (Data Cache) CLI Configuration

To see the cache container's unique configuration options, view the usage information of the run command.

```
docker exec privatedns_cache_1 supd run --help
```

The list of common CLI configuration options describes every option for the cache container.

5.2. Configuration with Web Interface

Each container has a web interface for manual configuration.

5.2.1. Data Container Configuration

To set up the data container via web interface:

1. Open a web browser to `https://<data-hostname-ip>:3300`

   Note: The configuration port 3300 is set in the docker-compose file.

Optional configurations include:

2. Under the Identifiers grouping, enter a **Location ID** (datacenter/pop) of the server where the container is run and **Server ID** to identify the specific host machine of the container.
3. Under the Operational Details grouping, enter the **Number of Metrics Processors** and the **Query Metrics TTL** to determine how long query metrics are retained.
4. **Enable Metrics Export** is configured to collect operational telemetry and application metrics locally by default. Other export options allow metrics to also feed into the following external data stores: InfluxDB, Elasticsearch, OpenTSDB, and Graphite.
   a. Each additional option besides of Default Internal Metrics requires additional parameters to ensure successful connection to the external data store.
   b. **InfluxDB** requires a Host, Database Name, Username, InfluxDB Password and SSL/TLS key information to securely connect.
   c. **Elasticsearch** cluster requires Hosts, Timeout, Index Name, Username, Password, optional Template, and SSL/TLS information to securely connect.
   d. **OpenTSDB** export requires a Host and Batch Size.
   e. **Graphite** requires a Host, Timeout, Template, and SSL/TLS information to securely connect.
5. Under the TLS Settings grouping, choosing to use **Defaults** for **Transport TLS** and **Management TLS** will operate with the system's self-signed certificates which were generated on startup.
   a. Manual Settings will require either bind mounting the certificates to Xyz or uploading a certificate bundle by visiting the **Certs and Files** page.

Important note: If Enable Metrics Export is enabled on other containers, they will send operational metrics back to data for aggregation and export to other data stores.

5.2.2. DNS Container Configuration

Before continuing with the dns container(s), the data container must be configured already. For each dns container, to set up the dns container via web interface:

1. Open a web browser to `https://<dns-hostname-ip>:3301`
2. Under the Data Container Connection grouping, enter the **Data Host**.
3. To configure metrics settings under the Operational Details grouping, choose a **Number of Query Metrics Processors**. NS1 recommends at least 2.
4. Enter a **Metrics Flush Interval** to determine at what interval metrics get sent back to data. NS1 recommends 10s for this value.
5. Enter a **DNS UDP Receive Buffer Length** in bytes. A setting of '0' uses the OS defined default value.
6. Enter the **Number of Authoritative DNS Servers**. NS1 recommends at least 2.
7. Under Resolving Configuration, toggle whether the container will operate as an **Authoritative Server** or **Recursive Resolver**.

Optional configurations include:

8. Entering a **Location ID** (datacenter/pop) of the server where the container is run
9. **Server ID** to identify the specific host machine of the container
10. An **External Resolver**'s IP can be set to handle ALIAS record resolution and also queries for non-authoritative zones in Recursive Resolver mode. By default, the container's built in resolver will be used if this value is not specified.
11. **Forwarded Zones** can be added individually with a Domain and one or more forwarding IP Addresses and Port.
12. Under the TLS Settings grouping, choosing to use **Defaults** for **Transport TLS** and **Management TLS** will operate with the system's self-signed certificates which were generated on startup.
    a. Manual Settings will require either bind mounting the certificates to Xyz or uploading a certificate bundle by visiting the **Certs and Files** page.

5.2.3. Web Container Configuration

Before continuing with the web container, the data container must be configured already. To set up the web container via web interface:

1. Open a web browser to `https://<web-hostname-ip>:3302`
2. Under the Data Container Connection grouping, enter the **Data Host**.
3. Enter an **API Hostname** which will be used for feed URLs.
4. Enter a **Portal Hostname** at which users can reach the NS1 Portal running in the web container.

Optional configurations include:

5. Entering a **Location ID** (datacenter/pop) of the server where the container is run
6. **Server ID** to identify the specific host machine of the container.
7. Under the Operational Details grouping, one more more **Nameservers** can be added along with a **Hostmaster E-mail** address to be associated with new zones' SOA records.
8. Under the TLS Settings grouping, choosing to use **Defaults** for **Transport TLS**, **Management TLS** , and **Web TLS** will operate with the system's self-signed certificates which were generated on startup.
    a. Manual Settings will require either bind mounting the certificates to Xyz or uploading a certificate bundle by visiting the **Certs and Files** page.

5.2.4. XFR Container Configuration

Before continuing with the xfr container, the data container must be configured already. To set up the xfr container via web interface:

1. Open a web browser to `https://<xfr-hostname-ip>:3303`
2. Under the Data Container Connection grouping, enter the **Data Host**.

Optional configurations include:

3. Entering a **Location ID** (datacenter/pop) of the server where the container is run
4. **Server ID** to identify the specific host machine of the container
5. Under the TLS Settings grouping, choosing to use **Defaults** for **Transport TLS**, **Management TLS** , and **Web TLS** will operate with the system's self-signed certificates which were generated on startup.
    a. Manual Settings will require either bind mounting the certificates to Xyz or uploading a certificate bundle by visiting the **Certs and Files** page.

5.2.5. Cache Container Configuration

Before continuing with the cache container, the data container must be configured already. To set up the cache container via web interface:

1. Open a web browser to `https://<cache-hostname-ip>:3304`
2. Under the Data Container Connection grouping, enter the **Data Host**.

Optional configurations include:

3. Entering a **Location ID** (datacenter/pop) of the server where the container is run
4. **Server ID** to identify the specific host machine of the container

5. Under the TLS Settings grouping, choosing to use **Defaults** for **Transport TLS**, **Management TLS** , and **Web TLS** will operate with the system's self-signed certificates which were generated on startup.
    a. Manual Settings will require either bind mounting the certificates to Xyz or uploading a certificate bundle by visiting the **Certs and Files** page.

6. Configuring for Resiliency

In a typical Private DNS deployment, core services (web, xfr, and data containers) are run in a control or "core" node. In an on-premise deployment, the system's core node is typically deployed to the enterprise's primary data center. There can be (and usually are) more than one dns container running at the system's edge in "edge nodes". For local development or a proof of concept deployment, it is possible to run all containers on one host machine; however, it is recommended that *at least* dns be on its own host.

6.1. Primary-Replica Configuration of Data

Multiple data containers can be run concurrently to provide resiliency and support disaster recovery. For correct failover configuration, data containers in the system should know about peers, which are specified with a comma separated list in the **Data Host** (data_host) configuration field.

6.1.1. Configure Primary (or Replica) with CLI

To check the current operating mode of the data container, execute the following:

```
sudo docker exec wip_data_1 supd primary
```

To configure a data container to run as a primary, execute the following:

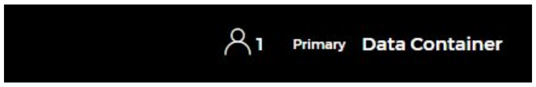
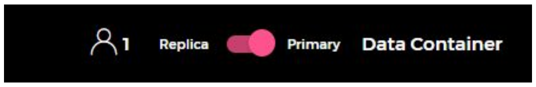
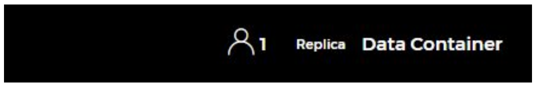
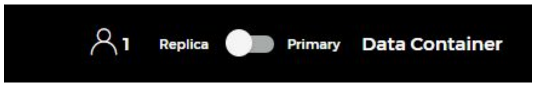
```
sudo docker exec wip_data_1 supd primary true
```

To configure a data container to run as a replica, execute the following:

```
sudo docker exec wip_data_1 supd primary false
```

6.1.2. Configure Primary or Replica with Web Interface

1. Open a web browser to `https://<data-hostname-ip>:3300`
2. Toggle visibility of the slider by pressing the hotkey safety (**?** or **Shift** + **/**).
3. When promoting a Replica to Primary, confirm your selection by selecting **Continue**.

	Hotkey Safety "On"	Hotkey Safety "Off"
Primary		
Replica		

Important note:

- Only one data container should operate as primary at any given time or API calls will commit data randomly to one main database or the other.
- It is possible to have no data containers acting as primary during a failover scenario. In this event no source of data is definitively authoritative. For this reason, the API and portal are not accessible. The system stands by until one of the replicas is promoted to primary by a system operator. This is reflected in the health statuses of the web container.

6.2. Redundant DNS, XFR, & Web Containers

Achieving highly available services for dns, xfr, cache, and web containers is possible in several ways depending on business objectives and SLAs. For example, an application load balancer can monitor container health and distribute load between two or more web containers.

When initialized, each dns container fetches and caches authoritative zones and records; therefore, dns containers can continue serving DNS responses even if the system core services are unavailable. Depending on organization resources, prior experience, and preferences, network administrators typically choose one of the following strategies to achieve redundancy and high availability of DNS:

1. Anycasting the network and advertising the same DNS server addresses over BGP
2. Using application- or appliance-based load balancers
3. Modifying resolv.conf to point at DNS servers running on different hosts

Important note: No matter the chosen strategy, when setting up more than one DNS service in multiple geographically distributed DNS nodes, filling in the Location ID and Server ID should be used to properly identify the host machine.

7. Administrative Actions

Administrative actions are included with each container for operations such as restarting a service or creating backups.

7.1. Admin Actions with CLI

Actions can be executed on the command line for each container.

7.1.1. Common CLI Actions

Certain actions are common and shared among all containers. For example, to view the usage information of the data contianer's supd actions, execute:

```
docker exec privatedns_data_1 supd -help
```

The following table describes common actions.

Commands	Description
`generate_runtime_logs`	Generates an archive of runtime reports in /ns1/data/log/health/
`update_basic_auth [options]`	Sets and updates your basic authorization credentials to the container's configuration UI
`restart_proxy`	This will cause an outage for all service connections. Refresh this configuration page when the action is successful.
`health [options]`	run all health checks. returns a json-formatted object in k:v format
`version`	print the standalone version number. You can also use -V. versionfile is located at /etc/version
`run [options]`	run ansible via flags or environment variables. flags supercede environment vars. Defaults to ignoring existing config.yml
`viewconfig [options]`	Lists all current config parameters; adding -a will display all available parameters, even null values

7.1.2. Data CLI Actions

To see available actions of the dns container, view the usage information of supd.

```
docker exec privatedns_data_1 supd --help
```

The following table describes action commands.

Commands	Description
`primary [value]`	`Sets isPrimary flag if value passed, otherwise returns isPrimary value; acceptable vals are true and false`
`backup_db`	- `Dumps database to ns1/data/backup/<timestamp>.tgz` - `Previous backup files are sent to ns1/data/backup/archived`
`restore_db`	`Restores database from last backup file present in the ns1/data/backup directory`
`restart_maindb`	`This will cause an outage for API and XFR (zone transfer) services.`
`restart_metricsdb`	`This will cause an outage for metrics.`
`restart_msg_svs`	`This will cause an outage for data container services.`
`restart_metrics_svs`	`This will cause an outage for operational metrics.`
`restart_in_mem_db`	`This will cause an outage for creation of portal sessions`

7.1.3. DNS CLI Actions

To see available actions of the dns container, view the usage information of supd.

```
docker exec privatedns_dns_1 supd --help
```

The following table describes action commands.

Commands	Description
`restart_dns`	`This will cause an outage for DNS.`
`restart_caching`	`This will cause DNS caching to be delayed.`
`restart_metrics`	`This will cause an outage for metrics reporting.`
`restart_resolver`	`This will cause an outage for DNS`
`restart_resolver_cache`	`This will reload the auth and zone caches`

7.1.4. Web CLI Actions

To see available actions of the the web container, view the usage information of supd.

```
docker exec privatedns_web_1 supd --help
```

The following table describes action commands.

Commands	Description
`restart_apid`	Restarting the API Service will cause an outage for the API.
`restart_http_server`	Restarting the HTTP Server Service

7.1.5. XFR CLI Actions

To see available actions of the xfr container, view the usage information of supd.

```
docker exec privatedns_xfr_1 supd --help
```

The following table describes action commands.

Commands	Description
`restart_xfr_svs`	This action restarts zone transfer services. Zone transfers will be unavailable during the restart.

7.1.6. Cache (Data Cache) CLI Actions

To see available actions of the xfr container, view the usage information of supd.

```
docker exec privatedns_cache_1 supd --help
```

The following table describes action commands.

Commands	Description
`restart_maindb`	This will cause an outage for API and XFR (zone transfer) services.
`restart_maindb_replication`	This will cause an outage for cache container services.

7.2. Admin Actions with the Web Interface

Actions are found on their own tab of each container configuration page. Certain actions are common and shared among all containers. Unique actions specific to a container role are detailed in subsequent tables.

7.2.1. Common Actions for All Containers

Commands	Description
Generate Runtime Report	Generates an archive of runtime reports in /ns1/data/log/health/
Set Basic Auth Credentials	Sets and updates your basic authorization credentials to the container's configuration UI
Restart Service Proxy	This will cause an outage for all service connections. Refresh this configuration page when the action is successful.

7.2.2. Other Actions for Data Containers

Actions	Description
Backup Database	Dumps database to data/backup/<timestamp>.tgz
Restart Main Database	This will cause an outage for API and XFR (zone transfer) services.
Restart Metrics Database	This will cause an outage for metrics.
Restart Messaging Service	This will cause an outage for data container services.
Restart Metrics Service	This will cause an outage for operational metrics.
Restart in Memory Database	This will cause an outage for creation of portal sessions

7.2.3. Other Actions for DNS Containers

Actions	Description
Restart Authoritative DNS Servers	This will cause an outage for DNS.
Restart Caching Service	This will cause DNS caching to be delayed.
Restart Metrics Service	This will cause an outage for metrics reporting.

Restart Recursive DNS Service	This will cause an outage for DNS

7.2.4. Other Actions for Web Containers

Actions	Description
Restart API Service	Restarting the API Service will cause an outage for the API.

7.2.5. Other Actions for XFR Containers

Actions	Description
Restart XFR Service	This action restarts zone transfer services. Zone transfers will be unavailable during the restart.

7.2.6. Other Actions for Cache (Data Cache) Containers

Actions	Description
Restart Main Database	This will cause an outage for API and XFR (zone transfer) services.
Restart Main Database Replication	This will cause an outage for cache container services.

8. Import New GeoIP Data

The dns container images ships with a built-in GeoIP database to facilitate geographic routing. This authoritative nameserver is populated from the free version of Maxmind GeoIP2's database files which NS1 has licensed for redistribution.

If available, a paid version of Maxmind's GeoIP database files can be uploaded to each dns container for more precise geo routing.

Important note: The dns container currently supports binary databases formatted according to MaxMind DB File Format Specification v2.0.

1. Open a web browser to the dns container's **Certs and Files** page.
2. Replace the default GeoIP-City.mmdb file by selecting **Upload**, browse to the file, and select **Open**.
3. Replace the default GeoIP-ASN.mmdb file by selecting **Upload**, browse to the file, and select **Open**.
4. Navigate to the **Configuration Manager** page, select the **Actions** tab, and select **Restart Authoritative DNS Service**. When successfully restarted, the new GeoIP data is loaded.

9. Upgrading to New Versions

When new software versions are available, CHANGELOG.md information can be found here: https://github.com/ns1/ns1-privatedns/blob/master/CHANGELOG.md.

Private DNS follows semantic versioning where the first digit indicates a major/breaking change, the second digit a minor change and/or fixes, and the third digit a patch version with relevant fixes. Unless otherwise noted with a major version release, upgrades can be performed "in place" and done in a "rolling" manner to ensure DNS services remain available and uninterrupted during the upgrade.

Before an upgrade:

1. In the `data` container's Configuration Management page, execute the **Create Backup** action.
2. Download the latest images to host machines using the instructions for Accessing Files & Images.

Using a Docker Compose resource, all containers on the host can be upgraded at once.

3. Stop and remove the containers.

   ```
   sudo docker-compose -p mytest stop && sudo docker-compose -p mytest rm -f
   ```

4. (Optionally) prune the network.

   ```
   sudo docker network prune
   ```

5. Run Docker Compose up command for the new version.

   ```
   sudo TAG=1.1.0 POP_ID=local SERVER_ID=localvm docker-compose -p mytest -f
   docker-compose.yml up -d
   ```

To upgrade a `dns` container in place without using Docker Compose:

3. Stop and remove the container.

   ```
   docker stop dns && docker rm dns
   ```

4. Run the new container version.

   ```
   docker run --name dns ns1inc/privatedns_data:1.1.0
   ```

Important notes:

- Always make a backup file following the first step above. It is also possible to take volume snapshots; however, volume snapshots are not required.
- To keep persistent data, do not prune or remove Docker volumes during the upgrade process.

10. Uninstalling

System uninstallation is simple. To remove the entire system, including volume data, execute the following:

```
docker-compose -p privatedns down -v
```

Important note: This command *is* destructive and will remove volumes. Ensure data is backed up appropriately.

11. Appendix A: Managing Operators

After bootstrap, other operator users can be added to the system. An operator user has full access to every operation and all system data.

Important note: It is highly recommended to add additional operator users after bootstrapping the first operator. If the bootstrapped operator's API Key is lost and no other operators are created yet, the system is rendered unmanageable and a new instance must be deployed.

11.1. View Operator(s)

```
curl -X GET \
  https://{{web-host}}/v1/ops/operators/{{id}} \
  -H 'X-NSONE-Key: {{operator_api_key}}'
```

11.2. Create an Operator

```
curl -X PUT \
  https://{{web-host}}/v1/ops/operators \
  -H 'X-NSONE-Key: {{operator_api_key}}\
  -d '{
      "user": "operator",
      "name": "Operator Name",
      "email": "ops@email.com",
      "password": "mypassword"
}'
```

11.3. Update an Operator

```
curl -X POST \
  https://{{web-host}}/v1/ops/operators/{{id}} \
  -H 'Content-Type: application/json' \
  -H 'X-NSONE-Key: {{operator_api_key}}' \
  -d '{
    "name": "Changed Name",
    "email": "changed@email.com"
}'
```

11.4. Delete an Operator

```
curl -X DELETE \
  https://{{web-host}}/v1/ops/operators/{{id}} \
  -H 'X-NSONE-Key: {{operator_api_key}}'
```

Important note: Operators cannot delete their own accounts from the system. This ensures bootstrapping system users is a one-time event.

12. Appendix B: Managing Organizations

With the exception of operators, all users, zones, and records belong to an organization. To continue with the initial setup of Private DNS, an operator must create the first organization using the 'PUT' API call (Section 12.2) below.

12.1. View Organizations

```
curl -X GET \
  'https://{{web-host}}/v1/ops/orgs' \
  -H 'X-NSONE-Key: {{operator_api_key}}'
```

12.2. Creating an Organization

```
curl -X PUT \
'https://{{web-host}}/v1/ops/orgs' \
-H 'X-NSONE-Key: {{operator_api_key}}' \
-d '{"name": "Organization Name"}'
```

In response, an `org_id` is created along with a unique object `id`. The first organization in the Private DNS instance is set to 2000.

```
{
    "org_id": 2000,
    "name": "Organization Name",
    "id": "5b16cb703ca56f"
}
```

12.3. Update an Organization

```
curl -X POST \
  'https://{{web-host}}/v1/ops/orgs/{{org_id}}' \
  -H 'X-NSONE-Key: {{operator_api_key}}' \
  -d '{"name": "New Organization Name"}'
```

12.4. Delete an Organization

Deleting an organization removes *all* data associated with the organization including: API Keys, Users, Teams, Zone and Record data. For this reason, the DELETE operation can only be performed using an operator user's 2-factor authentication (2FA) token.

Important note: An operator's 2FA secret was returned in response to operator creation. A 2FA token can then be generated with any 6-digit TOTP generator that rotates at 30-second intervals.

```
curl -X DELETE \
  'https://{{web-host}}/v1/ops/orgs/{{org_id}}?token={123456}' \
  -H 'X-NSONE-Key: {{operator_api_key}}'
```

13. Appendix C: Managing API Keys, Users & Teams

API Keys, Users, and Teams are objects within an organization. Each can be managed by an organization member with sufficient privileges; however, operators also have the ability to act on behalf of an organization. This is necessary when setting up the first organization's users or API keys.

If an operator is acting on behalf of an organization, appending exclamation point (`!`) plus the org_id (e.g. `!2000`) to an API key allows use of other endpoints described in NS1's Managed DNS API Documentation.

13.1. Example. Reset a User's Password

Operators can generate a new invite token for users if passwords are forgotten.

```
curl -X POST \
'https://{{web-host}}/v1/ops/account/{{org_id}}/user/{{user-name}}/password/reset' \
  -H 'X-NSONE-Key: {{operator_api_key}}'
```

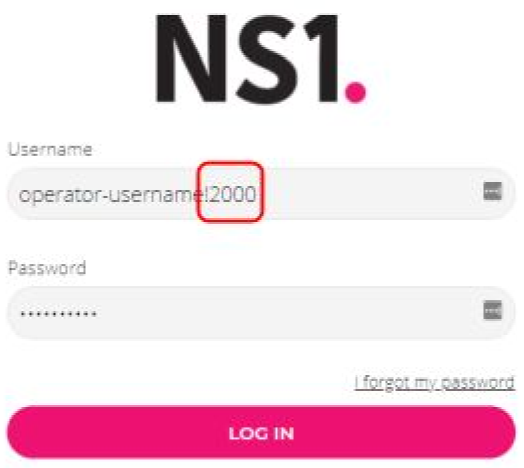
13.2. Example. Adding a Zone as an Operator User

And a zone can be added for an organization's account using the following cURL command:

```
curl -X PUT \
  'https://{{web-host}}/v1/zones/newzone.com' \
  -H 'X-NSONE-Key: {{operator_api_key}}!{{org_id}}' \
  -d  '{"zone":"newzone.com", "nx_ttl":60}'
```

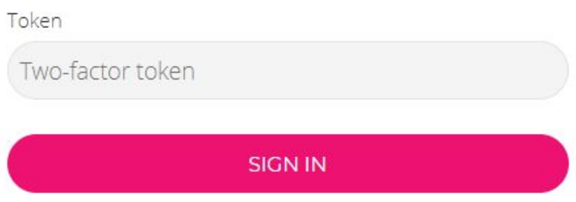
13.3. Example. Using the Portal as an Operator User

In the same way an operator can use the API on behalf of an organization, the operator can also sign into the Private DNS portal by appending exclamation point (**!**) and **org_id** to the operator username of the login page.



Important notes:

- An operator singing into the portal on behalf of an organization is required to provide a 2-factor authentication (2FA) token.
- An operator's 2FA secret was returned in response to operator creation. A 2FA token can then be generated with any 6-digit TOTP generator that rotates at 30-second intervals.



14. Appendix D: Unavailable API Endpoints in Private DNS

The NS1 Managed DNS API Documentation describe endpoints which are not applicable to Private DNS including those pertaining to Account billing and overages. These endpoints will return a 404: Not Found response. The following is a list of non-applicable endpoints:

- Monitoring & Notifications
  - Monitoring Job endpoints
  - Notification List endpoints
- Account Management
  - Overage Alert endpoints
  - Plan and Billing endpoints
  - Payment method endpoints
- Pulsar
  - All Pulsar endpoints