PowerDNS Cloud Control

Reference

Apr 24, 2024

Release 2.4.4

Contents

1 Overview

Configuration of a deployment can be achieved by overriding specific values that are exposed by the Helm chart. This reference guide lists all the values that can be set/configured. Since there are many different configuration options available, these have been separated into the following sections inside the helm values:

Generic configuration:

- **global**: Overrides to pull images from a private registry
- **registrySecrets**: Configuration of credentials for the image registry
- **ipFamily**: Configuration of the cluster networking
- **prometheus**: Configuration of Prometheus scrape settings

Configuration of default settings:

- **auth**: Default configuration for all auth instances
- **dnsdist**: Default configuration for all dnsdist instances
- **recursor**: Default configuration for all recursor instances
- **resolver**: Default configuration for all resolver instances
- **zonecontrol**: Default configuration for all zonecontrol instances

Configuration of instances:

- **auths**: Collection of auth instances to deploy
- **dnsdists**: Collection of dnsdist instances to deploy
- **recursors**: Collection of recursor instances to deploy
- **resolvers**: Collection of resolver instances to deploy
- **rulesets**: Sets of rules to deploy, for dnsdist to use
- **zonecontrols**: Collection of zonecontrol instances to deploy

Each of these sections is explained in detail in the chapters below.

For a basic example of how to use the values, please refer to the *Getting Started* chapter in the *Overview* guide.

2 Private Registries

If you intend to run Cloud Control on a kubernetes cluster which makes use of a local registry and/or a registry cache/proxy, you can use one or more of the following settings in your helm values to configure that registry (for both powerdns and powerdns-operators helm charts):

```
# Monitoring - global overrides
global:
  # Override image-related settings for this chart and all subcharts
  image:
    # Override registry for all images
    registry: "myregistry.local:8085"

    # Override repository for all images
    repository: "myrepository"

    # Override pullPolicy for all images
    pullPolicy: "IfNotPresent"

  # Add imagePullSecrets for this chart and all subcharts
  imagePullSecrets:
    myIPSSecret:
      registry: registry.example.com:5000
      username: regUsername
      password: regPassword
      email: admin@registry.example.com

  # Reference existing Image Pull Secrets for this chart and all subcharts
  imagePullSecretsList:
    - "my-existing-IPS-secret"
```

Each setting explained:

global.image.registry

All images will be attempted to be pulled from this registry (format: host:port)

global.image.repository

All images will be attempted to be pulled from this repository, on above configured registry

global.image.pullPolicy

This pull policy will be specified for each image

global.imagePullSecrets

For each entry a Secret will be created and assigned to each Pod

global.imagePullSecretsList

Each pre-existing Secret referenced in this list (by name) will be assigned to each Pod

If you have a need to override the above settings for specific images, you can find the corresponding 'image:' configuration blocks in the values file.

2.1 Example - Using a local registry with inline credentials

You can configure the local registry using 'global.image.registry' and provide the credentials inline by adding an entry to 'global.imagePullSecrets':

```
# Monitoring - global overrides
global:
  # Override image-related settings
  image:
    # Override registry for all images
    registry: "my.registry.local:5000"

  # Add imagePullSecrets
  imagePullSecrets:
    myRegistrySecret:
      registry: my.registry.local:5000
      username: regUsername
      password: regPassword
      email: admin@my.registry.local
```

2.2 Example - Using a local registry with a pre-provisioned image pull secret

You can configure the local registry using 'global.image.registry' and provide the name of the pre-provisioned image pull secret by adding an entry to 'global.imagePullSecretsList':

```
# Monitoring - global overrides
global:
  # Override image-related settings
  image:
    # Override registry for all images
    registry: "my.registry.local:5000"

  # Reference existing Image Pull Secrets
  imagePullSecretsList:
    - "my-pre-provisioned-IPS-secret"
```

3 Cluster networking

To be able to support Kubernetes clusters with IPv4, IPv6 or dual stack (IPv4 & IPv6) configurations, it is required to ensure the 'ipFamily' configuration in the helm values matches your cluster. The 'ipFamily' section contains the following parameters:

- **ipFamily.ipv4**: Whether or not your cluster has IPv4 enabled (Default: true)

- **ipFamily.ipv6**: Whether or not your cluster has IPv6 enabled (Default: false)

- **ipFamily.families**: Preference of IP families on your cluster, if it is a dualstack cluster

To ensure your deployment is correctly configured, you need to provide one of the 4 possible variations:

3.1 IPv4 only (default)

```
# Networking configuration
ipFamily:
  ipv4: true
  ipv6: false
  families:
    - "IPv4"
    - "IPv6"
```

Note: 'families' is ignored in this configuration. It is only used in a dualstack setup.

3.2 IPv6 only

```
# Networking configuration
ipFamily:
  ipv4: false
  ipv6: true
  families:
    - "IPv4"
    - "IPv6"
```

Note: 'families' is ignored in this configuration. It is only used in a dualstack setup.

3.3 Dualstack - IPv4 primary

If you are running a dualstack cluster, you can check any Pod to see if your cluster has a preference for IPv4 or IPv6. Your pods will have a 'podIP' and 2 values for 'podIPs'. If the 'podIP' is an IPv4 address as shown in the example below, then you are running a cluster with IPv4 as primary:

```
podIP: 172.17.183.4 # IPv4
podIPs:
  - ip: 172.17.183.4 # IPv4
  - ip: fd43:128b:8658:b73b:3eb7:2e30:8815:3f6 # IPv6
```

Configuration for dualstack with IPv4 primary:

```
# Networking configuration
ipFamily:
  ipv4: true
  ipv6: true
  families:
    - "IPv4" # IPv4 is primary
    - "IPv6"
```

3.4 Dualstack - IPv6 primary

If you are running a dualstack cluster, you can check any Pod to see if your cluster has a preference for IPv4 or IPv6. Your pods will have a 'podIP' and 2 values for 'podIPs'. If the 'podIP' is an IPv6 address as shown in the example below, then you are running a cluster with IPv6 as primary:

```
podIP: fd43:128b:8658:b73b:3eb7:2e30:8815:3f6 # IPv6
podIPs:
  - ip: fd43:128b:8658:b73b:3eb7:2e30:8815:3f6 # IPv6
  - ip: 172.17.183.4 # IPv4
```

Configuration for dualstack with IPv6 primary:

```
# Networking configuration
ipFamily:
  ipv4: true
  ipv6: true
  families:
    - "IPv6" # IPv6 is primary
    - "IPv4"
```

4 Resources

Cloud Control's powerdns chart offers several ways to configure Cloud Control in terms of resource requests/limits. These are as follows:

- Containers have no requests and/or limits specified (default)

- All containers have a default cpu & memory limit enabled

By default Cloud Control runs without any limits. You can enable all the containers to have limits by setting *resourceDefaults* to *true* (this is a top-level parameter). Example showing a set of dnsdist instances with the default resource demands enabled:

```
resourceDefaults: true

dnsdists:
  mydnsdist:
    replicas: 2
```

Note: You can also leave *resourceDefaults* on *false* and only specify resource demands for specific containers. The options for configuring specific containers are discussed below.

4.1 Kubernetes: Resource requests & limits

When considering resource allocation for Pods & Containers on Kubernetes, it is important to know how these are handled via *requests* and *limits*. When scheduling a Pod to a node, Kubernetes considers the following regarding capacity:

- **requests**: Minimum amount of a resource which should be available on a node to allow a Pod to be scheduled

- **limits**: Maximum amount of a resource which the Pod can utilize before the node takes action

The *requests* and/or *limits* amount of a resource on a Pod is the sum of the corresponding resource demand on all containers in that Pod (excluding init containers).

At runtime, the limits of each Container are important to understand. When a Container exceeds the *limits* amount, a node generally takes the following action:

- **cpu**: Throttles CPU usage by the Container, usually has a negative performance impact

- **memory**: Terminates the Container with exit code 137 (OOM)

Note: Your cluster might take different actions, check with a cluster administrator if you wish to know how your cluster will react.

If a Pod is scheduled to a node which meets the *requests* amount of resources, but later is unable to allow the Pod to grow to its *limits* amount of resources, a cluster typically attempts to rectify this situation by rescheduling Pods to other nodes. When a Pod is rescheduled to another node a new instance of it is spawned on the other node and this has certain implications:

- DNSdist, Auth and Recursor: It will start up again with a fresh cache
- Auth with Lightning Stream: It will need to start a fresh sync from the S3 bucket

Both are preferable to avoid, hence the preference to avoid having any *requests* lower than the *limits*.

4.2 Cloud Control defaults

The default resources (enabled via *resourceDefaults: true*) assigned to Cloud Control Pods / Containers are set based on the following:

- Defaults only have *limits*, not *requests*. Having no *requests* implies that Kubernetes only schedules the Pods to nodes which can meet the demanded *limits* and hence reduces the risk of having a Pod rescheduled to another node. Preventing unnecessary Pod rescheduling helps ensure DNSdist, Recursor & Auth do not have to rebuild their caches and Lightning Stream will not have to re-synchronise.
- The limits for DNSdist, Auth, Recursor and Lightning Stream containers are conservative, the should be seen as 'minimum requirements' rather than 'recommended requirements'. Subsequent chapters will help judge which of these settings you might want to increase depending on your type of deployment.

4.3 Resource tuning: DNSdist

A set of DNSdist instances has options for tuning the resources allocated to each container, via the following options:

- **agentResources**: Resources allocated to the 'agent' container
- **aggregatorResources**: Resources allocated to the 'aggregator' container
- **initResources**: Resources allocated to the 'dnsdist-init' container
- **resources**: Resources allocated to the 'dnsdist' container
- **rpcServerResources**: Resources allocated to the 'rpc-server' container
- **stateResources**: Resources allocated to the 'dnsdist-state' container

These options take a Kubernetes *resource* object as value.

Example modifying the limits for the *dnsdist* container:

```
dnsdists:
  mydnsdist:
    replicas: 2

    # Resources allocated to the 'dnsdist' container
    resources:
```

```
    limits:
      cpu: 6
      memory: 12Gi
```

Above will cause the dnsdist container to demand 6 cores (or vCPUs) and 12 GB of memory.

When tuning the resources allocated to a DNSdist Pod, it is important to consider the following:

- CPU requirements of the *dnsdist* container are largely dependent on the amount of requests which are handled and (if applicable) the amount of Lua script involved in processing each request

- Memory requirements of the *dnsdist* container are mainly based on the size of the cache

4.4 Resource tuning: Recursor

A set of Recursor instances has options for tuning the resources allocated to each container, via the following options:

- **agentResources**: Resources allocated to the 'recursor-agent' container

- **aggregatorResources**: Resources allocated to the 'aggregator' container

- **initResources**: Resources allocated to the 'recursor-init' container

- **resources**: Resources allocated to the 'recursor' container

These options take a Kubernetes *resource* object as value.

Example modifying the limits for the *recursor* container:

```
recursors:
  myrecursor:
    replicas: 2

    # Resources allocated to the 'recursor' container
    resources:
      limits:
        cpu: 6
        memory: 6Gi
```

Above will cause the recursor container to demand 6 cores (or vCPUs) and 6 GB of memory.

When tuning the resources allocated to a Recursor Pod, it is important to consider the following:

- CPU requirements of the *recursor* container are largely dependent on the amount of requests which are handled and (if applicable) the amount of Lua script involved in processing each request

- Memory requirements of the *recursor* container are mainly based on the size of the cache

4.5 Resource tuning: Auth

A set of Auth instances has options for tuning the resources allocated to each container, via the following options:

- **agentResources**: Resources allocated to the 'agent' container

- **aggregatorResources**: Resources allocated to the 'aggregator' container

- **initResources**: Resources allocated to the 'auth-init' container

- **lightningStreamResources**: Resources allocated to the 'ls' container

- **resources**: Resources allocated to the 'auth' container

These options take a Kubernetes *resource* object as value.

Example modifying the limits for the *auth* container:

```
auths:
  myauth:
    replicas: 2

    backends:
      {}

    # Resources allocated to the 'auth' container
    resources:
      limits:
        cpu: 6
        memory: 6Gi
```

Above will cause the auth container to demand 6 cores (or vCPUs) and 6 GB of memory.

If Lightning Stream is used as backend, it is advisable to adjust its resource limits as well, for example:

```
auths:
  myauth:
    replicas: 2

    backends:
      - type: ls
        mapSize: 5000
        access_key: my_access_key
        secret_key: my_secret_key
        region: my_region
        bucket: my_bucket
        endpoint: https://some.s3.endpoint.com/

    # Resources allocated to the 'auth' container
    resources:
      limits:
        cpu: 6
        memory: 7Gi

    # Resources allocated to the 'ls' container
    lightningStreamResources:
      limits:
```

(continues on next page)

```
        cpu: 2
        memory: 6Gi
```

In above example it is important to notice 3 figures:

- auth: memory limit

- ls: memory limit

- backend: mapsize (in MB)

All 3 of these closely relate to the size of the dataset of the Auth instance.

When tuning the resources allocated to an Auth Pod, it is important to consider the following:

- CPU requirements of the *auth* container are largely dependent on the amount of requests which are handled

- Memory requirements of the *auth* container are mainly based on the size of the cache and (if using Lightning Stream) the size of the dataset

- CPU requirements of the *ls* container are largely dependent on the amount and frequency of changes in the dataset

- Memory requirements of the *ls* container are mainly based on the size of the dataset

A good starting point for memory in an Auth Pod with Lightning Stream is to determine an estimate for the mapsize based on the expected size of the dataset. For the memory limit of the LS container you can take the *mapsize* + 10-20%. For the Auth container you start with a limit of *mapsize* + 20-40%.

4.5.1 Lightning Stream with large data sets

When using Lightning Stream with large data sets, there are additional options which are worth considering regarding memory usage. These options are related to the process of downloading snapshots and then processing them. This process works as follows:

1. Download snapshot

2. Decompress snapshot

3. Process snapshot

While there can only be 1 snapshot processing for a specific database, there are queues which ensure processing does not need to wait for a new snapshot to be downloaded and decompressed. These queues are as follows:

- Downloaded snapshots, ready for decompression

- Decompressed snapshots, ready for processing

Default capacity per queue:

- Downloaded snapshots: option *memoryDownloadedSnapshots*, default is *3*

- Decompressed snapshots: option *memoryDecompressedSnapshots*, default is *2*

If the data set is large enough for these queues to consume significant memory, you can consider tuning them down, but preferably not below the following:

- *memoryDownloadedSnapshots = 2*
- *memoryDecompressedSnapshots = 1*

5 Exposing dnsdist

By default a set of dnsdist instances will have a Service object created with type ClusterIP. The default configuration is as follows (Values: dnsdist.service):

```
service:
  type: ClusterIP
  ports:
    udp:
      port: 53
      protocol: UDP
    tcp:
      port: 53
      protocol: TCP
```

The above results in a Service object with UDP & TCP listeners on port 53. To modify the default service, you can specify a *service* item in the dnsdist instance, such as:

```
dnsdists:
  mydnsdist:
    replicas: 2
    pools:
      default:
        serverGroups:
          - group: myrecursor
        packetcache:
          maxEntries: 200000
    service:
      type: NodePort
```

This will override the type of the Service object from *ClusterIP* to *NodePort*.

Available parameters:

- type

- loadBalancerIP (Only applicable if type = LoadBalancer)

- loadBalancerSourceRanges (Only applicable if type = LoadBalancer)

- loadBalancerClass (Only applicable if type = LoadBalancer)

- lbMixedProtocol (Only applicable if type = LoadBalancer)

- lbPerIPFamily (Only applicable if type = LoadBalancer)

- clusterIP

- clusterIPs

- externalIPs

- externalName

- externalTrafficPolicy

- internalTrafficPolicy

- sessionAffinity

- healthCheckNodePort

- sessionAffinityConfig

- labels

- annotations

- ipv4Annotations (Only applicable if lbPerIPFamily = true)

- ipv6Annotations (Only applicable if lbPerIPFamily = true)

These parameters directly expose Kubernetes' configuration equivalents, for their usage & accepted values please refer to: https://kubernetes.io/docs/reference/kubernetes-api/service-resources/service-v1/

5.1 Example: Exposing via NodePort

To expose dnsdist via a NodePort, you can specify the appropriate type. In this example, both UDP & TCP will be exposed via a NodePort, where the NodePort will be randomly selected from the node-port range:

```
dnsdists:
  mydnsdist:
    replicas: 2
    pools:
      default:
        serverGroups:
          - group: myrecursor
        packetcache:
          maxEntries: 200000
    service:
      type: NodePort
```

You can also specify the NodePort which you want to use, for example:

```
dnsdists:
  mydnsdist:
    replicas: 2
    pools:
      default:
        serverGroups:
          - group: myrecursor
        packetcache:
          maxEntries: 200000
    service:
      type: NodePort
      ports:
        udp:
          port: 53
```

(continues on next page)

```
        protocol: UDP
        nodePort: 30053
      tcp:
        port: 53
        protocol: TCP
        nodePort: 30053
```

5.2 Example: Exposing via LoadBalancer

To expose dnsdist via a LoadBalancer, you can specify the appropriate type. When 'LoadBalancer' is configured as type, it will create the default ClusterIP Service + 1 LoadBalancer Service for each specified port. This is done to satisfy a Kubernetes constraint regarding LoadBalancer objects when multiple protocols are requested.

In this example, both the default UDP & TCP ports will be exposed via a LoadBalancer Service (MetalLB annotation provided as example, substitute with annotations required by your LoadBalancer provider) on port 53:

```
dnsdists:
  mydnsdist:
    replicas: 2
    pools:
      default:
        serverGroups:
          - group: myrecursor
        packetcache:
          maxEntries: 200000
    service:
      type: LoadBalancer
      annotations:
        metallb.universe.tf/address-pool: name_of_pool
```

If necessary, you can also request a specific LoadBalancer IP, as shown in the following example:

```
dnsdists:
  mydnsdist:
    replicas: 2
    pools:
      default:
        serverGroups:
          - group: myrecursor
        packetcache:
          maxEntries: 200000
    service:
      type: LoadBalancer
      annotations:
        metallb.universe.tf/address-pool: name_of_pool
      loadBalancerIP: 12.34.56.78
```

5.3 Example: Exposing via LoadBalancer with mixed protocols

By default, a separate LoadBalancer service will be created per protocol (tcp & udp for dnsdist). If your cluster has the 'MixedProtocolLBService' feature gate enabled you can force a single LB to be created with both protocols using the 'lbMixedProtocol' flag:

```
dnsdists:
  mydnsdist:
    replicas: 2
    pools:
      default:
        serverGroups:
          - group: myrecursor
        packetcache:
          maxEntries: 200000
    service:
      type: LoadBalancer
      lbMixedProtocol: true
```

5.4 Example: Separate LoadBalancers for IPv4 & IPv6 in a dualstack cluster

If you run a dualstack cluster, you can opt to have a separate LoadBalancer created for each IP family (IPv4 & IPv6). This can be done using the 'lbPerIPFamily' flag:

```
dnsdists:
  mydnsdist:
    replicas: 2
    pools:
      default:
        serverGroups:
          - group: myrecursor
        packetcache:
          maxEntries: 200000
    service:
      type: LoadBalancer
      lbPerIPFamily: true
```

This will create 4 LoadBalancer services, namely:

- TCP-IPv4

- UDP-IPv4

- TCP-IPv6

- UDP-IPv6

If your cluster has the 'MixedProtocolLBService' feature gate enabled, you can use the 'lbMixed-Protocol' flag to have this reduced to an IPv4 LoadBalancer and an IPv6 LoadBalancer, each with both TCP & UDP supported:

```
dnsdists:
  mydnsdist:
```

```yaml
    replicas: 2
    pools:
      default:
        serverGroups:
          - group: myrecursor
        packetcache:
          maxEntries: 200000
    service:
      type: LoadBalancer
      lbMixedProtocol: true
      lbPerIPFamily: true
```

If your LoadBalancer provider requires specific annotations on the IPv4 and IPv6 LoadBalancers, you can use the 'ipv4Annotations' and 'ipv6Annotations' parameters:

```yaml
dnsdists:
  mydnsdist:
    replicas: 2
    pools:
      default:
        serverGroups:
          - group: myrecursor
        packetcache:
          maxEntries: 200000
    service:
      type: LoadBalancer
      lbMixedProtocol: true
      lbPerIPFamily: true
      ipv4Annotations:
        ipv4.example/anno1: "ipv4_1"
        ipv4.example/anno2: "ipv4_2"
      ipv6Annotations:
        ipv6.example/anno1: "ipv6_1"
        ipv6.example/anno2: "ipv6_2"
```

6 Exposing auth API

By default a set of auth instances will not have an Ingress object created. If you wish to expose auth's API you can do so by configuring the Ingress object.

Note: We highly recommend providing the necessary stickiness annotations to any created Ingress object.

For these examples we will assume the following auth instance is to be exposed:

```
auths:
  myauth:
    replicas: 2
```

If the above is deployed, 2 replicas of auth will be created, but no Ingress object will exist. To add an Ingress, use the following additional config:

```
auths:
  myauth:
    replicas: 2
    api:
      ingress:
        ingressClassName: Name of the Ingress Class - omit this if you wish to use the␣
↪default Ingress Class or prefer to use the deprecated method of using annotations
        enabled: true
        annotations:
          <Annotations required by your Ingress controller>
        hosts:
          - <Hosts on which to have the Ingress listen>
        tls:
          - secretName: <Secret containing certificates - if HTTPS is desired>
            hosts:
              - <Hosts for which HTTPS is desired>
```

6.1 NGINX Example: HTTP

Example for plain HTTP using *nginx* Ingress Controller for host *myhost.mydomain.com* using the ingressClassName *nginx*:

```
auths:
  myauth:
    replicas: 2
    api:
      ingress:
        enabled: true
```

```
        ingressClassName: nginx
        hosts:
          - myhost.mydomain.com
```

6.2 NGINX Example: HTTPS with cert-manager

Example for HTTPS using *nginx* Ingress Controller & certificates generated into a secret named *auth-api-certs* by a cert-manager cluster issuer named *letsencrypt-prod* on host *myhost.mydomain.com*:

```
auths:
  myauth:
    replicas: 2
    api:
      ingress:
        enabled: true
        ingressClassName: nginx
        annotations:
          cert-manager.io/cluster-issuer: letsencrypt-prod
        hosts:
          - myhost.mydomain.com
        tls:
          - secretName: auth-api-certs
            hosts:
              - myhost.mydomain.com
```

6.3 NGINX Example: HTTP with stickiness using cookies

Example for HTTP using *nginx* Ingress Controller & stickiness using a cookie named *ccroute*. This can be a reliable way of ensuring stickiness if the clients connecting to the Ingress are capable of handling cookies.

```
auths:
  myauth:
    replicas: 2
    api:
      ingress:
        enabled: true
        ingressClassName: nginx
        annotations:
          nginx.ingress.kubernetes.io/affinity: "cookie"
          nginx.ingress.kubernetes.io/session-cookie-name: "ccroute"
          nginx.ingress.kubernetes.io/session-cookie-expires: "172800"
          nginx.ingress.kubernetes.io/session-cookie-max-age: "172800"
        hosts:
          - myhost.mydomain.com
```

6.4 NGINX Example: HTTP with stickiness using upstream-hash-by

Example for HTTP using *nginx* Ingress Controller & stickiness using consistent hashing. This can be a reliable way of ensuring stickiness if the clients connecting to the Ingress can not be relied upon to handle cookies.

```yaml
auths:
  myauth:
    replicas: 2
    api:
      ingress:
        enabled: true
        ingressClassName: nginx
        annotations:
          nginx.ingress.kubernetes.io/upstream-hash-by: "$binary_remote_addr"
        hosts:
          - myhost.mydomain.com
```

7 Labels & Annotations

To add labels & annotations to objects you can use several layers of configuration. To add labels & annotations to all pods or services you can use the following top-level configuration items:

- **podAnnotations**: Annotations to be added to each Pod (default: *{}*)

- **podLabels**: Labels to be added to each Pod (default: *{}*)

- **serviceLabels**: Labels to be added to each Service object (default: *{}*)

Alternatively, you can also add labels & annotations to all instances of a specific type, for example:

```yaml
# Labels on dnsdist pods & services
dnsdist:
  podLabels:
    mylabel: label_value
  serviceLabels:
    mylabel: label_value

# Annotations on Auth pods
auth:
  podAnnotations:
    my.example.com/annotation: annotation_value
    my.example.com/moreannotations: another_value

# Labels on Recursor services
recursor:
  serviceLabels:
    mylabel: label_value
```

Or you can add labels & annotations specifically to a set of instances, for example:

```yaml
# dnsdist instances with pod & service labels
dnsdists:
  mydnsdist:
    podLabels:
      mylabel: my_value
    replicas: 2
    pools:
      default:
        serverGroups:
          - group: myrecursor
    service:
      labels:
        my_service_label: some_value
        another_label: another_value
```

```
# Recursor instances with annotations
recursors:
  myrecursor:
    podAnnotations:
      my.example.com/anno1: annotation_value
      my.example.com/anno2: another_value
    replicas: 2
```

7.1 Labeling services with multiple ports

There are 2 types of service configurations which can lead to the creation of multiple service objects for the same set of instances, namely:

- Services of type: *LoadBalancer*

- Services with: *servicePerPort: true*

To give both service objects the same labels you can use the above documented *labels:* configuration item on the parent *service:* item.

However, if you wish to configure different labels for each of these services, you can go one step further and define the labels for each port, for example:

```
# dnsdist instances with pod & service labels
dnsdists:
  mydnsdist:
    replicas: 2
    pools:
      default:
        serverGroups:
          - group: recursor
    service:
      labels:
        service.label/generic: some_value
      type: LoadBalancer
      externalTrafficPolicy: Local
      ports:
        udp:
          labels:
            service.label/udp: udp_label_value
        tcp:
          labels:
            service.label/tcp: tcp_label_value
```

This will lead to the following labels on each service:

- dnsdist-mydnsdist-lb-tcp: Has labels *service.label/generic* & *service.label/tcp*

- dnsdist-mydnsdist-lb-udp: Has labels *service.label/generic* & *service.label/udp*

7.2 Precedence of labeling & annotating

Since you can add labels & annotations at multiple levels, it is important to understand the order of precedence. In CloudControl, the order is as follows:

(for Services: Port-specific labels) > Instance-specific labels/annotations > Instance-type labels/annotations > top-level labels/annotations

The precedence is only used to pick a label/annotation value if the exact name of a label/annotation is defined at multiple levels. If labels/annotations are defined at different levels with different names, they are all added to the final set of labels/annotations on an object.

8 Security contexts

By default Cloud Control deploys all Pods with the following security context:

```
securityContext:
  fsGroup: 953
  runAsUser: 953
  runAsGroup: 953
```

And all containers have the following applied:

```
securityContext:
  readOnlyRootFilesystem: true
  allowPrivilegeEscalation: false
```

The Pod *securityContext* and/or Container *securityContext* can be overwritten on a specific set of instances or globally for all Pods/Containers deployed as part of Cloud Control.

8.1 Configuring a global Pod security context

To overwrite the Pod securityContext for all instances you can add a *podSecurityContext* at the top-level:

```
podSecurityContext:
  runAsUser: 1000953

dnsdists:
  mydnsdist:
    replicas: 2
    pools:
      default:
        serverGroups:
          - group: myrecursor

recursors:
  myrecursor:
    replicas: 2
```

Since the configuration of this podSecurityContext overwrites the full default Pod securityContext, this leads to the following differences compared to the defaults:

- fsGroup - This is no longer present, so it is dropped from the Pod security context
- runAsUser - This is now set to '1000953'
- runAsGroup - This is no longer present, so it is dropped from the Pod security context

Notes:

- When overriding the Pod securityContext, make sure you always specify a *runAsUser* or have your platform assign one automatically

- Any field supported by Kubernetes in a Pod-level securityContext can be configured here

8.2 Configuring a Pod security context for a specific set of instances

To overwrite the Pod securityContext for a set of instances you can add a *podSecurityContext* on the instance level:

```
recursors:
  myrecursor:
    replicas: 2
    podSecurityContext:
      runAsUser: 1000953
```

8.3 Precedence of podSecurityContext

Since you can configure podSecurityContext at multiple levels, it is important to understand the order of precedence. In CloudControl, the order is as follows:

Instance-specific podSecurityContext > top-level podSecurityContext > default podSecurityContext

Note: The podSecurityContext configurations are not merged into each other. If you specify a podSecurityContext on an instance it will fully replace a podSecurityContext of lower precedence.

8.4 Configuring a global Container security context

To overwrite the Container securityContext for all instances you can add a *containerSecurityContext* at the top-level:

```
containerSecurityContext:
  capabilities:
    drop:
    - ALL

dnsdists:
  mydnsdist:
    replicas: 2
    pools:
      default:
        serverGroups:
        - group: myrecursor

recursors:
  myrecursor:
    replicas: 2
```

Since the configuration of this containerSecurityContext overwrites the full default Container securityContext, this leads to the following differences compared to the defaults:

- allowPrivilegeEscalation - This is no longer present, so it is dropped from the Container security context

- capabilities - This is now set to drop 'ALL' capabilities

- readOnlyRootFilesystem - This is no longer present, so it is dropped from the Container security context

Notes:

- Any field supported by Kubernetes in a Container-level securityContext can be configured here

8.5 Configuring a Container security context for a specific set of instances

To overwrite the PContainerod securityContext for a set of instances you can add a *containerSecurityContext* on the instance level:

```
recursors:
  myrecursor:
    replicas: 2
    containerSecurityContext:
      capabilities:
        drop:
          - ALL
```

8.6 Precedence of containerSecurityContext

Since you can configure containerSecurityContext at multiple levels, it is important to understand the order of precedence. In CloudControl, the order is as follows:

Instance-specific containerSecurityContext > top-level containerSecurityContext > default containerSecurityContext

Note: The containerSecurityContext configurations are not merged into each other. If you specify a containerSecurityContext on an instance it will fully replace a containerSecurityContext of lower precedence.

9 Storage

By default Cloud Control deploys all Pods with storage tailored for specific usecases, mostly with read-only privileges. If you wish to utilise storage for additional activities, you can configure additional storage to be available on Pods. Example usage of such additional storage could be (but is not limited to):

- Dumping cache to file for inspection

- Exporting zone(s) to a file

Additional storage can be configured at 2 levels:

- Global: Available on all Pods

- Instance Set: All Pods of a dnsdist, recursor or auth deployment

The following chapters show how these can both be configured.

9.1 Defaults

Additional volumes follow these defaults:

- additional volumes are mounted only to containers running PowerDNS products:

  - dnsdist: The *dnsdist* container

  - recursor: The *recursor* container

  - auth: the *auth* container + (if applicable) the lightning stream *ls* container

- additional volumes are mounted to containers with read-write permissions

These defaults can be modified using the configuration options below.

9.2 Global

To configure additional storage for all instances in a Cloud Control installation you can add a *volumes* at the top-level:

```
volumes:
  - name: example
    mountPath: /example
    volumeSource:
      emptyDir: {}
```

Each entry in *volumes* must always have the following parameters:

- name: Name of the volume

- mountPath: Path to which the volume will be mounted (make sure this path is not already in use by an internal Cloud Control volume)

- volumeSource: A valid Kubernetes *VolumeSource*

Optional parameters:

- allContainers - If the volume is mounted to all containers in a Pod (Default: *false*).

- mountPropagation - A valid Kubernetes *mountPropagation* value

- readOnly - If the volume is mounted as read-only (Default: *false*)

- subPath - A valid Kubernetes *subPath* value

- subPathExpr - A valid Kubernetes *subpathExpr* value

For more information regarding the Kubernetes parameters mentioned above, you can read: https://kubernetes.io/docs/concepts/storage/volumes/

9.3 Instance Set

To configure additional storage for Pods within a specific set of instances you can add a *volumes* on the instance level:

```
dnsdists:
  mydnsdist:
    replicas: 2
    volumes:
      - name: distonly
        mountPath: /distonly
        volumeSource:
          emptyDir:
            sizeLimit: 1Gi
            medium: "Memory"
      - name: all
        mountPath: /all
        allContainers: true
        volumeSource:
          emptyDir: {}
```

The above example adds 2 volumes:

- *distonly*, mounted at */distonly* on the dnsdist container and backed by node memory up to a capacity of 1 Gi.

- *all*, mounted at */all* on all containers and backed by whichever medium the cluster has configured as default for *emptyDir*

10 Default configuration

10.1 auth

Default settings for all auth instances, configurable under 'auth' in helm values.

- **affinity**: Kubernetes pod affinity (default: *{}*)

- **agentSecurityContext**: Default Security Context for the agent container (allowPrivilegeEscalation: *false*, readOnlyRootFilesystem: *true*)

- **antiAffinityPreset**: Pod anti affinity, if *affinity* is not set (default: *preferred*)

- **apiKey:** API key used to access the /api endpoint, used to configure a static key (default: generated and stored in a secret)

- **authSecurityContext**: Default Security Context for the auth container (allowPrivilegeEscalation: *false*, readOnlyRootFilesystem: *true*)

- **initSecurityContext**: Default Security Context for the auth-init init container (allowPrivilegeEscalation: *false*, readOnlyRootFilesystem: *true*)

- **lightningStreamSecurityContext**: Default Security Context for the ls (lightning stream) container (allowPrivilegeEscalation: *false*, readOnlyRootFilesystem: *true*)

- **lsMigratorSecurityContext**: Default Security Context for the ls migrator container (allowPrivilegeEscalation: *false*, readOnlyRootFilesystem: *true*)

- **lsMigratorInitSecurityContext**: Default Security Context for the ls migrator init container (allowPrivilegeEscalation: *false*, readOnlyRootFilesystem: *true*)

- **metricsPath**: Path on which Prometheus metrics will be served (default: */metrics*)

- **metricsPort**: Port on which Prometheus metrics will be served (default: *8082*)

- **nodeSelector**: Kubernetes pod nodeSelector (default: *{}*)

- **podAnnotations**: Annotations to be added to each Pod

- **podLabels**: Labels to be added to each Pod

- **serviceLabels**: Labels to be added to each Service object (default: *{}*)

- **readinessFailureThreshold**: When a probe fails, Kubernetes will try this many times before marking the container as Unready. Updates deployment, resulting in respawn of recursor pods (default: *2*)

- **readinessInitialDelaySeconds**: Number of seconds after the containers have started before readiness probes are initiated. Updates deployment, resulting in respawn of recursor pods. (default: *5*)

- **readinessPeriodSeconds**: How often (in seconds) to perform the readiness probes. Updates deployment, resulting in respawn of recursor pods (default: *2*)

- **readinessStateProbeInterval**: How often (in seconds) the agent will judge the health state of the recursor agent. (default: *2*)

- **readinessSuccessThreshold**: Minimum consecutive succesfull probes before a container is marked Ready. Updates deployment, resulting in respawn of recursor pods (default: *2*)

- **readinessTimeoutSeconds**: Number of seconds after which the readiness probes time out. Updates deployment, resulting in respawn of recursor pods (default: *1*)

- **replicas**: Default number of replicas in an auth deployment (default: *2*)

- **tolerations**: Default Pod tolerations (default: *[]*)

- **webserverACLAllowAll**: Allow all inbound traffic to auth webserver, regardless of source IP (default: *true*)

- **webserverACL**: Netmasks to allow webserver traffic from (default: *127.0.0.1/32, 192.168.0.0/16, 10.0.0.0/8, 172.16.0.0/12, ::1, fc00::/7*)

- **webserverPassword**: Instead of generating a password for the webserver, set a static one (default: generated and stored in a secret)

- **hpa**: (Horizontal Pod Autoscaler defaults)

  - **enabled**: Enable or disable the HPA (default: *false*)

  - **minReplicas**: Minimum # of replicas (default: *2*)

  - **maxReplicas**: Maximum # of replicas (default: *4*)

  - **metrics**: Metric(s) upon which to make decisions regarding scaling. Must be an array of MetricSpec (`https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#metricspec-v2beta2-autoscaling`). Examples are in the helm values.

  - **behavior**: Behavior of the HPA, must be instance of HorizontalPodAutoscalerBehavior (`https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#horizontalpodautoscalerspec-v2beta2-autoscaling`).

- **agentResources**: (auth agent resource allocation defaults)

  - **limits**: (Limit amounts)

    * **cpu**: Limit amount of CPU (Kubernetes default)

    * **memory**: Limit amount of memory (Kubernetes default)

  - **requests**: (Request amounts)

    * **cpu**: Request amount of CPU (Kubernetes default)

    * **memory**: Request amount of memory (Kubernetes default)

- **resources**: (auth resource allocation defaults)

  - **limits**: (Limit amounts)

    * **cpu**: Limit amount of CPU (Kubernetes default)

    * **memory**: Limit amount of memory (Kubernetes default)

- **requests**: (Request amounts)

  * **cpu**: Request amount of CPU (Kubernetes default)

  * **memory**: Request amount of memory (Kubernetes default)

- **lightningStreamResources**: (LightningStream resource allocation defaults)

  - **limits**: (Limit amounts)

    * **cpu**: Limit amount of CPU (Kubernetes default)

    * **memory**: Limit amount of memory (Kubernetes default)

  - **requests**: (Request amounts)

    * **cpu**: Request amount of CPU (Kubernetes default)

    * **memory**: Request amount of memory (Kubernetes default)

- **lightningStream**: (LightningStream defaults)

  - **logLevel**: Loglevel for LightningStream containers (default: *info*, available options: *debug*, *info*, *warning*, *error*, *fatal*)

10.2 dnsdist

Default settings for all dnsdist instances, configurable under 'dnsdist' in helm values.

- **aclAdd**: Netmasks allowed to access dnsdist, in addition to the loopback, RFC1918 and other local addresses. Only applicable if aclAllowAll is *false* (default: *[]*)

- **aclAllowAll**: Allow all inbound traffic to dnsdist, regardless of source IP. (default: *true*)

- **affinity**: Kubernetes pod affinity (default: *{}*)

- **agentSecurityContext**: Default Security Context for the agent container (allowPrivilegeEscalation: *false*, readOnlyRootFilesystem: *true*)

- **antiAffinityPreset**: Pod anti affinity, if *affinity* is not set (default: *preferred*)

- **apiKey**: API key used to access the /api endpoint, used to configure a static key (default: generated and stored in a secret)

- **consistentHashingBalancingFactor**: Set the maximum imbalance between the number of outstanding queries intended for a given server, based on its weight, and the actual number, when using the chashed consistent hashing load-balancing policy. Default is 0, which disables the bounded-load algorithm. (default: *0*)

- **dnsdistSecurityContext**: Default Security Context for the dnsdist container (allowPrivilegeEscalation: *false*, readOnlyRootFilesystem: *true*)

- **do53Locals**: Default amount of Do53 listen sockets per dnsdist pod (default: *1*)

- **do53TcpFastOpenQueueSize**: Default size of the TCP Fast Open queue on Do53 listen sockets (Dnsdist default)

- **do53TcpListenQueueSize**: Default size of the listen queue on Do53 listen sockets (Dnsdist default)

- **dohLocals**: Default amount of listen sockets per DoH listener (default: *1*)

- **dotLocals**: Default amount of listen sockets per DoT listener (default: *1*)

- **initSecurityContext**: Default Security Context for the dnsdist-init init container (allow-PrivilegeEscalation: *false*, readOnlyRootFilesystem: *true*)

- **metricsPath**: Path on which Prometheus metrics will be served (default: */metrics*)

- **metricsPort**: Port on which Prometheus metrics will be served (default: *8082*)

- **nodeSelector**: Kubernetes pod nodeSelector (default: *{}*)

- **podAnnotations**: Annotations to be added to each Pod

- **podLabels**: Labels to be added to each Pod

- **podSecurityContext**: Kubernetes Pod security context (default: *953* as fsGroup, runAsUser & runAsGroup)

- **port**: Port on which dnsdist will listen for Do53 traffic (default: *5353*)

- **readinessDo53ProbeInterval**: How often (in seconds) the agent will judge the health state of dnsdist via tests against the Do53 listeners. (default: *2*)

- **readinessDo53QDomain**: Domain used in the query to judge whether the Do53 listeners of a dnsdist container are healthy and ready for traffic. (default: *a.root-servers.net*)

- **readinessDo53QTimeout**: Number of seconds after which the query used to judge whether the Do53 listeners of a dnsdist container are healthy and ready for traffic time out. (default: *1*)

- **readinessDo53QType**: Type of query used to judge whether the Do53 listeners of a dnsdist container are healthy and ready for traffic. (default: *A*)

- **readinessFailureThreshold**: When a probe fails, Kubernetes will try this many times before marking the container as Unready. Updates deployment, resulting in respawn of dnsdist pods (default: *2*)

- **readinessInitialDelaySeconds**: Number of seconds after the containers have started before readiness probes are initiated. Updates deployment, resulting in respawn of dnsdist pods. (default: *5*)

- **readinessPeriodSeconds**: How often (in seconds) to perform the readiness probes. Updates deployment, resulting in respawn of dnsdist pods (default: *2*)

- **readinessStateProbeInterval**: How often (in seconds) the agent will judge the health state of the dnsdist agent. (default: *2*)

- **readinessSuccessThreshold**: Minimum consecutive succesfull probes before a container is marked Ready. Updates deployment, resulting in respawn of dnsdist pods (default: *2*)

- **readinessTimeoutSeconds**: Number of seconds after which the readiness probes time out. Updates deployment, resulting in respawn of dnsdist pods (default: *1*)

- **replicas**: Default number of replicas in a dnsdist deployment (default: *2*)

- **roundRobinFailOnNoServer**: By default the roundrobin load-balancing policy will still try to select a backend even if all backends are currently down. Setting this to true will make the policy fail and return that no server is available instead. (default: *false*)

- **rpcServerSecurityContext**: Default Security Context for the rpc-server container (allow-PrivilegeEscalation: *false*, readOnlyRootFilesystem: *true*)

- **serverPurgeDelay**: Delay (in seconds) after which replaced servers are purged from dns-dist (default: *5*)

- **servFailWhenNoServer**: If true, return a ServFail when no servers are available, instead of the default behaviour of dropping the query. (default: *false*)

- **serviceLabels**: Labels to be added to each Service object (default: *{}*)

- **stateSecurityContext**: Default Security Context for the dnsdist-state init container (allowPrivilegeEscalation: *false*, readOnlyRootFilesystem: *true*)

- **stekLength**: Length of a single STEK ticket, in bytes (default: *80*)

- **stekMaxAge**: Interval in seconds at which a new STEK ticket should be generated and the oldest one removed (default: *43200*)

- **stekPollInterval**: Interval in seconds at which a check should be performed for an updated set of STEK tickets (default: *15*)

- **stekCount**: Number of STEK tickets to keep stored. Recommended to keep this value at least at >= 2 (default: *5*)

- **tolerations**: Default Pod tolerations (default: *[]*)

- **webserverACLAllowAll**: Allow all inbound traffic to dnsdist webserver, regardless of source IP (default: *true*)

- **webserverACL**: Netmasks to allow webserver traffic from (default: *127.0.0.1/32, 192.168.0.0/16, 10.0.0.0/8, 172.16.0.0/12, ::1, fc00::/7*)

- **webserverPassword**: Instead of generating a password for the webserver, set a static one (default: generated and stored in a secret)

- **weightedBalancingFactor**: Set the maximum imbalance between the number of outstanding queries intended for a given server, based on its weight, and the actual number, when using the whashed or wrandom load-balancing policy. Default is 0, which disables the bounded-load algorithm. (default: *0*)

- **ecs**: (dnsdist ECS functions)

    - **setECSOverride**: Whether to override an existing EDNS Client Subnet option present in the query (only effective when the backend server has annotation useClientSubnet = true). (Dnsdist default)

    - **setECSSourcePrefixV4**: Truncate the requestors IPv4 address to this length, in bits (only effective when the backend server has annotation useClientSubnet = true). (Dnsdist default)

    - **setECSSourcePrefixV6**: Truncate the requestors IPv6 address to this length, in bits (only effective when the backend server has annotation useClientSubnet = true). (Dnsdist default)

- **hpa**: (Horizontal Pod Autoscaler defaults)

    - **enabled**: Enable or disable the HPA (default: *false*)

    - **minReplicas**: Minimum # of replicas (default: *2*)

    - **maxReplicas**: Maximum # of replicas (default: *4*)

    - **metrics**: Metric(s) upon which to make decisions regarding scaling. Must be an array of MetricSpec (https://kubernetes.io/docs/reference/generated/

kubernetes-api/v1.22/#metricspec-v2beta2-autoscaling). Examples are in the helm values.

– **behavior**: Behavior of the HPA, must be instance of HorizontalPodAutoscalerBehavior (https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#horizontalpodautoscalerspec-v2beta2-autoscaling).

- **ringBuffers**: (Ringbuffer configuration)

  – **retries**: Number of shards to attempt to lock without blocking before giving up and simply blocking while waiting for the next shard to be available. Defaults to 5 if there is more than 1 shard, else it defaults to 0. (Dnsdist default)

  – **shards**: Number of shards to use to limit lock contention, defaults to 1. (Dnsdist default)

  – **size**: Maximum amount of queries to keep in the ringbuffer, defaults to 10000. (Dnsdist default)

- **securityPolling**: (Security polling configuration)

  – **securityPollInterval**: Interval between security pollings, in seconds. Defaults to 3600. (Dnsdist default)

  – **securityPollSuffix**: Domain name from which to query security update notifications. Setting this to an empty string disables secpoll. (Dnsdist default)

- **tuning**: (dnsdist tuning functions)

  – **setCacheCleaningDelay**: Interval in seconds between two runs of the cache cleaning algorithm, removing expired entries. (Dnsdist default)

  – **setCacheCleaningPercentage**: Percentage of the cache that the cache cleaning algorithm will try to free by removing expired entries. By default (100), all expired entries are removed. (Dnsdist default)

  – **setMaxTCPClientThreads**: Maximum amount of TCP client threads, handling TCP connections. By default this value is 10, unless more than 10 TCP listen sockets have been defined. (Dnsdist default)

  – **setMaxTCPConnectionDuration**: Maximum duration of an incoming TCP connection, in seconds. 0 (the default) means unlimited. (Dnsdist default)

  – **setMaxTCPConnectionsPerClient**: Maximum number of TCP connections per client. 0 (the default) means unlimited. (Dnsdist default)

  – **setMaxTCPQueriesPerConnection**: Maximum number of queries in an incoming TCP connection. 0 (the default) means unlimited. (Dnsdist default)

  – **setMaxTCPQueuedConnections**: Maximum number of TCP connections queued (waiting to be picked up by a client thread), defaults to 1000. 0 means unlimited. (Dnsdist default)

  – **setMaxUDPOutstanding**: Maximum number of outstanding UDP queries to a given backend server, defaults to 65535. (Dnsdist default)

  – **setStaleCacheEntriesTTL**: TTL for expired cache entries to be eligible as answer when no backends are available for a query, in seconds. (Dnsdist default)

  – **setTCPRecvTimeout**: Read timeout on TCP connections from the client, in seconds. (Dnsdist default)

- **setTCPSendTimeout**: Write timeout on TCP connections from the client, in seconds. (Dnsdist default)

- **setUDPTimeout**: Maximum time dnsdist will wait for a response from a backend over UDP, in seconds. Defaults to 2. (Dnsdist default)

- **agentResources**: (dnsdist agent resource allocation defaults)

  - **limits**: (Limit amounts)

    * **cpu**: Limit amount of CPU (Kubernetes default)

    * **memory**: Limit amount of memory (Kubernetes default)

  - **requests**: (Request amounts)

    * **cpu**: Request amount of CPU (Kubernetes default)

    * **memory**: Request amount of memory (Kubernetes default)

- **rpcServerResources**: (dnsdist RPC server resource allocation defaults)

  - **limits**: (Limit amounts)

    * **cpu**: Limit amount of CPU (Kubernetes default)

    * **memory**: Limit amount of memory (Kubernetes default)

  - **requests**: (Request amounts)

    * **cpu**: Request amount of CPU (Kubernetes default)

    * **memory**: Request amount of memory (Kubernetes default)

- **stateResources**: (dnsdist state resource allocation defaults)

  - **limits**: (Limit amounts)

    * **cpu**: Limit amount of CPU (Kubernetes default)

    * **memory**: Limit amount of memory (Kubernetes default)

  - **requests**: (Request amounts)

    * **cpu**: Request amount of CPU (Kubernetes default)

    * **memory**: Request amount of memory (Kubernetes default)

- **resources**: (dnsdist resource allocation defaults)

  - **limits**: (Limit amounts)

    * **cpu**: Limit amount of CPU (Kubernetes default)

    * **memory**: Limit amount of memory (Kubernetes default)

  - **requests**: (Request amounts)

    * **cpu**: Request amount of CPU (Kubernetes default)

    * **memory**: Request amount of memory (Kubernetes default)

10.3 recursor

Default settings for all recursor instances, configurable under 'recursor' in helm values.

- **affinity**: Kubernetes pod affinity (default: *{}*)

- **agentSecurityContext**: Default Security Context for the agent container (allowPrivilegeEscalation: *false*, readOnlyRootFilesystem: *true*)

- **antiAffinityPreset**: Pod anti affinity, if *affinity* is not set (default: *preferred*)

- **apiKey**: API key used to access the /api endpoint, used to configure a static key (default: generated and stored in a secret)

- **defaultForwardZonePriority**: Default priority assigned to an entry in the forward zones list (default: *100*)

- **initSecurityContext**: Default Security Context for the recursor-init init container (allowPrivilegeEscalation: *false*, readOnlyRootFilesystem: *true*)

- **metricsPath**: Path on which Prometheus metrics will be served (default: */metrics*)

- **metricsPort**: Port on which Prometheus metrics will be served (default: *8082*)

- **nodeSelector**: Kubernetes pod nodeSelector (default: *{}*)

- **podAnnotations**: Annotations to be added to each Pod

- **podLabels**: Labels to be added to each Pod

- **podSecurityContext**: Kubernetes Pod security context (default: *953* as fsGroup, runAsUser & runAsGroup)

- **readinessFailureThreshold**: When a probe fails, Kubernetes will try this many times before marking the container as Unready. Updates deployment, resulting in respawn of recursor pods (default: *2*)

- **readinessInitialDelaySeconds**: Number of seconds after the containers have started before readiness probes are initiated. Updates deployment, resulting in respawn of recursor pods. (default: *5*)

- **readinessPeriodSeconds**: How often (in seconds) to perform the readiness probes. Updates deployment, resulting in respawn of recursor pods (default: *2*)

- **readinessStateProbeInterval**: How often (in seconds) the agent will judge the health state of the recursor agent. (default: *2*)

- **readinessSuccessThreshold**: Minimum consecutive succesfull probes before a container is marked Ready. Updates deployment, resulting in respawn of recursor pods (default: *2*)

- **readinessTimeoutSeconds**: Number of seconds after which the readiness probes time out. Updates deployment, resulting in respawn of recursor pods (default: *1*)

- **recursorSecurityContext**: Default Security Context for the recursor container (allowPrivilegeEscalation: *false*, readOnlyRootFilesystem: *true*)

- **replicas**: Default number of replicas in a recursor deployment (default: *2*)

- **serviceLabels**: Labels to be added to each Service object (default: *{}*)

- **tolerations**: Default Pod tolerations (default: *[]*)

- **webserverACLAllowAll**: Allow all inbound traffic to recursor webserver, regardless of source IP (default: *true*)

- **webserverACL**: Netmasks to allow webserver traffic from (default: *127.0.0.1/32, 192.168.0.0/16, 10.0.0.0/8, 172.16.0.0/12, ::1, fc00::/7*)

- **webserverPassword**: Instead of generating a password for the webserver, set a static one (default: generated and stored in a secret)

- **hpa**: (Horizontal Pod Autoscaler defaults)

    - **enabled**: Enable or disable the HPA (default: *false*)

    - **minReplicas**: Minimum # of replicas (default: *2*)

    - **maxReplicas**: Maximum # of replicas (default: *4*)

    - **metrics**: Metric(s) upon which to make decisions regarding scaling. Must be an array of MetricSpec (`https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#metricspec-v2beta2-autoscaling`). Examples are in the helm values.

    - **behavior**: Behavior of the HPA, must be instance of HorizontalPodAutoscalerBehavior (`https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#horizontalpodautoscalerspec-v2beta2-autoscaling`).

- **agentResources**: (recursor agent resource allocation defaults)

    - **limits**: (Limit amounts)

        * **cpu**: Limit amount of CPU (Kubernetes default)

        * **memory**: Limit amount of memory (Kubernetes default)

    - **requests**: (Request amounts)

        * **cpu**: Request amount of CPU (Kubernetes default)

        * **memory**: Request amount of memory (Kubernetes default)

- **resources**: (recursor resource allocation defaults)

    - **limits**: (Limit amounts)

        * **cpu**: Limit amount of CPU (Kubernetes default)

        * **memory**: Limit amount of memory (Kubernetes default)

    - **requests**: (Request amounts)

        * **cpu**: Request amount of CPU (Kubernetes default)

        * **memory**: Request amount of memory (Kubernetes default)

10.4 resolver

Default settings for resolver instances, configurable under 'resolver' in helm values.

- **backendDefaultPort**: Port on which the endpoints belonging to a resolver partition will receive traffic (default: *53*)

- **serviceLabels**: Labels to be added to each Service object (default: *{}*)

- **servicePort**: Port on which a resolver service will listen (unused in normal operation, as the service is only used to discover the corresponding endpoints). (default: *53*)

10.5 zoneControl

Default settings for all ZoneControl instances, configurable under 'zonecontrol' in helm values.

- **affinity**: Kubernetes pod affinity (default: *{}*)

- **antiAffinityPreset**: Pod anti affinity, if *affinity* is not set (default: *preferred*)

- **initSecurityContext**: Default Security Context for the zc-init init container (allowPrivilegeEscalation: *false*, readOnlyRootFilesystem: *true*)

- **nginxSecurityContext**: Default Security Context for the nginx container (allowPrivilegeEscalation: *false*, readOnlyRootFilesystem: *true*)

- **nodeSelector**: Kubernetes pod nodeSelector (default: *{}*)

- **podAnnotations**: Annotations to be added to each Pod

- **podLabels**: Labels to be added to each Pod

- **podSecurityContext**: Kubernetes Pod security context (default: *953* as fsGroup, runAsUser & runAsGroup)

- **replicas**: Default number of replicas in a recursor deployment (default: *2*)

- **serviceLabels**: Labels to be added to each Service object (default: *{}*)

- **syncJobTTL**: Keep finished synchronisation jobs for this amount of seconds before allowing Kubernetes to remove them (default: *86400*)

- **tolerations**: Default Pod tolerations (default: *[]*)

- **zcSecurityContext**: Default Security Context for the zc container (allowPrivilegeEscalation: *false*, readOnlyRootFilesystem: *true*)

- **zcOperatorSecurityContext**: Default Security Context for the zc operator container (allowPrivilegeEscalation: *false*, readOnlyRootFilesystem: *true*)

- **operator**: (Synchronisation operator defaults)

    - **podAnnotations**: Annotations to be added to each Pod

    - **podLabels**: Labels to be added to each Pod

- **resources**: (auth resource allocation defaults)

    - **limits**: (Limit amounts)

        * **cpu**: Limit amount of CPU (Kubernetes default)

* **memory**: Limit amount of memory (Kubernetes default)

– **requests**: (Request amounts)

* **cpu**: Request amount of CPU (Kubernetes default)

* **memory**: Request amount of memory (Kubernetes default)

11 Instances

11.1 auths

Configuration of auth instances

Key: name (Name of the auth instance)

11.1.1 Parameters

- **affinity**: Kubernetes Pod affinity

- **agentLogLevel**: Agent log level, defaults to *info*. Available options: *debug*, *info*, *warn*, *error*

- **agentLogFormat**: Agent log format, defaults to *text*. Available options: *text*, *json*

- **agentResources**: Resources allocated to the 'agent' container

- **antiAffinityPreset**: Pod anti affinity preset (Accepted values: *preferred* or *required*)

- **apiKey**: API key used to access the /api endpoint, used to configure a static key (default: generated and stored in a secret)

- **apiKeySecret**: Name of the secret from which to take the API key, used when there is a pre-provisioned secret holding the api key

- **apiKeySecretItem**: Name of the data item within *apiKeySecret* which holds the API key, used when there is a pre-provisioned secret holding the api key (default: *api_key*)

- **containerSecurityContext**: Container security context for all containers within a Pod (see 'Security contexts' chapter)

- **hostNetwork**: Use host networking for auth pods

- **initResources**: Resources allocated to the 'auth-init' container

- **lightningStreamResources**: Resources allocated to the 'ls' container

- **nodeSelector**: Kubernetes pod nodeSelector

- **podAnnotations**: Annotations to be added to each Pod

- **podLabels**: Labels to be added to each Pod

- **podSecurityContext**: Kubernetes Pod security context (see 'Security contexts' chapter)

- **replicas**: Default number of replicas in a auth deployment (default: *2*)

- **resources**: Resources allocated to the 'auth' container

- **revisionHistoryLimit**: Default 'revisionHistoryLimit' for auth deployments (default: *0*)

- **tolerations**: Kubernetes Pod tolerations

- **volumes**: List of additional volumes (see *Storage* chapter)

11.1.2 Parameter Sets

- **api** (Configuration of access to the auth API endpoint)

    – **ingress**: Ingress object to expose the auth API endpoint via an ingress managed by an ingress controller (For formatting example, see the *Exposing auth API* chapter)

    – **service**: Service object to expose the auth API endpoint (For formatting example, see the *Exposing dnsdist* chapter, which covers the *service* format to add ClusterIP, NodePort or LoadBalancer services)

- **dnsdist** (Settings to be applied to each auth instance when added to dnsdist as a server)

    – **addXPF**: Add the client's IP address and port to the query, along with the original destination address and port. Default is disabled (0)

    – **checkClass**: Number to use as QCLASS in the health-check query, default is DNSClass.IN

    – **checkInterval**: The time in seconds between health checks

    – **checkName**: String to use as QNAME in the health-check query, default is "a.root-servers.net."

    – **checkTimeout**: The timeout (in milliseconds) of a health-check query, default to 1000 (1s)

    – **checkType**: String to use as QTYPE in the health-check query, default is "A"

    – **disableZeroScope**: Disable the EDNS Client Subnet 'zero scope' feature, which does a cache lookup for an answer valid for all subnets (ECS scope of 0) before adding ECS information to the query and doing the regular lookup. This requires the *parseECS* option of the corresponding cache to be set to true

    – **healthCheckMode**: Type of health-check to perform, default is *auto* which is configured using the checkName, checkType, etc parameters. Alternatives are *up* (no healthcheck - always available for traffic) and *down* (no healthcheck - never available for traffic)

    – **maxCheckFailures**: Allow this amount of check failures before declaring the back-end down, default is 1

    – **mustResolve**: Set to true when the health check MUST return a RCODE different from NXDomain, ServFail and Refused. Default is false, meaning that every RCODE except ServFail is considered valid

    – **order**: The order of servers in this set, used by the *leastOutstanding* and *firstAvailable* policies

    – **qps**: Limit the number of queries per second to this amount, when using the *firstAvailable* policy

    – **reconnectOnUp**: Close and reopen the sockets when a server transits from Down to Up. This helps when an interface is missing when dnsdist is started. Default is false

- **retries**: The number of TCP connection attempts to servers in this set, for a given query

- **rise**: Require NUM consecutive successful checks before declaring the backend up, default is 1

- **setCD**: Set the CD (Checking Disabled) flag in the health-check query, default is false

- **sockets**: Number of sockets (and thus source ports) used toward the backend server, defaults to 1

- **source**: Name of the interface which Dnsdist will use to try to send traffic to this Recursor

- **tcpConnectTimeout**: The timeout (in seconds) of a TCP connection attempt

- **tcpFastOpen**: Whether to enable TCP Fast Open

- **tcpRecvTimeout**: The timeout (in seconds) of a TCP read attempt

- **tcpSendTimeout**: The timeout (in seconds) of a TCP write attempt

- **useClientSubnet**: Add the client's IP address in the EDNS Client Subnet option when forwarding the query to this backend

- **useProxyProtocol**: Add a proxy protocol header to the query, passing along the client's IP address and port along with the original destination address and port. Default is false

- **weight**: The weight of servers in this set, used by the *wrandom*, *whashed* and *chashed* policies, default is 1

- **config** (Auth configuration, any configuration item listed in the auth documentation (https://doc.powerdns.com/authoritative/settings.html) can be referenced here. Settings which are explicitly ignored due to conflicts with Cloud Control are filtered by 'webserver*', 'local*', 'disable-syslog', 'daemon', 'chroot', 'socket*', 'config*', 'primary', 'secondary' and 'autosecondary')

- **hpa** (Horizontal Pod Autoscaler - see helm values 'auth.hpa' for examples)

  - **enabled**: Enable or disable the HPA (default: *false*)

  - **minReplicas**: Minimum # of replicas (default: *2*)

  - **maxReplicas**: Maximum # of replicas (default: *4*)

  - **metrics**: Metric(s) upon which to make decisions regarding scaling. Must be an array of MetricSpec (https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#metricspec-v2beta2-autoscaling). Examples are in the helm values.

  - **behavior**: Behavior of the HPA, must be instance of HorizontalPodAutoscalerBehavior (https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#horizontalpodautoscalerspec-v2beta2-autoscaling).

- **postgres** (Settings to be applied to each postgres cluster/database when an auth deployment requests a postgres operator-managed database)

  - **parameters.max_connections**: Max connections parameter (default: *128*)

  - **storage.size**: Size of the volume to request for each Postgres pod (default: *5Gi*)

- **storage.storageClassName**: Name of the Storage Class for the volumes in which Postgres data will be stored (default: unset, which means the Kubernetes cluster's default sotrage class)

- **resources.limits.cpu**: Limit amount of CPU (default: *250m*)

- **resources.limits.memory**: Limit amount of memory (default: *256Mi*)

- **resources.requests.cpu**: Request amount of CPU (default: *500m*)

- **resources.requests.memory**: Request amount of memory (default: *512m*)

- **nodeAffinity**: Kubernetes node affinity (Note: Only configures the subset *nodeAffinity* of the parent *affinity*. See `https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/#node-affinity`)

11.1.3 Backends

When deploying Auth as part of CloudControl there are several backends you can choose (and combine, multiple backends is supported):

- **GeoIP**: This backend allows visitors to be sent to a server closer to them, with no appreciable delay, as would otherwise be incurred with a protocol level redirect.

- **LightningStream**: This backend allows synchronisation of multiple instances via an s3 bucket. This can be done within a single datacenter or spread over multiple datacenters.

- **MySQL**: This backend allows storing zones in a MySQL database.

- **Postgres**: This backend allows storing zones in a Postgres database, either pre-existing on your cluster or created at runtime using a Postgres Operator.

11.1.3.1 Postgres (pre-existing)

If you have an existing Postgres database available, you can configure a backend to utilize it. Configuring a pre-existing Postgres database can be done via the following configuration:

Format:

```
backends:
  - type: postgres
    host: host-of-postgres-cluster
    dbname: auth
    user: some_user
    password: some_user_password
```

Available parameters:

- type - Must be set to *postgres*

- host - Host of the Postgres cluster/service

- port - Port over which to access the Postgres database (Optional, default: *5432*)

- dbname - Name of the database

- sslmode - Postgres *sslmode* to use while connecting (Optional, omitted if no value is provided)

Parameters - database credentials:

You have to provide credentials to access the database, this can be done in 2 ways:

Inline username and pasword:

- user - Required, user with which to connect
- password - Required, password for *user*

Username and password from a pre-existing secret:

- secretName - Required, name of existing secret containing database credentials
- secretUsernameKey - Optional, name of the item inside the Secret which holds the username (Default: *username*)
- secretPasswordKey - Optional, name of the item inside the Secret which holds the password (Default: *password*)

Note: Cloud Control will manage the schema in the Postgres database. Before an Auth container starts, an init container checks the schema and creates/updates it if necessary.

11.1.3.2 Postgres (Operator-managed)

If you have deployed the postgres-operator chart to your Kubernetes cluster, you can configure a backend to use a operator-managed Postgres database. Configuring such a Postgres database can be done via the following configuration:

Format:

```
backends:
  - type: postgres
    operator: true
```

Available parameters:

- type – Must be set to *postgres*
- operator - Must be set to *true*

The Postgres database which is created utilizes the default values which are exposed via the helm values in *auth.postgres*.

11.1.3.3 MySQL

If you have an existing MySQL database available, you can configure a backend to utilize it. Configuring a pre-existing MySQL database can be done via the following configuration:

Format:

```
backends:
  - type: mysql
    host: host-of-mysql-cluster
    dbname: auth
    user: some_user
    password: some_user_password
```

Available parameters:

- type – Must be set to *mysql*

- host - Host/IP of the MySQL cluster/service

- port - Port over which to access the MySQL database (Optional, default: *3306*)

- dbname - Name of the database

- ssl - Can be set to *yes* to force connecting via SSL (Optional, default behaviour is to use SSL if the server announces this capability)

- group - Group to connect as (Optional, default: *client*)

- dnssec - Whether or not to enable DNSSEC processing for this backend (Optional, default: *no*)

- innodbReadCommitted - Use the InnoDB READ-COMMITTED transaction isolation level (Optional, default: *yes*)

- timeout - Timeout in seconds for each attempt to read from, or write to the server. A value of 0 will disable the timeout (Optional, default: *10*)

Parameters - database credentials:

You have to provide credentials to access the database, this can be done in 2 ways:

Inline username and pasword:

- user - Required, user with which to connect

- password - Required, password for *user*

Username and password from a pre-existing secret:

- secretName - Required, name of existing secret containing database credentials

- secretUsernameKey - Optional, name of the item inside the Secret which holds the username (Default: *username*)

- secretPasswordKey - Optional, name of the item inside the Secret which holds the password (Default: *password*)

Note: Cloud Control will manage the schema in the MySQL database. Before an Auth container starts, an init container checks the schema and creates/updates it if necessary.

11.1.3.4 GeoIP

Configuring a GeoIP backend requires configuring the following options:

Format:

```
backends:
  - type: geoip
    databases: <Configuration of one or more GeoIP databases>
    domains: <Configuration of domains>
```

Configuring *domains*: Please refer to the PowerDNS Authoritative Server documentation at https://doc.powerdns.com/authoritative/backends/geoip.html#keys-explained

Configuring *databases*:

The databases used by the GeoIP backend are MMDB files, which allows you to use databases from several sources:

- MaxMind – A provider of GeoIP databases
- Custom - A database built by the Auth init container based on data you provide in the Helm configuration values

11.1.3.5 GeoIP - MaxMind Database

Configuring a GeoIP backend using a MaxMind database can be done via 2 methods:

- HTTP - Download the database using HTTP
- OCI - Download the database from an OCI-enabled registry, using the ORAS Client

11.1.3.6 GeoIP - MaxMind Database - HTTP

Format:

```
databases:
  - name: MyDatabase
    type: http
    url: https://my_host/folder/my-database.mmdb
    insecure: false
    headers:
      <List of additional headers to be passed>
    params:
      <List of additional parameters to be passed, these must be valid Curl parameters>
```

Explanation of each parameter:

- name - Required, must be unique for this backend
- type - Required, must be *http* to force a MaxMind HTTP download
- url - Required, full address of the database to be downloaded
- insecure - Optional, set to *true* if you need to download the database from an HTTPS URL without valid certificates
- headers - Dict of *key: value* pairs, where *key* is the header name and *value* is the value of the header
- params - Dict of *key: value* pairs, where *key* is the name of the Curl parameter and *value* is the value of the parameter

Minimal required configuration for downloading a MaxMind database using HTTP(S):

```
databases:
  - name: MyDatabase
    type: http
    url: https://some_host/my-database.mmdb
```

Example using headers & params:

```
databases:
  - name: MyDatabase
    type: http
    url: https://some_host/my-database.mmdb
    headers:
      Authorization: Basic bXl1sfdc2VyOm15cGFzetbcvbc3sdfdvcmQ=
    params:
      proxy: "http://my_user:my_password@some_proxy:8080"
```

Above example will send a Basic auth header & attempt to connect via the proxy *some_proxy:8080*

11.1.3.7 GeoIP - MaxMind Database - OCI

Format:

```
databases:
  - name: MyOCIDB
    type: oci
    artifact: my_registry/repository/geo-city-mmdb:v2.4.0
    user: MyUser
    token: MyPassword
```

Explanation of each parameter:

- name - Required, must be unique for this backend

- type - Required, must be *oci* to force a MaxMind OCI download

- artifact - Required, full address of the database to be downloaded

- user - Required, User with which to connect to the registry

- token - Required, Token/Password for *user*

11.1.3.8 GeoIP - Custom Database

Configuring a GeoIP backend using a custom database can be done via the following configuration:

```
databases:
  - name: localdb
    type: generate
    datacenters:
      <List of datacenters - see below>
```

Explanation of each parameter:

- name - Required, must be unique for this backend

- type - Required, must be *generate* to force a custom database to be generated

- datacenters - Required, must be populated according to the below format

Example datacenter configuration:

```
- name: Amsterdam
  networks:
    - 10.0.1.0/24
    - 10.0.2.0/24
    - 10.0.3.0/24
    - 10.0.4.0/24
    - 10.0.5.0/24
    - 10.0.6.0/24
  lat: 52.370216
  long: 4.895168
```

Explanation of each parameter:

- name - Required, must be unique for this datacenter

- networks - Required, list of networks to be assigned to this datacenter

- lat - Required, longitude of this datacenter

- long - Required, latitude of this datacenter

The latitude & longitude are used to calculate distances between datacenters, which is how the GeoIP backend determines the closest available recipient of traffic. There are many tools available online to obtain the latitude & longitude of locations.

The following example simulates a setup where we have 2 datacenters, in Amsterdam & Rotterdam:

```
databases:
  - name: localdb
    type: generate
    datacenters:
      - name: Amsterdam
        networks:
          - 10.0.1.0/24
          - 10.0.2.0/24
          - 10.0.3.0/24
          - 10.0.4.0/24
          - 10.0.5.0/24
          - 10.0.6.0/24
        lat: 52.370216
        long: 4.895168
      - name: Rotterdam
        networks:
          - 10.1.1.0/24
          - 10.1.2.0/24
          - 10.1.3.0/24
          - 10.1.4.0/24
          - 10.1.5.0/24
          - 10.1.6.0/24
        lat: 51.924419
        long: 4.477733
```

11.1.3.9 LightningStream

Configuration of a LightningStream backend primarily requires supplying the details required to connect to a shared s3 bucket.

Notes:

- Account used to access the s3 bucket must have privileges to read, write & list

- All auth instances that have the same s3 bucket configured will contain the same zones & records. You cannot share the same s3 bucket amongst auth instances which you intend to have different data.

- Access and Secret keys must be limited to characters from the Base64 alphabet (defined in RFC 4648). Usage of characters outside this alphabet can lead to inability to connect to the s3 bucket.

The format to configure a LightningStream backend:

```
backends:
  - type: ls
    access_key: MY_ACCESS_KEY
    secret_key: MY_SECRET_KEY
    bucket: mybucket
    endpoint: https://my.s3.endpoint
```

Required parameters - s3 endpoint:

- type - Required, must be *ls* to force a LightningStream backend

- bucket - Required, name of the s3 bucket

- endpoint - Required, URL to access the s3 service

Required parameters - credentials:

You have to provide credentials to access the above described bucket. This can be done in 2 ways:

Inline access & secret key:

- access_key - Required, access key to authenticate with the endpoint

- secret_key - Required, secret key to authenticate with the endpoint

Access & secret key from a pre-existing Secret:

- secretName - Required, name of an existing secret containing s3 credentials

- secretAccessKey - Optional, name of the item inside the Secret which holds the access key (Default: *access_key*)

- secretSecretKey - Optional, name of the item inside the Secret which holds the secret key (Default: *secret_key*)

Note: If you need to change the credentials in the pre-existing Secret after initial deployment, it can take up to a minute for Kubernetes to synchronize the Secret to each Pod's filesystem.

Optional parameters:

- cleanupDisabled - Set to *true* to disable the cleanup mechanism (Default: *false*)

- cleanupInterval - Interval between cleanup runs (Default: *5m*)

- cleanupMustKeepInterval - Determines how long snapshots must be kept after they appear in the bucket, even if a newer snapshot is available. Defaults to *10m*

- cleanupRemoveOldInstancesInterval - Determines when an instance is considered stale and the latest snapshot for the instance can be considered for removal. Defaults to *168h* (7 days)

- health - Allows for configuration of the error and warning thresholds used to determine readiness of LightningStream via *unreadyIfError*, *unreadyIfWarning* and *waitForInitialSync* (see *health configuration* section below)

- loglevel - Defaults to *info*. Available options: *debug*, *info*, *warning*, *error*, *fatal*

- lmdbEmptyDir - Configuration of the *EmptyDir* volume used to store LMDB. Defaults to: *medium: "Memory"*

- lmdbPollInterval - Minimum time between checking for new LMDB transactions. Defaults to *1s*

- lmdbLogStatsInterval - Interval for logging LMDB stats. Defaults to *30m*

- mapSize - Size (in megabytes) of the LMDB databases. Defaults to *1000* and can be increased for large datasets

- memoryDownloadedSnapshots - Maximum amount of downloaded snapshots queued in memory before decompression. Defaults to *3*

- memoryDecompressedSnapshots - Maximum amount of decompressed snapshots in memory before processing. Defaults to *2*

- migratorInterval - Time between checks performed by the migrator to see if a migration is required. Defaults to *5s*

- migratorLogFormat - Migrator log format, defaults to *text*. Available options: *text*, *json*

- migratorLogLevel - Migrator log level, defaults to *info*. Available options: *trace*, *debug*, *info*, *warn*, *error*

- migratorPodAnnotations - Annotations to be added to the Migrator Pod

- migratorPodLabels - Labels to be added to the Migrator Pod

- mode - Synchronization mode, defaults to *sync*. Available options: *sync*, *receive*

- region - Set this to the region associated with your s3 bucket, if your s3 service requires this

- removeOldSchemaAge - Time since the last successful migration of an old schema before removing it, if no updates have been made to the schema since then. Defaults to *672h* `

- removeOldSchemaEnabled - Whether or not the migrator should clean up old schemas if they have been migrated successfully and not seen any updates since. Defaults to *true*

- storageForceSnapshotInterval - Interval between forced snapshots created by an instance, if no changes have been observed. Defaults to *4h*

- storagePollInterval - Minimum time between polling the storage backend for new snapshots. Defaults to *1s*

- storageRetryInterval - Interval to retry a storage operation after failure. Defaults to *5s*

- storageRetryCount - Number of times to retry a storage operation after failure, before giving up (only active if storageRetryForever is *false*). Defaults to *10*

- storageRetryForever - Retry failed storage operations forever before giving up. Defaults to *true*

- tls - Allows for configuration of a trusted CA and/or disable certificate validation (see *TLS Configuration* section below)

- unreadyIfError - Set readiness of LightingStream container to *NotReady* if LightningStream healthz endpoint reports any errors (Default: *true*)

- unreadyIfWarning - Set readiness of LightingStream container to *NotReady* if LightningStream healthz endpoint reports any warnings (Default: *false*)

- waitForInitialSync - Keep readiness of LightningStream container at *NotReady* until LightningStream has successfully finished the initial sync between the Auth container and the s3 bucket (Default: *true*)

LightningStream - Synchronization mode

Valid modes for synchronization are:

- sync (default) - Lightning Stream synchronizes changes made locally towards the s3 bucket and pulls in changes from the s3 bucket (ie: read & write)

- receive - Lightning Stream pulls in changes from the s3 bucket, but never writes to it (ie: read)

Note: Since *receive* mode does not write changes to the s3 bucket, there is no Migrator Pod deployed when this mode is selected

Example: Set *receive* mode

```
backends:
  - type: ls
    mode: receive
    access_key: MY_ACCESS_KEY
    secret_key: MY_SECRET_KEY
    bucket: mybucket
    endpoint: https://my.s3.endpoint
```

Since *receive* mode only attempts to list and get the contents of the s3 bucket, the amount of privileges required to run are also reduced. The privileges required on an s3 account used in a *receive* mode deployment are limited to:

- s3:ListBucket - To list the contents of a bucket

- s3:GetObject - To download an object from the bucket

LightningStream - TLS Configuration

Valid TLS options are:

- add_system_ca_pool - Whether or not to load the LightningStream image's embedded trusted CA certs. Defaults to 'true' if the 'ca' option is not used. Defaults to 'false' when the 'ca' option is supplied.

- ca - PEM formatted CA certificate to trust

- insecure_skip_verify - Defaults to 'false', set this to 'true' to disable certificate validation

Note: Using the 'insecure_skip_verify' flag is not recommended for production deployments as it nullifies the security benefits of using the SSL/TLS certificates on an HTTPS endpoint.

TLS example: Skip certificate validation

```
backends:
  - type: ls
    access_key: MY_ACCESS_KEY
    secret_key: MY_SECRET_KEY
    bucket: mybucket
    endpoint: https://my.s3.endpoint
    tls:
      insecure_skip_verify: true
```

TLS example: Provide custom CA to trust

```
backends:
  - type: ls
    access_key: MY_ACCESS_KEY
    secret_key: MY_SECRET_KEY
    bucket: mybucket
    endpoint: https://my.s3.endpoint
    tls:
      ca: |
        -----BEGIN CERTIFICATE-----
        MIICdjCCAhygAwIBAgIQEuVhXI/nMis4pfWxPMvPbDAKBggqhkjOPQQDAjAjMSEw
        HwYDVQQDDBhrM3Mtc2VydmVyLWNhQDE2NTI4NjE4NzEwHhcNMjIwNjAxMTIyNTEw
        WhcNMjMwNjAxMTIyNTEwWjBWMRUwEwYDVQQKEwxzeXN0ZW06bm9kZXMxPTA7BgNV
        BAMMNHN5c3RlbTpub2RlOioubWluaW90ZW5bnQtaGwubWluaW8uc3ZjLmNsdXN0
        ZXIubG9jjYWwwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAQnhNTJcv4z8l1mfBlV
        +xCyQiyrhZFNFE4GtZ1SuhlRFJj2Ze+LK7ySRHt8DbS75r+eF+GpKGhfhjAA3HXB
        KNFNo4H+MIH7MA4GA1UdDwEB/wQEAwIFoDATBgNVHSUEDDAKBggrBgEFBQcDATAM
        BgNVHRMBAf8EAjAAMB8GA1UdIwQYMBaAFEpXmMnwHdQ748wFl7IwQbLLyl28MIGk
        BgNVHREEgZwwgZmCO21pbmlvdGVuYW50LXBvb2wtMC0wLm1pbmlvdGVuYW50LWhs
        Lm1pbmlvLnN2Yy5jbHVzdGVyLmxvY2Fsgh1taW5pby5taW5pby5zdmMuY2x1c3Rl
        ci5sb2NhbIIILbWluaW8ubWluaW+CD21pbmlvLm1pbmlvLnN2Y4ICKi6CGSoubWlu
        aW8uc3ZjLmNsdXN0ZXIubG9jYWwwCgYIKoZIzj0EAwIDSAAwRQIhAMnZzOBRyHLQ
        vsYo3mhbF45SGv56y5DA6CqUT6/Jng2cAiB/skNx4xK1T5w2ViZD23LPx8ydzwOi
        i9S8Yqvl0BzOIA==
        -----END CERTIFICATE-----
```

TLS example: Provide custom CA to trust + trust LightningStream image's embedded trusted CA certs

```
backends:
  - type: ls
```

```
    access_key: MY_ACCESS_KEY
    secret_key: MY_SECRET_KEY
    bucket: mybucket
    endpoint: https://my.s3.endpoint
    tls:
      add_system_ca_pool: true
      ca: |
        -----BEGIN CERTIFICATE-----
        MIICdjCCAhygAwIBAgIQEuVhXI/nMis4pfWxPMvPbDAKBggqhkjOPQQDAjAjMSEw
        HwYDVQQDDBhrM3Mtc2VydmVyLWNhQDE2NTI4NjE4NzEwHhcNMjIwNjAxMTIyNTEw
        WhcNMjMwNjAxMTIyNTEwWjBWMRUwEwYDVQQKEwxzeXN0ZW06bm9kZXMxPTA7BgNV
        BAMMNHN5c3RlbTpub2RlOioubWluaW90ZW5hbnQtGwubWluaW8uc3ZjLmNsdXN0
        ZXIubG9jYWwwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAQnhNTJcv4z8l1mfBlV
        +xCyQiyrhZFNFE4GtZ1SuhlRFJj2Ze+LK7ySRHt8DbS75r+eF+GpKGhfhjAA3HXB
        KNFNo4H+MIH7MA4GA1UdDwEB/wQEAwIFoDATBgNVHSUEDDAKBggrBgEFBQcDATAM
        BgNVHRMBAf8EAjAAMB8GA1UdIwQYMBaAFEpXmMnwHdQ748wFl7IwQbLLyl28MIGk
        BgNVHREEgZwwgZmCO21pbmlvdGVuYW50LXBvb2wtMC0wLm1pbmlvdGVuYW50LWhs
        Lm1pbmlvLnN2Yy5jbHVzdGVyLmxvY2Fsgh1taW5pby5taW5pby5zdmMuY2x1c3Rl
        ci5sb2NhbIILbWluaW8ubWluaW+CD21pbmlvLm1pbmlvLnN2Y4ICKi6CGSoubWlu
        aW8uc3ZjLmNsdXN0ZXIubG9jYWwwCgYIKoZIzj0EAwIDSAAwRQIhAMnZzOBRyHLQ
        vsYo3mhbF45SGv56y5DA6CqUT6/Jng2cAiB/skNx4xK1T5w2ViZD23LPx8ydzwOi
        i9S8Yqvl0BzOIA==
        -----END CERTIFICATE-----
```

LightningStream - Health Configuration

Default health configuration settings are as follows:

```
health:
  storage_list:
    warn_duration: 30s
    error_duration: 2m
    interval: 1s
  storage_load:
    warn_duration: 30s
    error_duration: 2m
    interval: 1s
  storage_store:
    warn_duration: 30s
    error_duration: 2m
    interval: 1s
  start:
    warn_duration: 30s
    error_duration: 2m
    interval: 1s
```

The 4 main sections each have their own purpose:

- storage_list: Thresholds for reporting errors/warnings regarding the ability to list snapshots in the S3 bucket

- storage_load: Thresholds for reporting errors/warnings regarding the ability to retrieve snapshots from the S3 bucket

- storage_store: Thresholds for reporting errors/warnings regarding the ability to store

snapshots in the S3 buckets

- start: Thresholds for reporting errors/warnings regarding the initial sync performed when LightningStream starts

Each section has the same 3 options:

- warn_duration: Duration after which a failing operation will report as being in *warning* state

- error_duration: Duration after which a failing operation will report as being in *error* state

- interval: Interval between evalutation of the state of the operation

For example, a configuration as follows for *load*:

```
storage_load:
  warn_duration: 30s
  error_duration: 2m
  interval: 1s
```

Would lead to the following health monitoring:

```
Every 1 second
  If this operation has failed for more than 2 minutes, report error
  Else
    If this operation has failed for more than 30 seconds, report warning
    Else do not report any warnings/errors for this operation
```

Overriding these defaults can be done as follows:

```
backends:
  - type: ls
    access_key: MY_ACCESS_KEY
    secret_key: MY_SECRET_KEY
    bucket: mybucket
    endpoint: https://my.s3.endpoint
    health:
      storage_load:
        warn_duration: 2m
        error_duration: 5m
        interval: 5s
```

11.2 dnsdists

Configuration of dnsdist instances

Key: name (Name of the dnsdist instance)

11.2.1 Parameters

- **aclAdd**: Netmasks allowed to access dnsdist, in addition to the loopback, RFC1918 and other local addresses. Overrules a AllowAll setting (if configured) on the global dnsdist defaults.

- **aclAllowAll**: Allow all inbound traffic to dnsdist, regardless of source IP.

- **affinity**: Kubernetes pod affinity

- **agentLogLevel**: Agent log level, defaults to *info*. Available options: *debug*, *info*, *warn*, *error*

- **agentLogFormat**: Agent log format, defaults to *text*. Available options: *text*, *json*

- **agentResources**: Resources allocated to the 'agent' container

- **antiAffinityPreset**: Pod anti affinity preset (Accepted values: *preferred* or *required*)

- **apiKey**: API key used to access the /api endpoint, used to configure a static key (default: generated and stored in a secret)

- **apiKeySecret**: Name of the secret from which to take the dnsdist API key, used when there is a pre-provisioned secret holding the api key

- **apiKeySecretItem**: Name of the data item within *apiKeySecret* which holds the dnsdist API key, used when there is a pre-provisioned secret holding the api key (default: *api_key*)

- **containerSecurityContext**: Container security context for all containers within a Pod (see 'Security contexts' chapter)

- **do53Locals**: Default amount of Do53 listen sockets per dnsdist pod (default: *1*)

- **do53TcpFastOpenQueueSize**: Default size of the TCP Fast Open queue on Do53 listen sockets (Dnsdist default)

- **do53TcpListenQueueSize**: Default size of the listen queue on Do53 listen sockets (Dnsdist default)

- **hostNetwork**: Use host networking for dnsdist pods

- **initResources**: Resources allocated to the 'dnsdist-init' container

- **luaScript**: Lua script to be included in each dnsdist pod (See 'Overview' document for usage examples)

- **nodeSelector**: Kubernetes pod nodeSelector

- **podAnnotations**: Annotations to be added to each Pod

- **podLabels**: Labels to be added to each Pod

- **podSecurityContext**: Kubernetes Pod security context (see 'Security contexts' chapter)

- **replicas**: Default number of replicas in a dnsdist deployment (default: *2*)

- **resources**: Resources allocated to the 'dnsdist' container

- **revisionHistoryLimit**: Default 'revisionHistoryLimit' for dnsdist deployments (default: *0*)

- **rpcServerResources**: Resources allocated to the 'rpc-server' container

- **serverPurgeDelay**: Delay (in seconds) after which replaced servers are purged from dnsdist (default: *5*)

- **stateResources**: Resources allocated to the 'dnsdist-state' container

- **tolerations**: Kubernetes Pod tolerations

- **verbose**: Be verbose (default: *false*)

- **volumes**: List of additional volumes (see *Storage* chapter)

11.2.2 Parameter Sets

- **config** (Dnsdist Configuration)

  - **addEDNSToSelfGeneratedResponses**: Whether to add EDNS to self-generated responses, provided that the initial query had EDNS. (Dnsdist default)

  - **allowEmptyResponse**: Set to true (defaults to false) to allow empty responses (qdcount=0) with a NoError or NXDomain rcode (default) from backends. dnsdist drops these responses by default because it can't match them against the initial query since they don't contain the qname, qtype and qclass, and therefore the risk of collision is much higher than with regular responses. (Dnsdist default)

  - **consistentHashingBalancingFactor**: Maximum imbalance between the number of outstanding queries intended for a given server, based on its weight, and the actual number, when using the chashed consistent hashing load-balancing policy. Default is 0, which disables the bounded-load algorithm. (Dnsdist default)

  - **payloadSizeOnSelfGeneratedAnswers**: Set the UDP payload size advertised via EDNS on self-generated responses. In accordance with RFC 6891, values lower than 512 will be treated as equal to 512. (Dnsdist default)

  - **roundRobinFailOnNoServer**: By default the roundrobin load-balancing policy will still try to select a backend even if all backends are currently down. Setting this to true will make the policy fail and return that no server is available instead. (Dnsdist default)

  - **servFailWhenNoServer**: If set, return a ServFail when no servers are available, instead of the default behaviour of dropping the query. (Dnsdist default)

  - **verboseHealthChecks**: Set whether health check errors should be logged. This is turned off by default. (Dnsdist default)

  - **weightedBalancingFactor**: Maximum imbalance between the number of outstanding queries intended for a given server, based on its weight, and the actual number, when using the whashed or wrandom load-balancing policy. Default is 0, which disables the bounded-load algorithm. (Dnsdist default)

- **ecs** (dnsdist ECS functions)

  - **setECSOverride**: Whether to override an existing EDNS Client Subnet option present in the query (only effective when the backend server has annotation useClientSubnet = true). (Dnsdist default)

  - **setECSSourcePrefixV4**: Truncate the requestors IPv4 address to this length, in bits (only effective when the backend server has annotation useClientSubnet = true). (Dnsdist default)

- **setECSSourcePrefixV6**: Truncate the requestors IPv6 address to this length, in bits (only effective when the backend server has annotation useClientSubnet = true). (Dnsdist default)

- **hpa** (Horizontal Pod Autoscaler - see helm values 'dnsdist.hpa' for examples)

  - **enabled**: Enable or disable the HPA (default: *false*)

  - **minReplicas**: Minimum # of replicas (default: *2*)

  - **maxReplicas**: Maximum # of replicas (default: *4*)

  - **metrics**: Metric(s) upon which to make decisions regarding scaling. Must be an array of MetricSpec (https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#metricspec-v2beta2-autoscaling). Examples are in the helm values.

  - **behavior**: Behavior of the HPA, must be instance of HorizontalPodAutoscalerBehavior (https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#horizontalpodautoscalerspec-v2beta2-autoscaling).

- **readiness** (Readiness probe Configuration)

  - **bindInterface**: Name of interface to which the readiness probe will bind

  - **disabled**: Whether or not to instruct Kubernetes to poll the agent for readiness state. (default: *false*)

  - **do53ProbeInterval**: How often (in seconds) the agent will judge the health state of dnsdist via tests against the Do53 listeners. (default: *2*)

  - **do53QDomain**: Domain used in the query to judge whether the Do53 listeners of a dnsdist container are healthy and ready for traffic. (default: *a.root-servers.net*)

  - **do53QTimeout**: Number of seconds after which the query used to judge whether the Do53 listeners of a dnsdist container are healthy and ready for traffic time out. (default: *1*)

  - **do53QType**: Type of query used to judge whether the Do53 listeners of a dnsdist container are healthy and ready for traffic. (default: *A*)

  - **failureThreshold**: When a probe fails, Kubernetes will try this many times before marking the container as Unready. Updates deployment, resulting in respawn of dnsdist pods (default: *2*)

  - **initialDelaySeconds**: Number of seconds after the containers have started before readiness probes are initiated. Updates deployment, resulting in respawn of dnsdist pods. (default: *5*)

  - **periodSeconds**: How often (in seconds) to perform the readiness probes. Updates deployment, resulting in respawn of dnsdist pods (default: *2*)

  - **stateProbeInterval**: How often (in seconds) the agent will judge the health state of the dnsdist agent. (default: *2*)

  - **successThreshold**: Minimum consecutive succesfull probes before a container is marked Ready. Updates deployment, resulting in respawn of dnsdist pods (default: *2*)

  - **timeoutSeconds**: Number of seconds after which the readiness probes time out. Updates deployment, resulting in respawn of dnsdist pods (default: *1*)

- **recursor** (Enable & configure recursor containers co-hosted within the dnsdist pod)

  – **backendsPerServer**: Amount of server objects to create in dnsdist for each recursor container (default: *1*)

  – **pool**: Name of the pool to which the co-hosted recursors should be added (default: *default*)

  – **replicas**: Number of recursor containers which should be added to each dnsdist pod. This must be set to >= 1 for any recursors to be added (default: *0*)

  – All options configurable on standalone Recursor instances can also be added here, except for: affinity, antiAffinity, hostNetwork, nodeSelector & reivisionHistoryLimit (as those are pod-specific and those are controlled by dnsdist for this pod)

- **ringBuffers** (Ringbuffer configuration)

  – **retries**: Number of shards to attempt to lock without blocking before giving up and simply blocking while waiting for the next shard to be available. Defaults to 5 if there is more than 1 shard, else it defaults to 0. (Dnsdist default)

  – **shards**: Number of shards to use to limit lock contention, defaults to 1. (Dnsdist default)

  – **size**: Maximum amount of queries to keep in the ringbuffer, defaults to 10000. (Dnsdist default)

- **securityPolling** (Security polling configuration)

  – **securityPollInterval**: Interval between security pollings, in seconds. Defaults to 3600. (Dnsdist default)

  – **securityPollSuffix**: Domain name from which to query security update notifications. Setting this to an empty string disables secpoll. (Dnsdist default)

- **service** (Configuration of the Kubernetes service object **Note**: See the section 'Exposing dnsdist' for more details and examples on how to configure this)

  – **sharedService**: Whether or not to create a single service with multiple ports (ie: UDP & TCP). Default: *true*

  – **servicePerPort**: Whtehr or not to create a service for each port (ie: a service for UDP & a service for TCP). Default: *false*

  – **type**: Type of service (One of NodePort, ClusterIP, LoadBalancer). Defaults to ClusterIP

  – **annotations**: List of annotations to add to the Service

  – **loadBalancerIP**: For a Service of type LoadBalancer, a loadbalancer IP can be requested here. If left empty/undefined, this will be assigned by your LoadBalancer provider

  – **ports**: A dictionary of ports you want to have exposed via the Service

- **tuning** (dnsdist tuning functions)

  – **setCacheCleaningDelay**: Interval in seconds between two runs of the cache cleaning algorithm, removing expired entries. (Dnsdist default)

- **setCacheCleaningPercentage**: Percentage of the cache that the cache cleaning algorithm will try to free by removing expired entries. By default (100), all expired entries are removed. (Dnsdist default)

- **setMaxTCPClientThreads**: Maximum amount of TCP client threads, handling TCP connections. By default this value is 10, unless more than 10 TCP listen sockets have been defined. (Dnsdist default)

- **setMaxTCPConnectionDuration**: Maximum duration of an incoming TCP connection, in seconds. 0 (the default) means unlimited. (Dnsdist default)

- **setMaxTCPConnectionsPerClient**: Maximum number of TCP connections per client. 0 (the default) means unlimited. (Dnsdist default)

- **setMaxTCPQueriesPerConnection**: Maximum number of queries in an incoming TCP connection. 0 (the default) means unlimited. (Dnsdist default)

- **setMaxTCPQueuedConnections**: Maximum number of TCP connections queued (waiting to be picked up by a client thread), defaults to 1000. 0 means unlimited. (Dnsdist default)

- **setMaxUDPOutstanding**: Maximum number of outstanding UDP queries to a given backend server, defaults to 65535. (Dnsdist default)

- **setStaleCacheEntriesTTL**: TTL for expired cache entries to be eligible as answer when no backends are available for a query, in seconds. (Dnsdist default)

- **setTCPRecvTimeout**: Read timeout on TCP connections from the client, in seconds. (Dnsdist default)

- **setTCPSendTimeout**: Write timeout on TCP connections from the client, in seconds. (Dnsdist default)

- **setUDPTimeout**: Maximum time dnsdist will wait for a response from a backend over UDP, in seconds. Defaults to 2. (Dnsdist default)

11.2.3 Server Pools

pools (Configuration of pools)

Key: name (Name of the pool)

- **ecs**: Set to true if dnsdist should add EDNS Client Subnet information to the query before looking up into the cache, when all servers from this pool are down. If at least one server is up, the preference of the selected server is used, this parameter is only useful if all the backends in this pool are down and have EDNS Client Subnet enabled, since the queries in the cache will have been inserted with ECS information. Default is false. (Dnsdist default)

- **serverGroups**: List of [Name] of recursor & resolver instances whose endpoints will be member of this pool (Dnsdist default)

- **serverPolicy**: Policy for dnsdist to use to select the server in this pool to send a query to. Supported policies are 'leastOutstanding', 'firstAvailable', 'wrandom', 'whashed', 'chashed' & 'roundrobin' (Dnsdist default)

- **packetcache** (Configurable cache that holds responses from prior requests served by the pool)

– **cookieHashing**: Whether EDNS Cookie values will be hashed, resulting in separate entries for different cookies in the packet cache. This is required if the backend is sending answers with EDNS Cookies, otherwise a client might receive an answer with the wrong cookie. (Dnsdist default)

– **deferrableInsertLock**: Whether the cache should give up insertion if the lock is held by another thread, or simply wait to get the lock. (Dnsdist default)

– **keepStaleData**: Whether to suspend the removal of expired entries from the cache when there is no backend available in at least one of the pools using this cache. (Dnsdist default)

– **maxEntries**: Max (Dnsdist default)

– **maxNegativeTTL**: Cache a NXDomain or NoData answer from the backend for at most this amount of seconds, even if the TTL of the SOA record is higher. (Dnsdist default)

– **maxTTL**: Cap the TTL for records to his number. (Dnsdist default)

– **minTTL**: Do not cache entries with a TTL lower than this. (Dnsdist default)

– **numberOfShards**: Number of shards to divide the cache into, to reduce lock contention. (Dnsdist default)

– **parseECS**: Whether any EDNS Client Subnet option present in the query should be extracted and stored to be able to detect hash collisions involving queries with the same qname, qtype and qclass but a different incoming ECS value. (Dnsdist default)

– **staleTTL**: When the backend servers are not reachable, and global configuration setStaleCacheEntriesTTL is set appropriately, TTL that will be used when a stale cache entry is returned. (Dnsdist default)

– **temporaryFailureTTL**: On a SERVFAIL or REFUSED from the backend, cache for this amount of seconds. (Dnsdist default)

11.2.4 DNS over HTTP(S)

doh (List of configurations of DoH listeners)

• **name**: Name of the DoH listener (used for naming the corresponding Service objects, etc)

• **locals**: Amount of listen sockets per dnsdist pod (default: 1)

• **headers**: Dictionary of HeaderName:HeaderValue pairs to add to each response (Omitted on redirect responses)

• **secrets**: List of names of Secrets (Type: kubernetes.io/tls) containing certificates & keys to be used on the DoH listener. Key must be in *tls.key* node and certificate (+ intermediates if applicable) must be in *tls.crt* node.

• **service**: Configuration of the Kubernetes service object (Note: See the section 'Exposing dnsdist' for more details and examples on how to configure this)

• **stekSecret**: Name of a Secret (Type: Opaque) with stek tickets included in the *tickets* node. This secret is monitored for changes and will be updated in dnsdist whenever a change is detected. Default: CloudControl will generate and update tickets as per the defaults (See parameters named stek* in the dnsdist defaults section)

- **urls**: List of URLs to respond to (Default: /dns-query)

- **config** (Configuration options for the DoH listener)

  – **ciphers**: The TLS ciphers to use, in OpenSSL format. Ciphers for TLS 1.3 must be specified via ciphersTLS13.

  – **ciphersTLS13**: The TLS ciphers to use for TLS 1.3, in OpenSSL format.

  – **enableRenegotiation**: Whether secure TLS renegotiation should be enabled. Default: false (Disabled by default since it increases the attack surface and is seldom used for DNS)

  – **exactPathMatching**: Whether to do exact path matching of the query path against the paths configured in urls (true) or to accepts sub-paths (false). Default: true

  – **idleTimeout**: Set the idle timeout, in seconds. (Default: 30)

  – **internalPipeBufferSize**: Set the size in bytes of the internal buffer of the pipes used internally to pass queries and responses between threads. Default: 1048576

  – **maxConcurrentTCPConnections**: Maximum number of concurrent incoming TCP connections. Default: 0 (which means unlimited).

  – **minTLSVersion**: Minimum version of the TLS protocol to support. Possible values are *tls1.0 tls1.1*, *tls1.2* and *tls1.3*. Default is to require at least TLS 1.0.

  – **numberOfStoredSessions**: The maximum number of sessions kept in memory at the same time. Default is 20480. Setting this value to 0 disables stored session entirely.

  – **preferServerCiphers**: Whether to prefer the order of ciphers set by the server instead of the one set by the client. Default is true, meaning that the order of the server is used.

  – **releaseBuffers**: Whether OpenSSL should release its I/O buffers when a connection goes idle, saving roughly 35 kB of memory per connection. Default: true

  – **sendCacheControlHeaders**: Whether to parse the response to find the lowest TTL and set a HTTP Cache-Control header accordingly. Default is true.

  – **serverTokens**: The content of the Server: HTTP header returned by dnsdist. Default: *h2o/dnsdist*

  – **sessionTimeout**: Set the TLS session lifetime in seconds, this is used both for TLS ticket lifetime and for sessions kept in memory.

  – **sessionTickets**: Whether session resumption via session tickets is enabled. Default: true (ie: tickets are enabled)

  – **tcpFastOpenQueueSize**: Set the TCP Fast Open queue size, enabling TCP Fast Open when available and the value is larger than 0.

  – **tcpListenQueueSize**: Set the size of the listen queue.

  – **trustForwardedForHeader**: Whether to parse any existing X-Forwarded-For header in the HTTP query and use the right-most value as the client source address and port, for ACL checks, rules, logging and so on. Default is false.

- **responses** (Responses to URLs in *urls* which should not be a DNS answer, but a custom HTTP response)

- **regex**: Regular expression to match the path against

- **status**: HTTP code to answer with

- **content**: Content of the HTTP response (or a URL in case of a redirect - HTTP-3XX)

- **headers**: Dictionary of HeaderName:HeaderValue pairs to add to each response (Omitted on redirect responses)

• **certificates** (List of certificates to be used on the DoH listener)

- **cert**: Certificate in PEM format, including intermediates if applicable

- **key**: Private key corresponding to the certificate

11.2.5 DNS over TLS

doh (List of configurations of DoT listeners)

• **name**: Name of the DoT listener (used for naming the corresponding Service objects, etc)

• **locals**: Amount of listen sockets per dnsdist pod (default: 1)

• **secrets**: List of names of Secrets (Type: kubernetes.io/tls) containing certificates & keys to be used on the DoT listener. Key must be in *tls.key* node and certificate (+ intermediates if applicable) must be in *tls.crt* node.

• **service**: Configuration of the Kubernetes service object (Note: See the section 'Exposing dnsdist' for more details and examples on how to configure this)

• **stekSecret**: Name of a Secret (Type: Opaque) with stek tickets included in the *tickets* node. This secret is monitored for changes and will be updated in dnsdist whenever a change is detected. Default: CloudControl will generate and update tickets as per the defaults (See parameters named stek* in the dnsdist defaults section)

• **config** (Configuration options for the DoT listener)

- **ciphers**: The TLS ciphers to use, in OpenSSL format. Ciphers for TLS 1.3 must be specified via ciphersTLS13.

- **ciphersTLS13**: The TLS ciphers to use for TLS 1.3, in OpenSSL format.

- **enableRenegotiation**: Whether secure TLS renegotiation should be enabled. Default: false (Disabled by default since it increases the attack surface and is seldom used for DNS)

- **maxConcurrentTCPConnections**: Maximum number of concurrent incoming TCP connections. Default: 0 (which means unlimited).

- **maxInFlight**: Maximum number of in-flight queries. The default is 0, which disables out-of-order processing.

- **minTLSVersion**: Minimum version of the TLS protocol to support. Possible values are *tls1.0 tls1.1*, *tls1.2* and *tls1.3*. Default is to require at least TLS 1.0.

- **numberOfStoredSessions**: The maximum number of sessions kept in memory at the same time. Default is 20480. Setting this value to 0 disables stored session entirely.

– **preferServerCiphers**: Whether to prefer the order of ciphers set by the server instead of the one set by the client. Default is true, meaning that the order of the server is used.

– **releaseBuffers**: Whether OpenSSL should release its I/O buffers when a connection goes idle, saving roughly 35 kB of memory per connection. Default: true

– **sessionTimeout**: Set the TLS session lifetime in seconds, this is used both for TLS ticket lifetime and for sessions kept in memory.

– **sessionTickets**: Whether session resumption via session tickets is enabled. Default: true (ie: tickets are enabled)

– **tcpFastOpenQueueSize**: Set the TCP Fast Open queue size, enabling TCP Fast Open when available and the value is larger than 0.

– **tcpListenQueueSize**: Set the size of the listen queue.

- **certificates** (List of certificates to be used on the DoT listener)

– **cert**: Certificate in PEM format, including intermediates if applicable

– **key**: Private key corresponding to the certificate

11.3 recursors

Configuration of recursor instances

Key: name (Name of the recursor instance)

11.3.1 Parameters

- **affinity**: Kubernetes pod affinity

- **agentLogLevel**: Agent log level, defaults to *info*. Available options: *debug*, *info*, *warn*, *error*

- **agentLogFormat**: Agent log format, defaults to *text*. Available options: *text*, *json*

- **agentResources**: Resources allocated to the 'recursor-agent' container

- **antiAffinityPreset**: Pod anti affinity preset (Accepted values: *preferred* or *required*)

- **apiKey**: API key used to access the /api endpoint, used to configure a static key (default: generated and stored in a secret)

- **apiKeySecret**: Name of the secret from which to take the API key, used when there is a pre-provisioned secret holding the api key

- **apiKeySecretItem**: Name of the data item within *apiKeySecret* which holds the API key, used when there is a pre-provisioned secret holding the api key (default: *api_key*)

- **containerSecurityContext**: Container security context for all containers within a Pod (see 'Security contexts' chapter)

- **hostNetwork**: Use host networking for recursor pods

- **inboundInterfaces**: List of names of interfaces to which recursor will bind

- **initResources**: Resources allocated to the 'recursor-init' container

- **luaConfig**: Lua configuration to be included in each recursor pod (See 'Overview' document for usage examples)

- **luaScript**: Lua script to be included in each recursor pod (See 'Overview' document for usage examples)

- **metricsInterfaces**: List of names of interfaces to which the metrics aggregator will bind

- **nodeSelector**: Kubernetes pod nodeSelector

- **outboundInterfaces**: List of names of interfaces from which recursor will try to send outbound traffic

- **podAnnotations**: Annotations to be added to each Pod

- **podLabels**: Labels to be added to each Pod

- **podSecurityContext**: Kubernetes Pod security context (see 'Security contexts' chapter)

- **replicas**: Default number of replicas in a recursor deployment (default: *2*)

- **resources**: Resources allocated to the 'recursor' container

- **revisionHistoryLimit**: Default 'revisionHistoryLimit' for recursor deployments (default: *0*)

- **tolerations**: Kubernetes Pod tolerations

- **volumes**: List of additional volumes (see *Storage* chapter)

11.3.2 Parameter Sets

- **dnsdist** (Settings to be applied to each recursor instance when added to dnsdist as a server)

  – **addXPF**: Add the client's IP address and port to the query, along with the original destination address and port. Default is disabled (0)

  – **checkClass**: Number to use as QCLASS in the health-check query, default is DNSClass.IN

  – **checkInterval**: The time in seconds between health checks

  – **checkName**: String to use as QNAME in the health-check query, default is "a.root-servers.net."

  – **checkTimeout**: The timeout (in milliseconds) of a health-check query, default to 1000 (1s)

  – **checkType**: String to use as QTYPE in the health-check query, default is "A"

  – **disableZeroScope**: Disable the EDNS Client Subnet 'zero scope' feature, which does a cache lookup for an answer valid for all subnets (ECS scope of 0) before adding ECS information to the query and doing the regular lookup. This requires the *parseECS* option of the corresponding cache to be set to true

  – **healthCheckMode**: Type of health-check to perform, default is *auto* which is configured using the checkName, checkType, etc parameters. Alternatives are *up* (no healthcheck - always available for traffic) and *down* (no healthcheck - never available for traffic)

- **maxCheckFailures**: Allow this amount of check failures before declaring the backend down, default is 1

- **mustResolve**: Set to true when the health check MUST return a RCODE different from NXDomain, ServFail and Refused. Default is false, meaning that every RCODE except ServFail is considered valid

- **order**: The order of servers in this set, used by the *leastOutstanding* and *firstAvailable* policies

- **qps**: Limit the number of queries per second to this amount, when using the *firstAvailable* policy

- **reconnectOnUp**: Close and reopen the sockets when a server transits from Down to Up. This helps when an interface is missing when dnsdist is started. Default is false

- **retries**: The number of TCP connection attempts to servers in this set, for a given query

- **rise**: Require NUM consecutive successful checks before declaring the backend up, default is 1

- **setCD**: Set the CD (Checking Disabled) flag in the health-check query, default is false

- **sockets**: Number of sockets (and thus source ports) used toward the backend server, defaults to 1

- **source**: Name of the interface which Dnsdist will use to try to send traffic to this Recursor

- **tcpConnectTimeout**: The timeout (in seconds) of a TCP connection attempt

- **tcpFastOpen**: Whether to enable TCP Fast Open

- **tcpRecvTimeout**: The timeout (in seconds) of a TCP read attempt

- **tcpSendTimeout**: The timeout (in seconds) of a TCP write attempt

- **useClientSubnet**: Add the client's IP address in the EDNS Client Subnet option when forwarding the query to this backend

- **useProxyProtocol**: Add a proxy protocol header to the query, passing along the client's IP address and port along with the original destination address and port. Default is false

- **weight**: The weight of servers in this set, used by the *wrandom*, *whashed* and *chashed* policies, default is 1

- **config** (Recursor configuration, any configuration item listed in the recursor documentation (https://doc.powerdns.com/recursor/settings.html) can be referenced here. Settings which are explicitly ignored due to conflicts with Cloud Control are filtered by 'webserver*', 'local*', 'disable-syslog', 'daemon', 'cpu-map', 'chroot', 'socket*' and 'config*')

- **hpa** (Horizontal Pod Autoscaler - see helm values 'recursor.hpa' for examples)

  - **enabled**: Enable or disable the HPA (default: *false*)

  - **minReplicas**: Minimum # of replicas (default: *2*)

  - **maxReplicas**: Maximum # of replicas (default: *4*)

- **metrics**: Metric(s) upon which to make decisions regarding scaling. Must be an array of MetricSpec (`https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#metricspec-v2beta2-autoscaling`). Examples are in the helm values.

- **behavior**: Behavior of the HPA, must be instance of HorizontalPodAutoscalerBehavior (`https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#horizontalpodautoscalerspec-v2beta2-autoscaling`).

- **readiness** (Readiness probe Configuration)

  - **bindInterfaces**: List of names of interfaces to which the readiness probe will bind

11.3.3 Forward Zones

forward (List of sets of zones to be forwarded to Auth instances, Recursor instances or other resolvers)

- **exclude**: If *learnFrom* is configured, exclude any zones which match this list of regular expressions (this is processed after the *include* filter)

- **include**: If *learnFrom* is configured, include only zones which match this list of regular expressions (this is processed before the *exclude* filter)

- **learnFrom**: Name of a set of Auth instances from which forward zones must be learned (can be filtered using *include* and/or *exclude*)

- **nta**: Add a negative trust anchor for the forwarded zones (default: *false*)

- **priority**: Priority when the list of sets of forward zones has multiple entries containing duplicate static and/or learned zones, lowest value for priority has precedence (default: *100*)

- **recurse**: Set the recursion desired (RD) bit to 1 for the forwarded zones (default: *false*)

- **zones**: List of zones to be forwarded (not affected by *include* and *exclude*)

- **serverGroups**: (List of server groups to forward to - Auth, Recursor or Resolver instances)

  - **group**: Name of the group of instances

11.4 resolvers

Configuration of external resolvers

Key: name (Name of the resolver instance)

11.4.1 Parameters

- **ips**: List of IP addresses of resolver endpoints

- **port**: Port to send traffic to for resolver endpoints

11.4.2 Parameter Sets

- **dnsdist** (Settings to be applied to each resolver instance when added to dnsdist as a server)

  - **addXPF**: Add the client's IP address and port to the query, along with the original destination address and port. Default is disabled (0)

  - **checkClass**: Number to use as QCLASS in the health-check query, default is DNSClass.IN

  - **checkInterval**: The time in seconds between health checks

  - **checkName**: String to use as QNAME in the health-check query, default is "a.root-servers.net."

  - **checkTimeout**: The timeout (in milliseconds) of a health-check query, default to 1000 (1s)

  - **checkType**: String to use as QTYPE in the health-check query, default is "A"

  - **disableZeroScope**: Disable the EDNS Client Subnet 'zero scope' feature, which does a cache lookup for an answer valid for all subnets (ECS scope of 0) before adding ECS information to the query and doing the regular lookup. This requires the *parseECS* option of the corresponding cache to be set to true

  - **healthCheckMode**: Type of health-check to perform, default is *auto* which is configured using the checkName, checkType, etc parameters. Alternatives are *up* (no healthcheck - always available for traffic) and *down* (no healthcheck - never available for traffic)

  - **maxCheckFailures**: Allow this amount of check failures before declaring the backend down, default is 1

  - **mustResolve**: Set to true when the health check MUST return a RCODE different from NXDomain, ServFail and Refused. Default is false, meaning that every RCODE except ServFail is considered valid

  - **order**: The order of servers in this set, used by the *leastOutstanding* and *firstAvailable* policies

  - **qps**: Limit the number of queries per second to this amount, when using the *firstAvailable* policy

  - **reconnectOnUp**: Close and reopen the sockets when a server transits from Down to Up. This helps when an interface is missing when dnsdist is started. Default is false

  - **retries**: The number of TCP connection attempts to servers in this set, for a given query

  - **rise**: Require NUM consecutive successful checks before declaring the backend up, default is 1

- **setCD**: Set the CD (Checking Disabled) flag in the health-check query, default is false

- **sockets**: Number of sockets (and thus source ports) used toward the backend server, defaults to 1

- **source**: Name of the interface which Dnsdist will use to try to send traffic to this external resolver

- **tcpConnectTimeout**: The timeout (in seconds) of a TCP connection attempt

- **tcpFastOpen**: Whether to enable TCP Fast Open

- **tcpRecvTimeout**: The timeout (in seconds) of a TCP read attempt

- **tcpSendTimeout**: The timeout (in seconds) of a TCP write attempt

- **useClientSubnet**: Add the client's IP address in the EDNS Client Subnet option when forwarding the query to this backend

- **useProxyProtocol**: Add a proxy protocol header to the query, passing along the client's IP address and port along with the original destination address and port. Default is false

- **weight**: The weight of servers in this set, used by the *wrandom*, *whashed* and *chashed* policies, default is 1

11.5 rulesets

Rulesets allow for the configuration of dnsdist Packet Policies. For a detailed example of how to use these rulesets, please refer to the *Getting Started* chapter in the *Overview* guide.

A basic ruleset looks as follows:

```
rulesets:
  tcp-refusal-ruleset:
    group: block-traffic
    type: DNSDistRule
    priority: 100
    rules:
    - name: Refuse TCP
      combinator: AND
      selectors:
        - TCP: true
        - QName: "tcptrue.example.com"
      action:
        RCode:
          rcode: "REFUSED"
```

Where 'tcp-refusal-ruleset' is the name of the ruleset, which will be used to create a uniquely named object. Under this name, the ruleset is defined, using the following values:

- **group**: The value of this parameter can be referenced in a dnsdist configuration to apply the rules from this ruleset to that instance

- **type**: Type of ruleset, currently limited to 'DNSDistRule'

- **priority**: Priority of this ruleset. If multiple are assigned to a dnsdist instance, it will process the rule with the lowest 'priority' value first.

- **rules**: An array of rules to be applied to this ruleset (see below for more details on rules)

11.5.1 Rules

Rules are configurable via 4 different parameters:

- **name**: Name of the rule
- **combinator**: One of 'AND', 'OR' or 'NOT'
- **selectors**: List of filters on which to apply the logic of the combinator
- **action**: Action to apply to the traffic selected by the above selectors

As a result, a single rule will look as follows:

```
name: Refuse TCP
combinator: AND
selectors:
  - TCP: true
  - QName: "tcptrue.example.com"
action:
  RCode:
    rcode: "REFUSED"
```

11.5.2 Combinators

The available combinators function as follows:

- **AND**: If all selectors match, the action will be applied
- **OR**: If any of the selectors match, the action will be applied
- **NOT**: If the selector does not match, the action will be applied (Only 1 selector is allowed when using a NOT combinator)

11.5.3 Selectors

The following selectors are available:

11.5.3.1 TCP

Format:

```
TCP: true
```

If 'true', this will select queries received over TCP. If 'false' it will select non-TCP traffic (ie: UDP)

11.5.3.2 MaxQPS

Format:

```
MaxQPS:
  qps: 50
```

Matches traffic not exceeding this qps limit. If e.g. this is set to 50, starting at the 51st query of the current second traffic stops being matched. This can be used to enforce a global QPS limit.

11.5.3.3 MaxQPSIP

Format:

```
MaxQPSIP:
  qps: 20
  v4Mask: 32
  v6Mask: 64
  burst: 20
  expiration: 300
  cleanupDelay: 60
  scanFraction: 10
```

Explanation of each parameter:

- qps (int) – The number of queries per second allowed, above this number traffic is matched

- v4Mask (int) – The IPv4 netmask to match on. Default is 32 (the whole address)

- v6Mask (int) – The IPv6 netmask to match on. Default is 64

- burst (int) – The number of burstable queries per second allowed. Default is same as qps

- expiration (int) – How long to keep netmask or IP addresses after they have last been seen, in seconds. Default is 300

- cleanupDelay (int) – The number of seconds between two cleanups. Default is 60

- scanFraction (int) – The maximum fraction of the store to scan for expired entries, for example 5 would scan at most 20% of it. Default is 10 so 10%

Since most of the parameters have defaults, you can define a basic MaxQPSIP selector as follows:

```
MaxQPSIP:
  qps: 20
```

11.5.3.4 NetmaskGroup

Format:

```
NetmaskGroup:
  nmg:
  - "192.0.2.0/28"
  - "2001:db8:1234::/64"
  src: true
  quiet: false
```

Explanation of each parameter:

- nmg (NetMaskGroup) – The NetMaskGroups to match on

- src (bool) – Whether to match source or destination address of the packet. Defaults to true (matches source)

- quiet (bool) – Do not display the list of matched netmasks in Rules. Default is false.

Since most of the parameters have defaults, you can define a basic NetmaskGroup selector as follows:

```
NetmaskGroup:
  nmg:
  - "192.0.2.0/28"
  - "2001:db8:1234::/64"
```

11.5.3.5 Opcode

Format:

```
Opcode: "Notify"
```

This selector matches queries with the specified opcode exactly. An example usecase for this selector is to route zone update notification queries to secondary Auth instances, ie:

```
name: authnotify
combinator: AND
selectors:
  - Opcode: "Notify"
action:
  Pool:
    poolname: "auth"
```

11.5.3.6 QName

Format:

```
QName: "host.example.com"
```

This selector matches queries with the specified qname exactly.

11.5.3.7 QType

Format:

```
QType: "SOA"
```

This selector matches queries with the specified qtype exactly. An example usecase for this selector is to route zone transfer related queries to Auth instances, ie:

```
name: authxfr
combinator: OR
selectors:
  - QType: "SOA"
  - QType: "AXFR"
  - QType: "IXFR"
action:
  Pool:
    poolname: "auth"
```

11.5.4 Actions

The following actions are available:

11.5.4.1 Allow

Format:

```
Allow: true
```

Let these packets go through.

11.5.4.2 Drop

Format:

```
Drop: true
```

Drop the packet.

11.5.4.3 Pool

Format:

```
Pool:
  poolname: "name-of-a-pool"
```

The Pool action will send the packet into the specified pool

11.5.4.4 QPS

Format:

```
QPS:
  maxqps: 10
```

Drop a packet if it does exceed the maxqps queries per second limits.

11.5.4.5 RCode

Format:

```
RCode:
  rcode: "REFUSED"
```

The RCode action will answer any selector queries with the rcode specified.

11.5.4.6 TC

Format:

```
TC: true
```

Create answer to query with the TC bit set, and the RA bit set to the value of RD in the query, to force the client to TCP.

11.6 zonecontrols

Configuration of zonecontrol instances

Key: name (Name of the zonecontrol instance)

11.6.1 Parameters

- **adminUser**: Username of user to create with administrative privileges (Default: *zcadmin*)
- **adminPassword**: Password of the user specified by *adminUser* (Default: generated)
- **adminSecret**: Name of an existing Secret holding the desired username and password of a user to be provisioned with administrative privileges
- **adminSecretUserKey**: Name of the item within *adminSecret* which holds the username of the admin user (Default: *admin_user*)
- **adminSecretPasswordKey**: Name of the item within *adminSecret* which holds the password of the admin user (Default: *admin_password*)
- **affinity**: Kubernetes pod affinity
- **antiAffinityPreset**: Pod anti affinity preset (Accepted values: *preferred* or *required*)
- **containerSecurityContext**: Container security context for all containers within a Pod (see 'Security contexts' chapter)

- **hostNetwork**: Use host networking for zonecontrol pods

- **ingress**: Ingress object to expose the zonecontrol endpoint via an ingress managed by an ingress controller (For formatting example, see the *Exposing auth API* chapter)

- **initResources**: Resources allocated to the 'zc-init' container

- **nginxResources**: Resources allocated to the 'nginx' container

- **nodeSelector**: Kubernetes pod nodeSelector

- **podSecurityContext**: Kubernetes Pod security context (see 'Security contexts' chapter)

- **postgres**: Postgres database configuration (See: *Postgres Database* chapter below for details)

- **replicas**: Default number of replicas in a auth deployment (default: *2*)

- **resources**: Resources allocated to the 'zonecontrol' container

- **service**: Service object to expose the zonecontrol endpoint (For formatting example, see the *Exposing dnsdist* chapter, which covers the *service* format to add ClusterIP, NodePort or LoadBalancer services)

- **tolerations**: Kubernetes Pod tolerations

11.6.2 Endpoints

There are 3 ways to configure endpoints of Authoritative instances in ZoneControl:

- Static URL + Static API key

- Static URL + API key from Secret

- Learned from a set of Auth instances (within the same namespace)

11.6.2.1 Static URL + Static API key

You can configure an Auth endpoint with static URL and API key as follows:

```
zonecontrols:
  myzc:
    replicas: 2
    postgres:
      <database config here>
    authEndpoints:
      - name: ls1
        url: http://url.of.auth:8080
        key: "SomeAPIKey"
```

Explanation of each parameter:

- name - Required, name of the endpoint in the ZoneControl GUI

- url - Required, URL to connect to the Auth endpoint

- key - Required, API Key for the endpoint

11.6.2.2 Static URL + API key from Secret

You can configure an Auth endpoint with static URL and Secret holding the API key as follows:

```
zonecontrols:
  myzc:
    replicas: 2
    postgres:
      <database config here>
    authEndpoints:
      - name: ls1
        url: http://url.of.auth:8080
        keySecret: auth-testauth
        keySecretItem: "api_key"
```

Explanation of each parameter:

- name - Required, name of the endpoint in the ZoneControl GUI

- url - Required, URL to connect to the Auth endpoint

- keySecret - Required, name of the secret which holds the API key

- keySecretItem - Optional, name of the item inside the Secret which holds the API key (Default: *api_key*)

11.6.2.3 Learned from a set of Auth instances

You can configure an Auth endpoint learned from a set of Auth instances (within the same namespace) as follows:

```
zonecontrols:
  myzc:
    replicas: 2
    postgres:
      <database config here>
    authEndpoints:
      - name: ls1
        auth: testauth
```

Explanation of each parameter:

- name - Required, name of the endpoint in the ZoneControl GUI

- auth - Required, name of the set of Auth instances from which to learn the endpoint and API key

11.6.3 Postgres Database

ZoneControl requires a Postgres database to run. You can either specifiy the details of an existing database or have the bundled Postgres Operator create one. Both options are described in further detail below.

11.6.3.1 Postgres (pre-existing)

If you have an existing Postgres database available, you can configure a zonecontrol instance to utilize it. Configuring a pre-existing Postgres database can be done via the following configuration:

Format:

```
postgres:
  host: host-of-postgres-cluster
  dbname: auth
  user: some_user
  password: some_user_password
```

Parameters - database endpoint:

- host - Host of the Postgres cluster/service

- port - Port over which to access the Postgres database (Optional, default: *5432*)

- dbname - Name of the database

- sslmode - Postgres *sslmode* to use while connecting (Optional, omitted if no value is provided)

Parameters - database credentials:

You have to provide credentials to access the database, this can be done in 2 ways:

Inline username and pasword:

- user - Required, user with which to connect

- password - Required, password for *user*

Username and password from a pre-existing secret:

- secretName - Required, name of existing secret containing database credentials

- secretUsernameKey - Optional, name of the item inside the Secret which holds the username (Default: *username*)

- secretPasswordKey - Optional, name of the item inside the Secret which holds the password (Default: *password*)

11.6.3.2 Postgres (Operator-managed)

If you have deployed the postgres-operator chart to your Kubernetes cluster, you can configure a zonecontrol instance to use a operator-managed Postgres database. Configuring such a Postgres database can be done via the following configuration:

Format:

```
postgres:
  operator: true
```

Available parameters:

- operator - Must be set to *true*

The Postgres database which is created utilizes the default values which are exposed via the helm values in *zonecontrol.postgres*.

12 Prometheus

This section allows you to configure 2 forms of Prometheus scraping configurations:

- Via PodMonitor objects (if a Prometheus Operator) is available

- Via Prometheus scraping annotations on Pods

By default, all Pods will have the following annotations present for Prometheus scraping:

- prometheus.io/path = the path where the metrics endpoint serves metrics (default: */metrics*)

- prometheus.io/port = the port to which the metrics endpoint listens (default: *8082*)

- prometheus.io/scheme = the scheme for serving metrics (default: *http*)

- prometheus.io/scrape = whether or not to scrape this endpoint (default: *true*)

To disable this default behavior, you can set the value *prometheus.annotations* to *false* (default: *true*)

If you have a Prometheus Operator available or if you deployed the stack including the *Monitoring Operators* chart, you can set the value *prometheus.operator.available* to *true* (default: *false*). As a result PodMonitor objects will be created for each dnsdist & recursor instance, with all the necessary scraping details automatically included. These pods should then all be discovered automatically by the Prometheus Operator.