Nodelocal DNS Cache

Using NodeLocal DNSCache in Kubernetes clusters(https://kubernetes.io/docs/tasks/administer-cluster/nodelocaldns/). This addon runs a node-local-dns pod on all cluster nodes. The pod runs CoreDNS as the dns cache. It runs with `hostNetwork:True` and creates a dedicated dummy interface with a link local ip(169.254.20.10/32 by default) to listen for DNS queries. The cache instances connect to clusterDNS in case of cache misses.

Design details here

This feature is graduating to GA in release 1.18(Beta in release 1.15).

nodelocaldns addon template

This directory contains the addon config yaml - `nodelocaldns.yaml` The variables will be substituted by the configure scripts when the yaml is copied into master.

We have the following variables in the yaml: `__PILLAR__DNS__SERVER__` - set to kube-dns service IP. `__PILLAR__LOCAL__DNS__` - set to the link-local IP(169.254.20.10 by default). `__PILLAR__DNS__DOMAIN__` - set to the cluster domain(cluster.local by default).

Note: The local listen IP address for NodeLocal DNSCache can be any address that can be guaranteed to not collide with any existing IP in your cluster. It's recommended to use an address with a local scope, per example, from the link-local range 169.254.0.0/16 for IPv4 or from the Unique Local Address range in IPv6 fd00::/8.

The following variables will be set by the node-cache images - k8s.gcr.io/k8s-dns-node-cache:1.15.6 or later. The values will be determined by reading the kube-dns configMap for custom Upstream server configuration. `__PILLAR__CLUSTER__DNS__` - Upstream server for in-cluster queries. `__PILLAR__UPSTREAM__SERVERS__` - Upstream servers for external queries.

Network policy and DNS connectivity

When running nodelocaldns addon on clusters using network policy, additional rules might be required to enable dns connectivity. Using a namespace selector for dns egress traffic as shown here might not be enough since the node-local-dns pods run with `hostNetwork: True`

One way to enable connectivity from node-local-dns pods to clusterDNS ip is to use an ipBlock rule instead:

```
spec:
  egress:
  - ports:
    - port: 53
```

```
      protocol: TCP
    - port: 53
      protocol: UDP
    to:
    - ipBlock:
        cidr: <well-known clusterIP for DNS>/32
  podSelector: {}
  policyTypes:
  - Ingress
  - Egress
```

Negative caching

The `denial` cache TTL has been reduced to the minimum of 5 seconds here. In the unlikely event that this impacts performance, setting this TTL to a higher value make help alleviate issues, but be aware that operations that rely on DNS polling for orchestration may fail (for example operators with StatefulSets).