

Kemp Ingress Controller for Kubernetes

Feature Description

UPDATED: 27 July 2023

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

These materials and all Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. The information in these materials is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear therein. The references in these materials to specific platforms supported are subject to change.

#1 Load Balancer in Price/Performance, 360 Central, 360 Vision, Chef, Chef (and design), Chef Habitat, Chef Infra, Code Can (and design), Compliance at Velocity, Corticon, Corticon.js, DataDirect (and design), DataDirect Cloud, DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, DataRPM, Defrag This, Deliver More Than Expected, DevReach (and design), Driving Network Visibility, Flowmon, Inspec, Ipswitch, iMacros, K (stylized), Kemp, Kemp (and design), Kendo UI, Kinvey, LoadMaster, MessageWay, MOVEit, NativeChat, OpenEdge, Powered by Chef, Powered by Progress, Progress, Progress Software Developers Network, SequeLink, Sitefinity (and Design), Sitefinity, Sitefinity (and design), Sitefinity Insight, SpeedScript, Stylized Design (Arrow/3D Box logo), Stylized Design (C Chef logo), Stylized Design of Samurai, TeamPulse, Telerik, Telerik (and design), Test Studio, WebSpeed, WhatsConfigured, WhatsConnected, WhatsUp, and WS_FTP are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries.

Analytics360, AppServer, BusinessEdge, Chef Automate, Chef Compliance, Chef Desktop, Chef Workstation, Corticon Rules, Data Access, DataDirect Autonomous REST Connector, DataDirect Spy, DevCraft, Fiddler, Fiddler Classic, Fiddler Everywhere, Fiddler Jam, FiddlerCap, FiddlerCore, FiddlerScript, Hybrid Data Pipeline, iMail, InstaRelinker, JustAssembly, JustDecompile, JustMock, KendoReact, OpenAccess, PASOE, Pro2, ProDataSet, Progress Results, Progress Software, ProVision, PSE Pro, Push Jobs, SafeSpaceVR, Sitefinity Cloud, Sitefinity CMS, Sitefinity Digital Experience Cloud, Sitefinity Feather, Sitefinity Thunder, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, Supermarket, SupportLink, Unite UX, and WebClient are trademarks or service marks of Progress Software Corporation and/or its subsidiaries or affiliates in the U.S. and other countries. Java is a registered trademark of Oracle and/or its affiliates. Any other marks contained herein may be trademarks of their respective owners.

Please refer to the NOTICE.txt or Release Notes – Third-Party Acknowledgements file applicable to a particular Progress product/hosted service offering release for any related required third-party acknowledgements.

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

Table of Contents

1 Introduction **5**

2 Prerequisites **6**

2.1 AWS EKS Setup Steps 6

3 Install the LoadMaster Ingress Controller **9**

4 Enable LoadMaster to Kubernetes Integration **11**

5 Service Mode and Ingress Mode **13**

5.1 Service Mode 14

5.1.1 How Service Mode Works 16

5.1.2 When to Use Service Mode 16

5.1.3 Configure Service Mode 17

5.1.3.1 Configure a LoadMaster Virtual Service 17

5.1.3.2 Configure the Kubernetes Service 17

5.1.3.3 Verify the Application 18

5.2 Ingress Mode 19

5.2.1 How Ingress Mode Works 23

5.2.2 When to Use Ingress Mode 23

5.2.3 Configure Ingress Mode 24

5.2.3.1 Create a Kubernetes Ingress Controller 24

5.2.3.2 Verify the Application 25

5.2.4 Additional Annotations for Ingress Mode 25

6 Ensuring Connectivity from the LoadMaster to Pods **29**

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

6.1 Overlay Networks 29

6.2 How to Check if an Overlay Network is in Use 29

6.3 LoadMaster to Pod Connectivity 30

6.4 Adding Routes 30

6.4.1 Calico 31

7 Creation and Deletion of Objects 33

8 LoadMaster Ingress Logs 35

Last Updated Date 36

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

1 Introduction

The Kemp Ingress Controller for Kubernetes enables a Kemp LoadMaster Virtual Service to be used to publish access to Kubernetes applications. This enables application server pools in published Virtual Services to dynamically update when Pods running on a Kubernetes platform are added or removed from a proxied service object. This provides a convenient way for managing the connection between the Load Balancer and Container network for connections into Kubernetes (commonly referred to as North-South Traffic).

As LoadMaster operates outside of Kubernetes, the Kemp Ingress Controller provides a convenient way for managing end points for either Containerized or Monolithic Applications or a combination of both through the same device.

You first need to install the Ingress Controller on the LoadMaster. Once installed, you can enable this functionality in Service or Ingress mode.

This functionality provides the following capabilities:

- Automated mapping of Kubernetes Service or Ingress object configurations to LoadMaster Virtual Services and SubVSs
- Support for mapping Kubernetes annotations to Virtual Service attributes
- Automated addition and deletion of Real Servers as a result of changes in Kubernetes (for example, scale up or scale down operations)

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

2 Prerequisites

Some prerequisites that must be in place before configuring the Kemp Ingress Controller functionality for Kubernetes are listed below:

- At least one licensed and running LoadMaster instance
- A Kube config file with the necessary permissions to read configuration using the Kubernetes Application Programming Interface (API)
- Network access between the LoadMaster and the Kubernetes API server
- Details of the namespace for which objects are defined
- The LoadMaster must be able to route to the Pod CIDR networks within the relevant Kubernetes Clusters and the Cluster Nodes. This may require additional routes to be created. For further details on how to do this, refer to the **Ensuring Connectivity from the LoadMaster to Pods** section.

The Kemp Ingress Controller is currently confirmed to be supported on the following platforms:

- Cloud:
    - Kubernetes running in Azure (Azure Kubernetes Service (AKS))
    - Kubernetes running in Amazon Web Services (AWS) Elastic Kubernetes Service (EKS). For setup steps that you must perform if using AWS EKS, refer to the **AWS EKS Setup Steps** section.
    - Kubernetes running in Rancher Kubernetes Management Platform
- On-prem Linux: Kubernetes running on Linux Operating System (OS)

2.1 AWS EKS Setup Steps

If you intend to use the Kemp Ingress Controller for Kubernetes in the AWS EKS platform, there are some setup steps that you must perform:

1. Install the AWS Command Line Interface (CLI) 2.0:

   curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"

   unzip awscliv2.zip

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

sudo ./aws/install

2. Install kubectl:

curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.18.9/2020-11-02/bin/linux/amd64/kubectl

3. Install eksctl:

curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp

sudo mv /tmp/eksctl /usr/local/bin

eksctl version

> The AWS user must have permissions to create an EKS cluster. There must also be available resources to create one Virtual Private Cloud (VPC).

4. Run the command to create the cluster:

eksctl create cluster --name=eksctl --region=us-east-1 --ssh-public-key=aws_eks_ted.pub --nodes=4 --node-type=m5.large

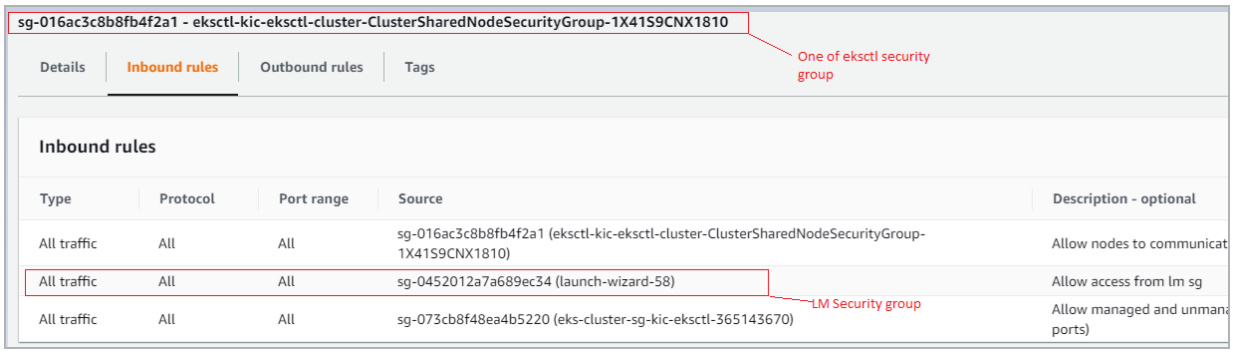
> The above command takes approximately 5 to 10 minutes to execute. Upon successful execution, it creates one VPC, four subnets, and three security groups.

5. After creating the cluster, create the LoadMaster on AWS with one public interface (subnet) of eksctl and a security group with **All** traffic allowed.

LoadMaster security group rules: For example, launch-wizard-58

Type	Protocol	Port range
All traffic	All	All

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.



6. Then, add the LoadMaster security group (created above) to all the security groups created by eksctl to ensure communication and to not be restricted by ports, protocols, and so on. The screenshot above shows an example of the eksctl security group.

7. License the LoadMaster, install the addons and upload the Kubeconfig file.

8. Create the configuration on eksctl and health checking should then work.

For further details, refer to the following AWS User Guide: Getting started with Amazon EKS - AWS Management Console and AWS CLI.

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

3 Install the LoadMaster Ingress Controller

The Ingress Controller is not installed on the LoadMaster by default. You can easily install it by following these steps:

1. In the LoadMaster User Interface (UI), go to **Virtual Services > Kubernetes Settings**.
2. Click **Install**.
3. Wait for the installation to complete and click **OK** on the confirmation message.
4. Reboot the LoadMaster to activate all required add-ons - **System Configuration > System Administration > System Reboot > Reboot**.

After rebooting, you can use the **Kubernetes Settings** configuration page to enable the LoadMaster Kubernetes integration.

> For LoadMasters deployed in AWS or Azure in a High Availability Pair, please ensure that the add-ons are installed on both devices.

> LoadMaster users (apart from the default admin **bal** user) must be assigned the **All Permissions** option in their user permissions to be permitted access to modify Kubernetes settings in the LoadMaster.

Manual add-on installation

If the LoadMaster is deployed to a location where connectivity to Kemp infrastructure is not available, you can install the Kemp Ingress Controller by following the below steps:

1. Download the required add-ons. Go to https://support.kemptechnologies.com/hc/en-us/sections/200409933-Other-Downloads and select the **Kemp Ingress Contoller**. Here you can download the required add-on files.

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

2. On LoadMaster, navigate to **System Configuration>System Administration>Update Software**.
3. Under the **Install new Addon Package** menu add each file and verification file and click **Install Addon Package**.
4. Reboot the LoadMaster to activate all required add-ons, navigate to **System Configuration > System Administration > System Reboot > Reboot**.

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

4 Enable LoadMaster to Kubernetes Integration

To enable LoadMaster to Kubernetes communication, the LoadMaster must have Kubernetes API access. To enable this, follow the steps below in the LoadMaster UI:

1. In the main menu, go to **Virtual Services > Kubernetes Settings**.
2. Click **Choose File**.
3. Browse to and select your **Kube Config** file.

   This allows the LoadMaster to communicate with Kubernetes.

   The default location for the Kube Config file is ~/.kube/config, for example, if you are using the Azure Cloud Shell you can access this using **/home/<YourName>/.kube/config** using the **Download File** option at the top of the Cloud Shell window.

   If using Minikube, it is recommended to embed the certificates used for authentication inside the kubeconfig file. This can be done with the following command in Minikube:
   minikube config set embed-certs true

   If the backup of one LoadMaster is restored to another LoadMaster, it is recommended to upload the **Kube Config** file separately to the restored LoadMaster.
4. Click **Install**.
5. Once the **Kube Config** file is successfully installed, some information is populated in the **Contexts** section.

   The **Name**, **Cluster**, and **User** are shown.
6. Select the relevant **K8S Operations Mode**.

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

> For further details on each mode, including instructions on how to configure each mode in Kubernetes, refer to the **Service Mode and Ingress Mode** section.

7. Select the **Namespace to Watch**.

> All namespaces are watched if this field is unset.

8. Optional: Set the **Ingress Watch Timeout** in seconds.

> This is the Ingress Controller watch timeout. Valid values range from 30 - 900.

When configured correctly, details on the Kubernetes **Nodes** and relevant objects (Ingress Objects with an Ingress class specification of "kempLB" and Service Objects labelled "kempLB:Enabled") are shown at the bottom of the screen.

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

5 Service Mode and Ingress Mode

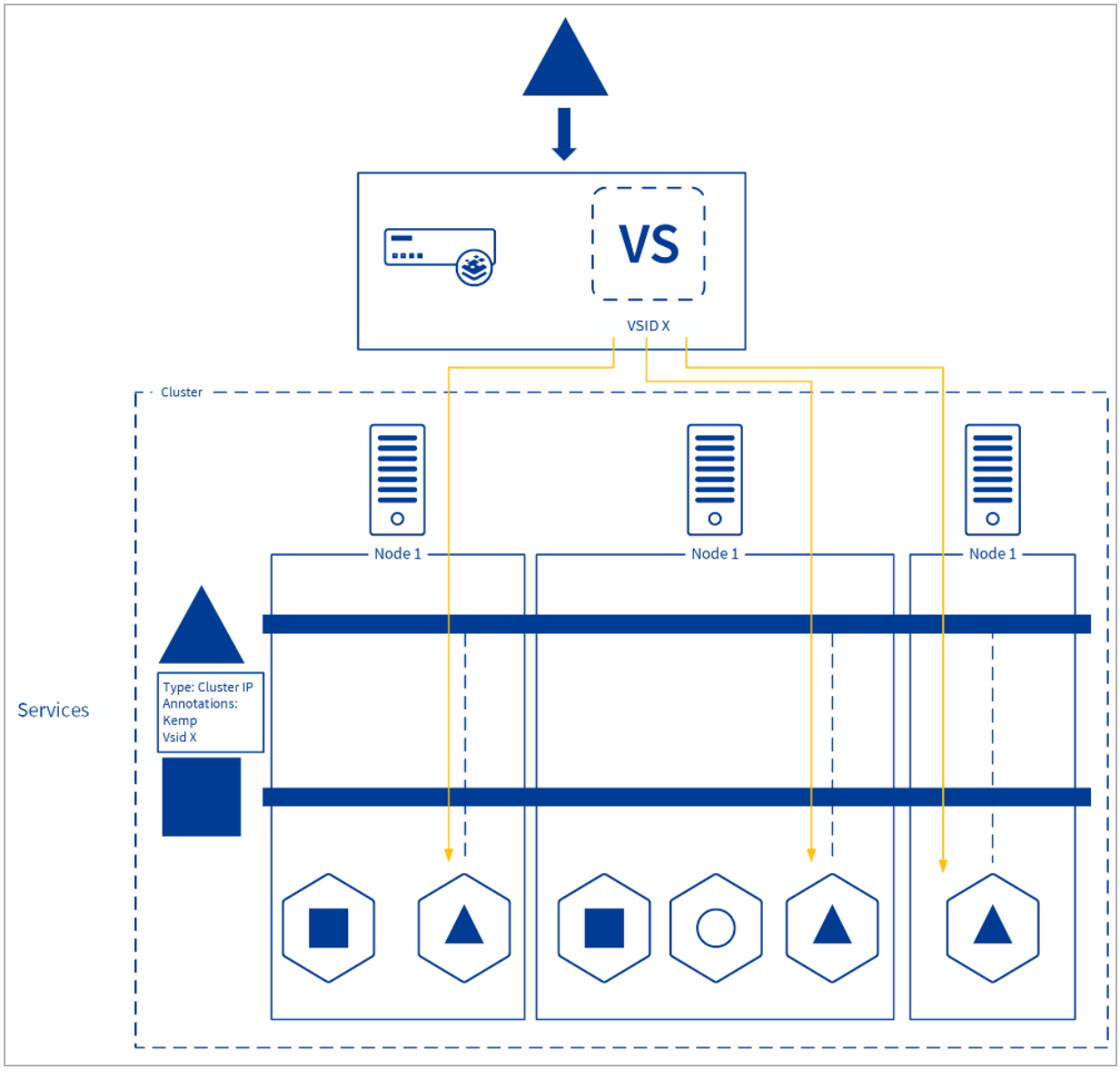
The Kemp Ingress Controller supports two modes of operation:

- Service Mode: In this mode, the LoadMaster configuration is driven through configuration of Service Objects in Kubernetes.
- Ingress Mode: In this mode, the LoadMaster configuration is driven through configuration of the Ingress Object in Kubernetes.

For further details on each mode, refer to the sections below.

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

5.1 Service Mode



Service mode is an operating mode developed by Kemp to meet the needs of Network Operations and Application Developers to work together more seamlessly despite different tool-chains and working practices. This is done by enabling the Network Operations team to create dedicated Virtual Services that can be provided to Application Developers operating in Kubernetes. By dynamically updating based on the state of the defined application in Kubernetes this allows for dynamic application publishing without exposing full access to the LoadMaster.

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

Service mode allows you to expose a service on the Kubernetes cluster that is not available to access externally by tagging the service with a Virtual Service ID. The LoadMaster makes API calls to determine what pods are linked to the service and then adds the appropriate Real Servers to the Virtual Service.

The advantages and disadvantages of service mode are listed below.

Advantages:

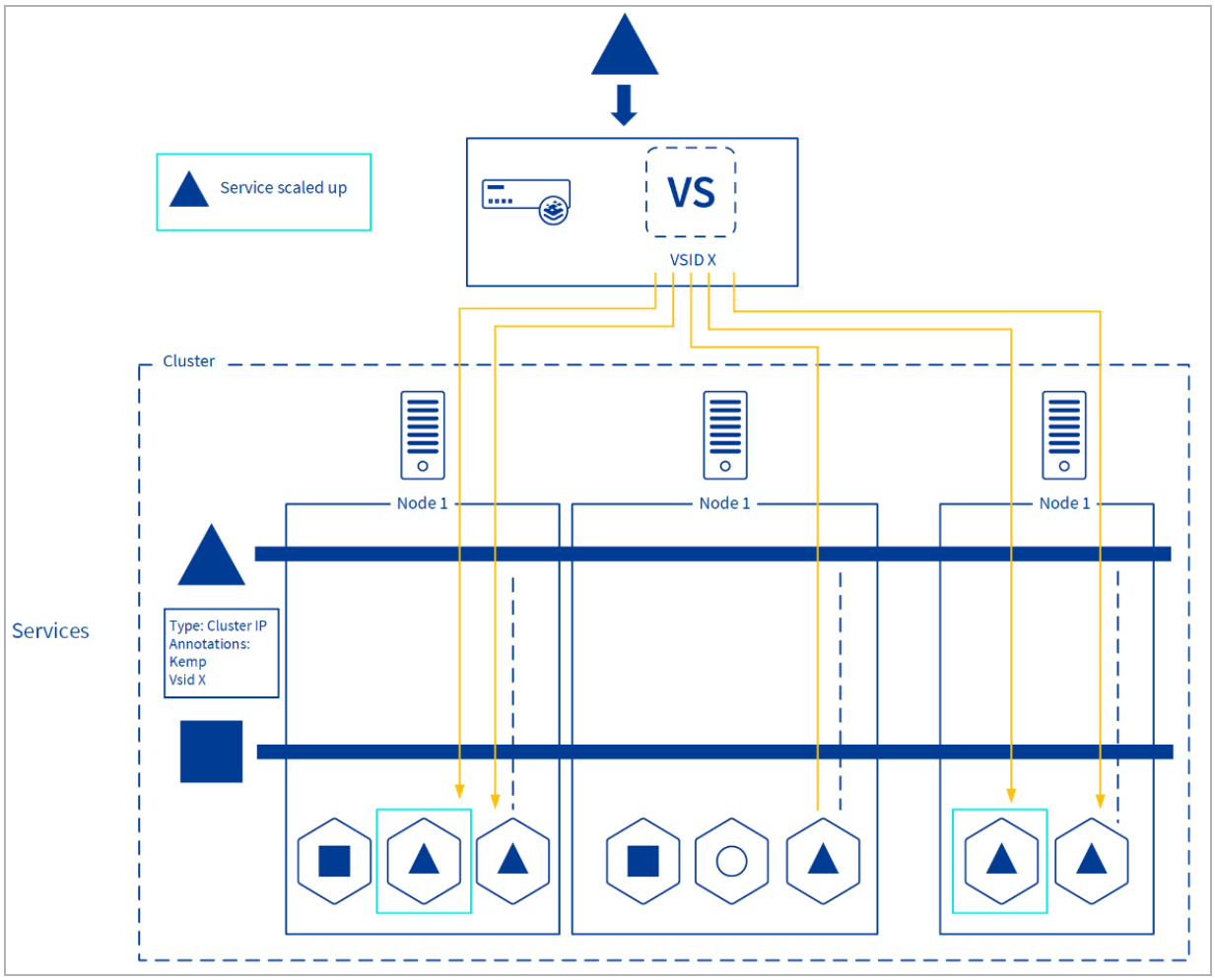
- Efficient routing of traffic to pods
- Eliminates unnecessary East-West traffic
- LoadMaster changes are restricted to the specific Virtual Service defined

Disadvantages:

- May need additional routes to pods defined
- The pod network must not overlap with network IPs
- Nodes must be on the same subnet as the LoadMaster
- A single Virtual Service per Kubernetes Service is required

To enable Service Mode, a Kubernetes service is annotated as being "KempLB enabled" along with a Virtual Service reference (typically the Virtual Service ID) to link it to a Virtual Service on the LoadMaster. The LoadMaster monitors the configuration and if it matches an existing Virtual Service, the appropriate Real Servers are created to route traffic into the service pods. In this mode, the Virtual Service must pre-exist and no new Virtual Service is created. If the service scales, more Real Servers will be added.

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.



To illustrate the operation of service mode, if the number of pods scales up from three to five, for example, the Virtual Service updates automatically and will have five Real Servers.

5.1.1 How Service Mode Works

In Service Mode, a Virtual Service is created on the LoadMaster and details of this (for example VSID) is provided to the Application Developers (or persons configuring Kubernetes). When publishing a Kubernetes Service it is labeled as kempLB-enabled and associated with the Virtual Service (for example, through the VSID). The LoadMaster automatically detects this and configures itself to match the Service configuration defined in Kubernetes by adding the appropriate Real Servers. This Virtual Service then dynamically changes when pods are added/created/destroyed.

5.1.2 When to Use Service Mode

Service Mode is suitable if you are a:

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

- NetOps Team without knowledge or access of the Kubernetes infrastructure
- NetOps Team who uses a different deployment and configuration management toolchain than their AppDev Teams
- Managed Service Provider operating shared Kubernetes and Network infrastructure while their customers self-manage their Kubernetes-based applications

5.1.3 Configure Service Mode

Refer to the sections below for instructions on how to configure service mode.

5.1.3.1 Configure a LoadMaster Virtual Service

You must configure a Virtual Service:

1. In the LoadMaster UI, go to **Virtual Services > Add New**.
2. Expand the **Real Servers** section.
3. Ensure the **Real Server Check Method** is set to **HTTP Protocol**.
4. Select **GET** as the **HTTP Method**.
5. Do not modify any of the other default settings.
6. Take note of the Virtual Service **Id** number.

> The Virtual Service **Id** number is available at the top of the Virtual Service modify screen. This is needed to connect your Kubernetes service to the LoadMaster Virtual Service.

5.1.3.2 Configure the Kubernetes Service

Follow the steps below to link an existing service in Kubernetes to the LoadMaster Virtual Service:

1. Modify the relevant YAML file.
2. Add the following label:

   kempLB: Enabled

3. Add the following annotation:

   "vsid":"<VirtualServiceID>"

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

> Throughout this document, angular brackets (**< >**) indicate a value that is variable that you must configure appropriately based on your environment.

4. In the Kubernetes Shell, apply the following configuration:

   kubectl apply -f <Filename>.yaml

5. In the Kubernetes Shell, run the following command to see the service details:

   kubectl describe service <Name>

Here is an example of a YAML file:

```
apiVersion: v1
kind: Service
metadata:
       name: <Name>
       labels:
              kempLB: Enabled
       annotations:
              "vsid": "<VirtualServiceID>"
spec:
       type: ClusterIP
       ports:
              - port: <VirtualServicePort>
       selector:
              app: <ApplicationSelector>
```

5.1.3.3 Verify the Application

The LoadMaster should have picked up that a Kubernetes service has been labeled as **kempLB=Enabled** and configured the LoadMaster Virtual Service with matching VSID with Real Servers corresponding to Kubernetes Services pods. To verify this, follow the steps below:

1. In the LoadMaster UI, go to **Virtual Services > View/Modify Services**.

   > You should see the status as up/green and a Real Server matching the IP address of your Kubernetes pod.

2. Navigate to the Virtual Service IP address in your browser.

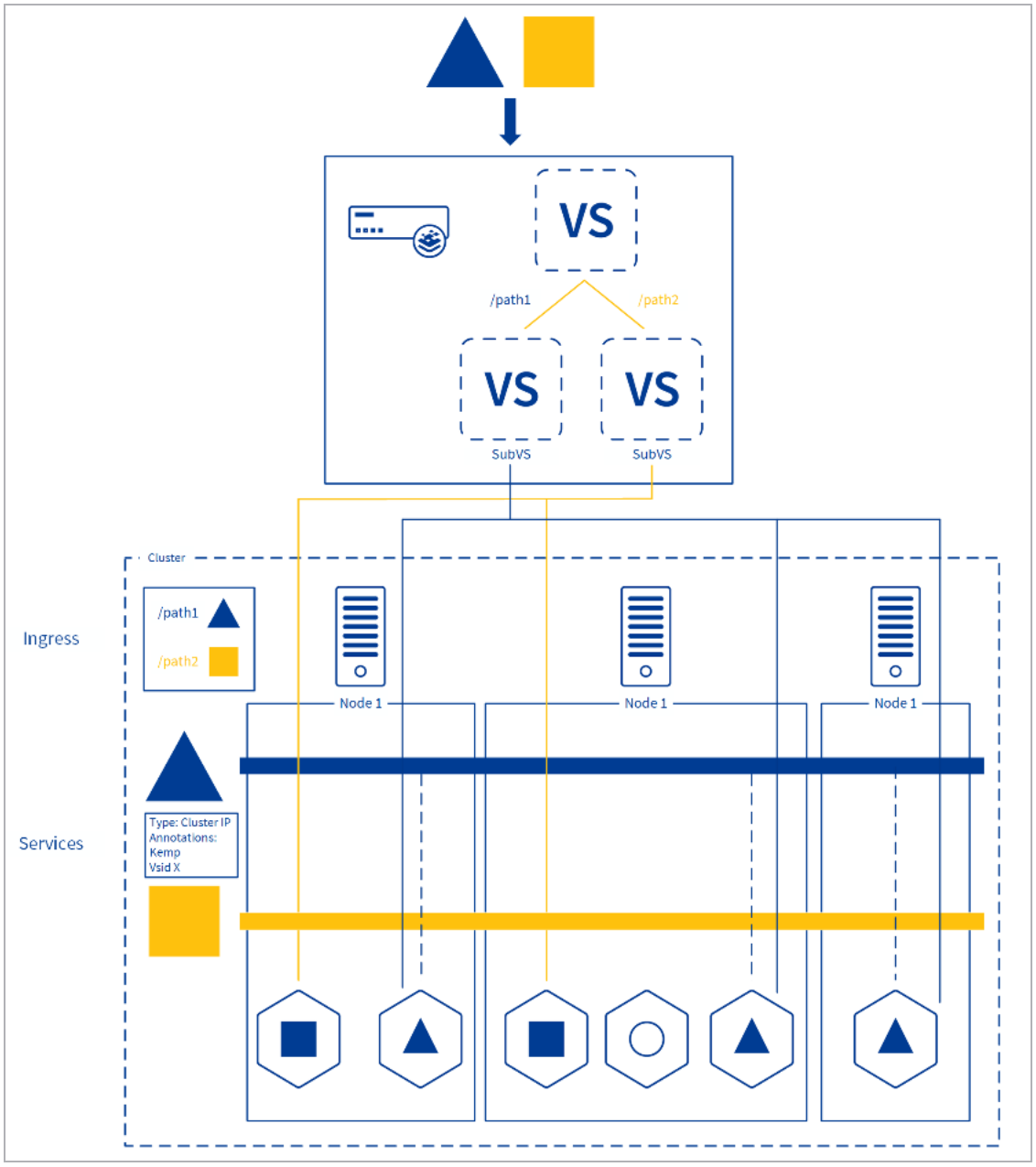
   > You should be able to open the appropriate application.

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

If you deploy more replicas of the service, these should also appear as Real Servers in the LoadMaster Virtual Service. In your Kubernetes Shell, run the following command to see this work:

kubectl scale --replicas=2 deployment/<Application>

5.2 Ingress Mode



© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

Ingress Mode is designed to allow DevOps Teams to use the LoadMaster as an Ingress Controller for their Kubernetes Clusters in place of containerized Ingress controllers.

Ingress mode allows you to define the Ingress with an Ingress controller. The LoadMaster creates one Virtual Service with one or more SubVSs. Any Virtual Services created by Kubernetes are labelled as **L7+K8S** in the **Layer** column on the **View/Modify Services** screen.

Each SubVS maps to the corresponding service in Kubernetes as defined by the Ingress Object rules. This means you do not need a separate Virtual Service for each service. The one Virtual Service performs routing based on the path and/or host as defined. If a new pod is added, a new Real Server gets automatically added to the relevant SubVS.

In Ingress Mode, Ingress IP address and port numbers are defined in the Kubernetes Ingress Object in addition to attributes used to route specific traffic to particular service pods. In this mode, the Virtual Service is created based on the information defined (if appropriate values are used). If more traffic paths and services are added, these are updated on the Virtual Service with each service typically represented as a SubVS. If a service scales, more Real Servers are added.

In Ingress mode, it is possible to define multiple Ingress resources. You can map the master Ingress resource (with multiple associated Slave Ingress resources) to one Virtual Service using the following annotation:

kemp.ax/ingress.ref: "<namespace>:<ingressname>"

It is necessary to ensure that either:

- All the Ingress resources are defined in one namespace (Master, Slave, and Services), or;
- The master Ingress is defined in the "default" namespace and the Slave Ingress resources are defined in different namespaces

> Multiple Ingress resources cannot be mapped to one Virtual Service if the master Ingress is defined in a namespace other than "default" and the Slave Ingress resources are also defined in different namespaces.

Here is an example YAML file:

> The new Ingress Class (**ingressClassName: lmingress**) is supported on Kubernetes versions 1.22 and above.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
```


© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

```
metadata:
        name: master-ingress
        namespace: default
        annotations:
                "kemp.ax/vsip": "10.35.46.104"
                "kemp.ax/vsport": "443"
                "kemp.ax/vsprot": "tcp"
                "kemp.ax/vstype": "http"
                "kemp.ax/sslaccel": "1"
                "kemp.ax/certfile": "server"
spec:
        ingressClassName: lmingress
        defaultBackend:
                service:
                        name: qa-default
                        port:
                                number: 80
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
        name: slave-ingress1
        namespace: slave1
        annotations:
                "kemp.ax/ingress.ref": "default/master-ingress"
spec:
        ingressClassName: lmingress
        rules:
        - host: slave1.kemptech.net
                http:
                        paths:
                        - path: /api/v1/deployer
                                pathType: Prefix
                                backend:
                                        service:
                                                name: service1
                                                port:
                                                        number: 80
                        - path: /api/v1/logout
                                pathType: Prefix
                                backend:
                                        service:
                                        name: service1
```


© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

```
                                        port:
                                                number: 80
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
       name: slave-ingress2
       namespace: slave2
       annotations:
              "kemp.ax/ingress.ref": "default/master-ingress"
spec:
       ingressClassName: lmingress
       rules:
       - host: slave2.kemptech.net
              http:
                     paths:
                     - path: /api/v1/server
                            pathType: Prefix
                            backend:
                                   service:
                                          name: service2
                                          port:
                                                 number: 80
                     - path: /api/v1/authenticate
                            pathType: Prefix
                            backend:
                                   service:
                                          name: service2
                                          port:
                                                 number: 80
```

If the LoadMaster firmware is updated from an older version (other than 7.2.55) that already has an Ingress resource defined then after update the older configuration of VS and SubVS remains unchanged until you delete the older SubVS entries to get the updated rule entries for the SubVS.

Ingress Mode is the standard Kubernetes Ingress Controller operating mode designed for cross-functional teams operating purely through the Kubernetes API.

The advantages and disadvantages of Ingress mode are listed below.

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

Advantages:

- Efficient routing of traffic to pods
- Eliminates unnecessary East-West traffic
- Single Virtual Service for multiple services
- No need for double load balancing
- Kubernetes endpoints can be administered along with monolithic load-balanced services
- Create multiple rules for one service/SubVS instead of multiple SubVSs

Disadvantages:

- May need routes to pods defined
- Pod network must not overlap with network IP addresses
- Nodes must be on the same subnet as the LoadMaster

5.2.1 How Ingress Mode Works

The Kubernetes API defines an 'Ingress Controller' object which allows DevOps Teams to publish their Services to external systems and users.

Ingress Mode allows DevOps Teams to define a new class of Ingress Controller utilizing the LoadMaster which automatically detects and matches the configuration as defined in Kubernetes.

The LoadMaster detects when Services Endpoints (that is, Kubernetes Pods) are added or removed and ensures the LoadMaster SubVS Real Servers are updated accordingly.

5.2.2 When to Use Ingress Mode

Ingress Mode is designed and recommended if you are a:

- Cross-functional DevOps Team who own and operate both the network and Kubernetes infrastructure in addition to the applications
- NetOps Team with knowledge of and access to the Kubernetes infrastructure
- NetOps Team who use the same deployment toolchain and processes as their AppDev Teams
- Managed Service Provider operating shared Kubernetes and Network infrastructure in addition to managing their customers' Kubernetes-based applications

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

5.2.3 Configure Ingress Mode

Refer to the sections below for instructions on how to configure ingress mode.

5.2.3.1 Create a Kubernetes Ingress Controller

You must define a Kubernetes Ingress Object that stores the configuration that the LoadMaster requires. Here is an example YAML file:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
       name: kemp-ingress
       annotations:
               "kubernetes.io/ingress.class": "kempLB"
               "kemp.ax/vsip": "<VirtualServiceAddress>"
               "kemp.ax/vsport": "<VirtualServicePort>"
               "kemp.ax/vsprot": "<tcp/udp>"
spec:
       ingressClassName: lmingress
       rules:
       - host: <HostAddress1>
               http:
                       paths:
                       - path: </Path1>
                       pathType: <Prefix/ImplementationSpecific/Exact>
                       backend:
                               service:
                                       name: <ServiceName1>
                                       port:
                                       number: <ServicePort1>
       - host: <HostAddress2>
    http:
      paths:
      - path: </Path2>
        pathType: <Prefix/ImplementationSpecific/Exact>
        backend:
          service:
          name: <ServiceName2>
          port:
            number: <ServicePort2>
```

In your Kubernetes Shell session, run:

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

kubectl apply -f <Filename>.yaml

In your Kubernetes Shell session, run the following command to see the Ingress details:

kubectl describe ingress <Name>

5.2.3.2 Verify the Application

The LoadMaster should have picked up that a Kubernetes Ingress controller has been published and configured a new Virtual Service (if necessary) and the required SubVS(s) for each Kubernetes Services pod. To verify this, follow the steps below:

1. In the LoadMaster UI, go to **Virtual Services > View/Modify Services**.

   > You should see the status as up/green and a SubVS. If you click the Virtual Service IP address it should expand to show the SubVS with a Real Server matching the IP address of your Kubernetes pod.

2. Modify the hosts file of your client system to resolve an address to the Virtual Service IP address.

3. Navigate to this address in your browser. This should open the application. By using different hostnames or paths you should be able to access the different services as defined in the Ingress Rules.

4. If you deploy more replicas of the service, these should also appear as Real Servers in the LoadMaster SubVS. To do this, in your Cloud Shell run:

   kubectl scale --replicas=2 deployment/<Application>

5.2.4 Additional Annotations for Ingress Mode

In addition to defining Virtual Service attributes for creation (for example, Virtual Service IP addresses and port), an Ingress object may also be used to define other attributes. Below is a table of attributes supported. For further details on each attribute, refer to the RESTful API Interface Description.

Ingress attr	lmapi attr
addvia	AddVia
alertthresh	AlertThreshold
allowhttp2	AllowHTTP2

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

Ingress attr	lmapi attr
altaddress	AltAddress
cache	Cache
cachepercent	CachePercent
certfile	CertFile
checkhost	CheckHost
checkport	CheckPort
checkurl	CheckUrl
checkuse1.1	CheckUse1.1
ciphers	Ciphers
cipherset	CipherSet
clientcert	ClientCert
compress	Compress
copyhdrfrom	CopyHdrFrom
copyhdrto	CopyHdrTo
defaultgw	DefaultGW
enable	Enable
enhealthchecks	EnhancedHealthChecks
errorcode	ErrorCode
errorurl	ErrorUrl
espenabled	EspEnabled
extraports	ExtraPorts
followvsid	FollowVSID
forcel7	ForceL7
idletime	Idletime

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

Ingress attr	lmapi attr
inauthmode	InputAuthMode
locbindaddr	LocalBindAddrs
machlen	MatchLen
multiconnect	MultiConnect
needhostname	NeedHostName
non_local	non_local
ocspverify	OCSPVerify
outauthmode	OutputAuthMode
persist	persist
perstout	PersistTimeout
portfollow	PortFollow
preprec	PreProcPrecedence
preprecpos	PreProcPrecedencePos
qos	QoS
querytag	QueryTag
reqprec	RequestPrecedence
reqprecpos	RequestPrecedencePos
reqrules	RequestRules
respprec	ResponsePrecedence
resprecpos	ResponsePrecedencePos
resprules	ResponseRules
rsminimum	RsMinimum
rsnihostname	ReverseSNIHostname
shed	Schedule

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

Ingress attr	lmapi attr
sechdropt	SecurityHeaderOptions
serverinit	ServerInit
sslaccel	SSLAcceleration
sslreencrypt	SSLReencrypt
sslreverse	SSLReverse
sslrewrite	SSLRewrite
standbyaddr	StandbyAddr
standbyport	StandbyPort
starttlsmode	StartTLSMode
subnetorig	SubnetOriginating
sslaccel	SSLAcceleration
tlstype	TLSType
transparent	Transparent
useforsnatt	UseforSnatt
verify	Verify
vsaddr	VSAddress
vsip	VSAddress
vsname	NickName
vsport	Protocol
vstype	VStype

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

6 Ensuring Connectivity from the LoadMaster to Pods

When running Kubernetes there are multiple options for the configuration of how pods can communicate. Kubernetes requirements are that pods on a node can communicate with all pods on all nodes without Network Address Translation (NAT). Agents on a node (for example, system daemons, kubelet) can communicate with all pods on that node and pods in the host network of a node can communicate with all pods on all nodes without NAT. For further details, refer to the following Kubernetes page: Cluster Networking.

There are a number of ways to implement these and it varies based on the Container Network Interface (CNI) plugins used, cloud provider integrations, or any Border Gateway Protocol (BGP) peering with the physical network in use.

If the pods are routable from outside the network there is no requirement for outbound Secure Network Address Translation (SNAT) and pods can be accessed directly without using Kubernetes Services. The disadvantages are that the IP addresses used must be reserved for pod use.

6.1 Overlay Networks

An overlay network allows network devices to communicate across an underlying network (referred to as the underlay) without the underlay network having any knowledge of the devices connected to the overlay network.

There are a number of common overlay network types that can be used such as Flannel Calico and Weave Net. Typically, when using an overlay network, pods are assigned with an address from a set of addresses assigned to the overlay network and this is different from the Host Network in which the Node's IP addresses are from.

In most cases when using an overlay network, it is typical that pods are not routable.

6.2 How to Check if an Overlay Network is in Use

If you are unsure, the simplest way to check if an overlay network is used and pods are not routable is to view the pod IP addresses and compare to the IP addresses of the nodes. If these are on a different network space then routes must be added.

For example:

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

```
[root@master-node ~]# kubectl get nodes -o wide
NAME STATUS ROLES AGE VERSION INTERNAL-IP EXTERNAL-IP OS-IMAGE KERNEL-VERSION CONTAINER-
RUNTIME
master-node Ready master 46d v1.19.2 10.1.151.30 <none> CentOS Linux 7 (Core) 3.10.0-
1127.19.1.el7.x86_64 docker://19.3.13
node-1 Ready <none> 46d v1.19.2 10.1.151.31 <none> CentOS Linux 7 (Core) 3.10.0-
1127.19.1.el7.x86_64 docker://19.3.13
node-2 Ready <none> 46d v1.19.2 10.1.151.32 <none> CentOS Linux 7 (Core) 3.10.0-
1127.19.1.el7.x86_64 docker://19.3.13
[root@master-node ~]#
[root@master-node ~]# kubectl get pods -o wide
NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES
azure-vote-back-59d587dbb7-x297k 1/1 Running 1 3d 192.168.84.150 node-1 <none> <none>
azurefront-5f47b44b9b-nhdwb 1/1 Running 1 3d 192.168.247.14 node-2 <none> <none>
frontend-6c6d6dfd4d-6xzpd 1/1 Running 1 3d 192.168.247.2 node-2 <none> <none>
frontend-6c6d6dfd4d-g26cb 1/1 Running 1 3d 192.168.247.1 node-2 <none> <none>
frontend-6c6d6dfd4d-qbbsv 1/1 Running 1 3d 192.168.84.155 node-1 <none> <none>
redis-master-f46ff57fd-jjhwx 1/1 Running 1 3d 192.168.84.153 node-1 <none> <none>
redis-slave-bbc7f655d-5xnkk 1/1 Running 1 3d 192.168.84.156 node-1 <none> <none>
redis-slave-bbc7f655d-n9vgh 1/1 Running 1 3d 192.168.247.12 node-2 <none> <none>
```

In the above example, it is clear that the PODs are on a different network to the nodes.

6.3 LoadMaster to Pod Connectivity

When using Kemp Ingress Controller for Kubernetes, because the LoadMaster operates outside the Kubernetes Cluster, when it receives traffic it needs a mechanism to route to the correct pods. This means there must be network connectivity between the LoadMaster and the pods. The simplest way to achieve this is to manually create the necessary routes. It is important to note that to do this the LoadMaster must have an interface in the same network as the Kubernetes Hosts.

6.4 Adding Routes

To see what routes are necessary the following Kubernetes commands can be used.

For most CNIs the following can be used:

```
kubectl get nodes -o jsonpath="{range .items[*]}{'Destination: '}
{.spec.podCIDR}{'\t'}{'Gateway: '}{.status.addresses[0].address}{'\n'}{end}"
```


© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

For example:

```
[root@master-node ~]# kubectl get nodes -o jsonpath="{range .items[*]}
{'Destination: '}{.spec.podCIDR}{'\t'}{'Gateway: '}{.status.addresses
[0].address}{'\n'}{end}"
```

```
Destination: 192.168.0.0/24 Gateway: 10.1.151.30
Destination: 192.168.1.0/24 Gateway: 10.1.151.31
Destination: 192.168.2.0/24 Gateway: 10.1.151.32
```

The above command tells us what routes are needed on the LoadMaster and you can add these routes in the LoadMaster UI by going to **System Configuration > Network Setup > Additional Routes**.

Destination	Gateway
192.168.0.0/24	10.1.151.30
192.168.1.0/24	10.1.151.31
192.168.2.0/24	10.1.151.32

6.4.1 Calico

If Calico is being used for the overlay network, the commands to use are a little different.

```
kubectl get nodes -o jsonpath="{range .items[*]}{'Destination: }
{.metadata.annotations.projectcalico\.org/IPv4IPIPTunnelAddr}{'\tGateway: '}
{.metadata.annotations.projectcalico\.org/IPv4Address}{'\n'}"
kubectl get IPpool -o jsonpath="{range .items[*]}{'Destination Mask: /'}
{.spec.blockSize}{'\n'}"
```

For example:

```
[root@master-node ~]# kubectl get nodes -o jsonpath="{range .items[*]}
{'Destination: '}{.metadata.annotations.projectcalico\.org/IPv4IPIPTunnelAddr}
{'\tGateway: '}{.metadata.annotations.projectcalico\.org/IPv4Address}{'\n'}"
```

```
Destination: 192.168.77.128 Gateway: 10.1.151.30/24
Destination: 192.168.84.128 Gateway: 10.1.151.31/24
Destination: 192.168.247.0 Gateway: 10.1.151.32/24
Destination: Gateway:
```

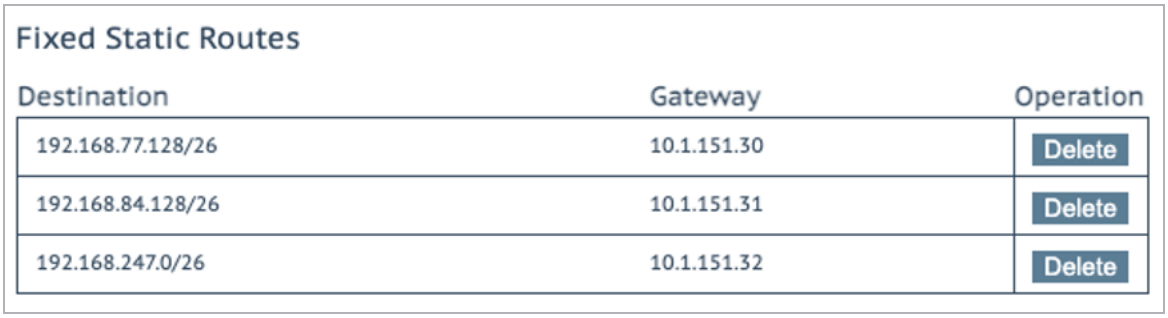
> The above does not provide the netmask to use for the Destination Address Space. By default, Calico uses /26 but this can be verified using the following command.

```
[root@master-node ~]# kubectl get IPpool -o jsonpath="{range .items[*]}
{'Destination Mask: /'}{.spec.blockSize}{'\n'}"
```

```
Destination Mask: /26
```


© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

```
Destination Mask: /
[root@master-node ~]#
```



Routes to add to the LoadMaster (in **System Configuration > Network Setup > Additional Routes**):

Destination	Gateway
192.168.77.128/26	10.1.151.30
192.168.84.128/26	10.1.151.31
192.168.247.0/26	10.1.151.32

The subnet mask defaults to /26 for Calico.

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

7 Creation and Deletion of Objects

When using the Kemp Ingress Controller for Kubernetes it is important to understand the expected behaviour with regard to deletion and creation of Virtual Services and Real Servers on the LoadMaster.

'Desired State' is one of the core principles of Kubernetes, whereby a desired configuration is defined and the Kubernetes Controller monitors this and implements any changes required to meet the desired state (for example creating/destroying pods in the cluster). When using the Kemp Ingress Controller for Kubernetes, the LoadMaster operates in a similar manner. The LoadMaster periodically monitors the desired state of how traffic should reach services and updates the configuration (if needed) to reflect this.

It is important to understand some expected behaviour when using the Kemp Ingress Controller for Kubernetes in the two modes of operation:

- Service Mode: In this mode, a service may be annotated with attributes to link it to an existing Virtual Service (typically the Virtual Service ID) on the LoadMaster. The LoadMaster monitors the configuration and if it matches an existing Virtual Service, the appropriate Real Servers are created to route traffic into the service pods. In this mode the Virtual Service must pre-exist and no new Virtual Services are created. If the service scales, more Real Servers are added.
  If Real Servers are manually deleted on the LoadMaster, the service definition will still map to an existing Virtual Service and will simply re-add these Real Servers. However, If a Virtual Service is deleted, the service will no longer have a valid Virtual Service defined in the Service configuration and no Virtual Service will be recreated.
  You can delete a Virtual Service using the LoadMaster (along with removing the Virtual Service ID on the Service) but when removing a Real Server this should be done using Kubernetes pod scaling.
- Ingress Mode: In Ingress Mode, the LoadMaster uses the Ingress Object created in Kubernetes. This defines Ingress IP:Port numbers in addition to other attributes used to route to particular services. In this mode the Virtual Service itself is created based on the information defined here (if appropriate values are used). If more traffic paths and services are added, these are updated on the Virtual Service with each service typically represented as a SubVS. If any one

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

service scales, more Real Servers are added.
If the Virtual Service or Real Server is deleted on the LoadMaster, this results in the Virtual Service/Real Server being recreated as long as the Ingress object still exists in Kubernetes which defined these as a desired state.
When deleting a Virtual Service, you should do this using the Kubernetes Ingress Object and when adding/removing Real Servers this should be done using Kubernetes pod scaling.

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

8 LoadMaster Ingress Logs

You can access the Kemp Ingress Controller logs by going to **System Configuration > Logging Options > Extended Log Files**. There are two log types:

- **Ingress Controller Logs**
- **Ingress Resource Watcher Logs**

The Ingress logs are only shown when:

- The LoadMaster Ingress controller is installed, and;
- Ingress logs exist

The following information can be seen in the logs:

- Issues connecting with Kubernetes
- Virtual Services created by Kemp Ingress Controller
- Changes made as a result of scaling
- Configuration changes
- Unrecognized Virtual Service IDs
- Incorrect YAML configuration

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

Last Updated Date

This document was last updated on 27 July 2023.

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.