F5 CIS Operations Guide Documentation

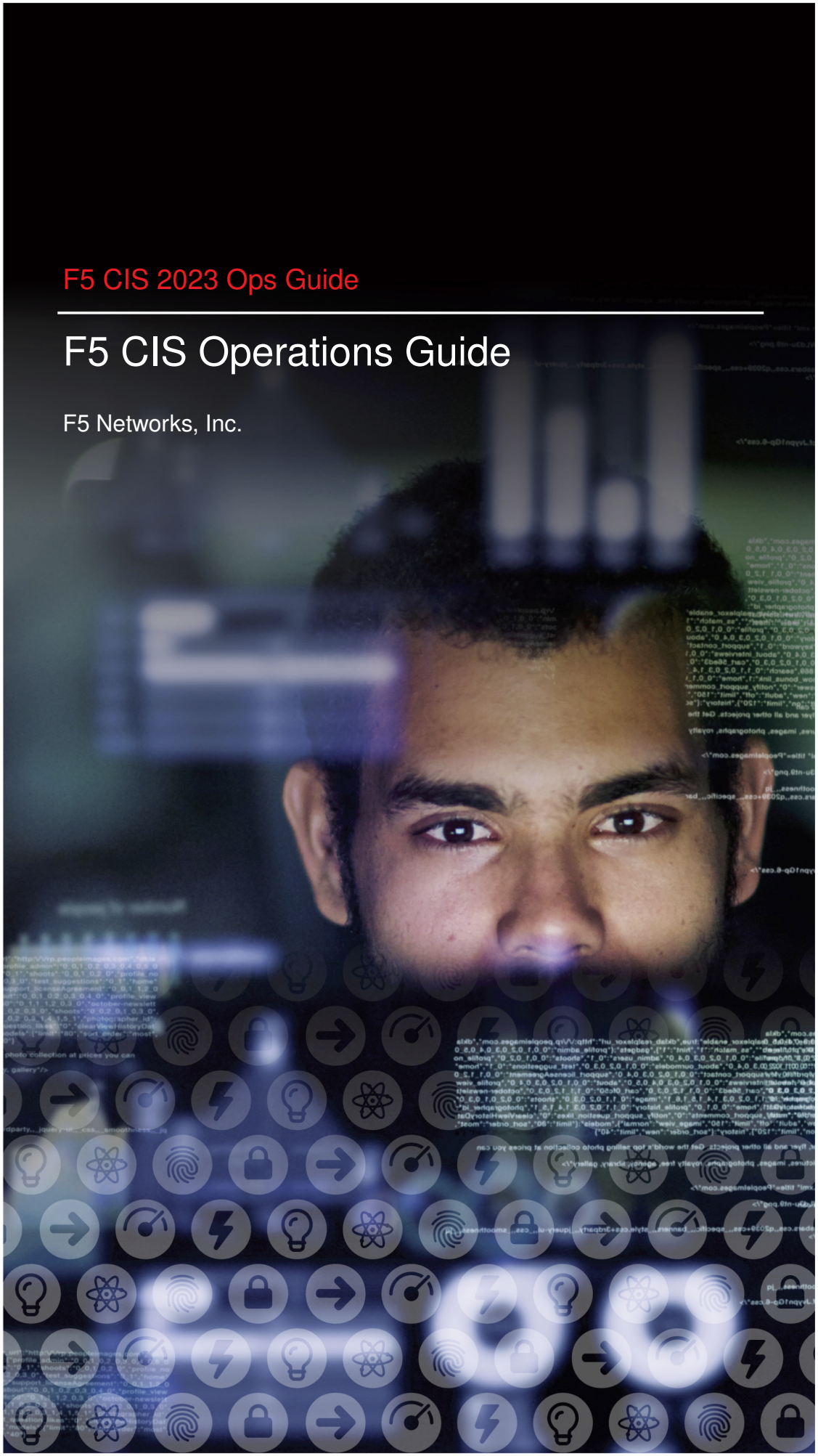
F5 Networks, Inc.

Jun 13, 2023

F5 CIS 2023 Ops Guide

F5 CIS Operations Guide

F5 Networks, Inc.



Contents:

1 Overview

This unofficial document provides best practices, tips, and caveats when validating F5 Container Ingress Services (CIS) in a Kubernetes environment.

Note: This is a supplemental guide to official documentation on F5 Cloud Docs. Testing should be done in a staging or sandbox environment to confirm expected behavior before deploying to production.

1.1 F5 Container Ingress Services - Overview

When deploying applications in Kubernetes, you will eventually need a way to provide external access to your services. F5 enables not only load balancing, but also security, advanced functionality and high performance often required by enterprise-grade applications.

The F5 BIG-IP Controller (k8s-bigip-ctlr) is a cloud-native connector that can use either Kubernetes or OpenShift as a BIG-IP orchestration platform.

The BIG-IP Controller watches the Kubernetes API for specially formatted resources with CIS labels. It will take those labels and update the BIG-IP configuration accordingly.

1.2 Features

- Dynamically create, and manage BIG-IP objects.
- Forward traffic from the BIG-IP device to Kubernetes clusters via NodePort or ClusterIP.
- Support F5 AS3 Extension declarations.

2 CIS Architecture and Design

Description: This section will cover some best practices, tips, and caveats when designing various implementations of F5 Container Ingress Services (CIS) in a Kubernetes environment.

2.1 Traffic Groups

Description: F5's ScaleN (N+1) architecture allows you to create a redundant system configuration for multiple BIG-IP devices on a network. This is made possible by using traffic groups.

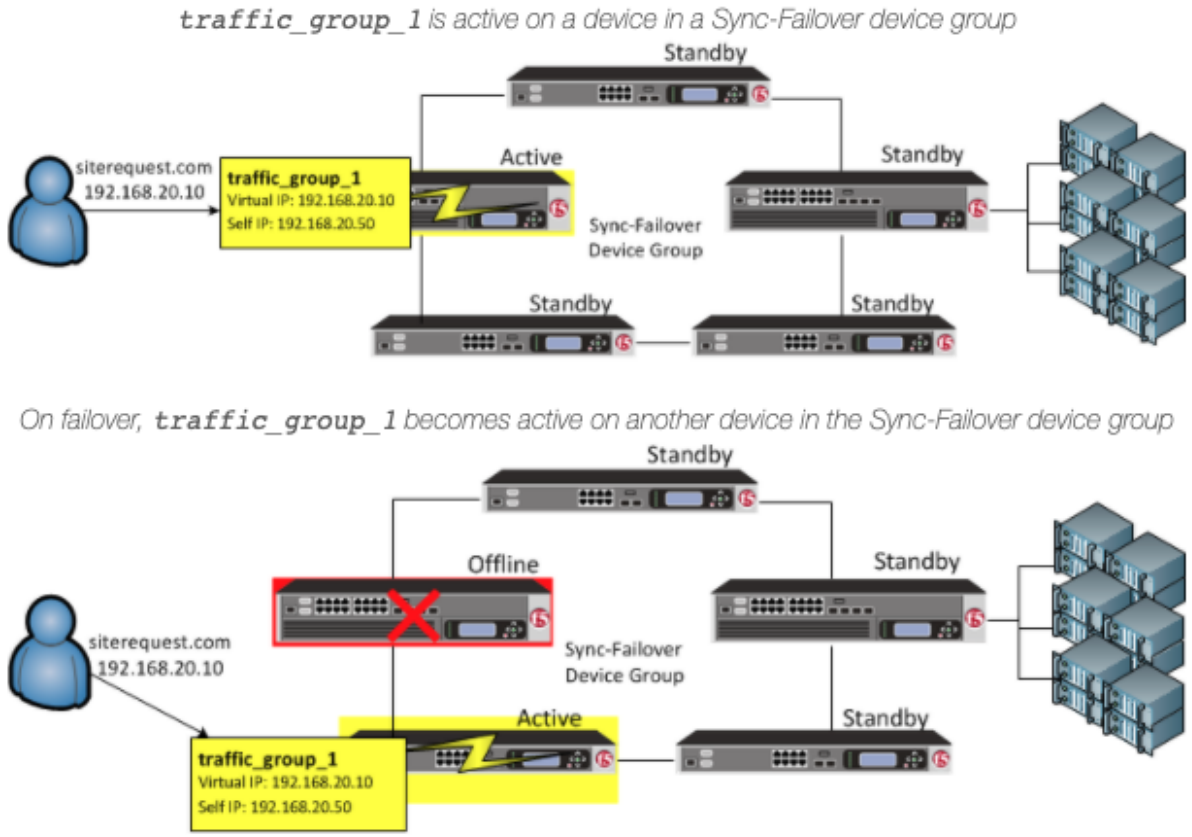
Traffic groups

A traffic group is a collection of related configuration objects, such as a **floating self IP address** , a **virtual IP address** , and a **SNAT translation address** , that run on a BIG-IP device. Together, these objects process a particular type of application traffic on that device. When a BIG-IP device becomes unavailable, a traffic group floats (that is, fails over) to another device in a device group to ensure that application traffic continues to be processed with little to no interruption in service. In general, a traffic group ensures that when a device becomes unavailable, all of the failover objects in the traffic group fail over to any one of the available devices in the device group.

A traffic group is initially active on the device on which you create it, until the traffic group fails over to another device. For example, if you initially create three traffic groups on Device A, these traffic groups remain active on Device A until one or more traffic groups fail over to another device. If you want an active traffic group to become active on a different device in the device group when failover has not occurred, you can intentionally force the traffic group to switch to a standby state, thereby causing failover to another device.

Only objects with floating IP addresses can be members of a floating traffic group.

An example of a set of objects in a traffic group is an iApps application service. If a device with this traffic group is a member of a device group, and the device becomes unavailable, the traffic group floats to another member of the device group, and that member becomes the device that processes the application traffic.



For a full guide to traffic groups, please refer to the official documentation on AskF5 or F5 Cloud Docs.

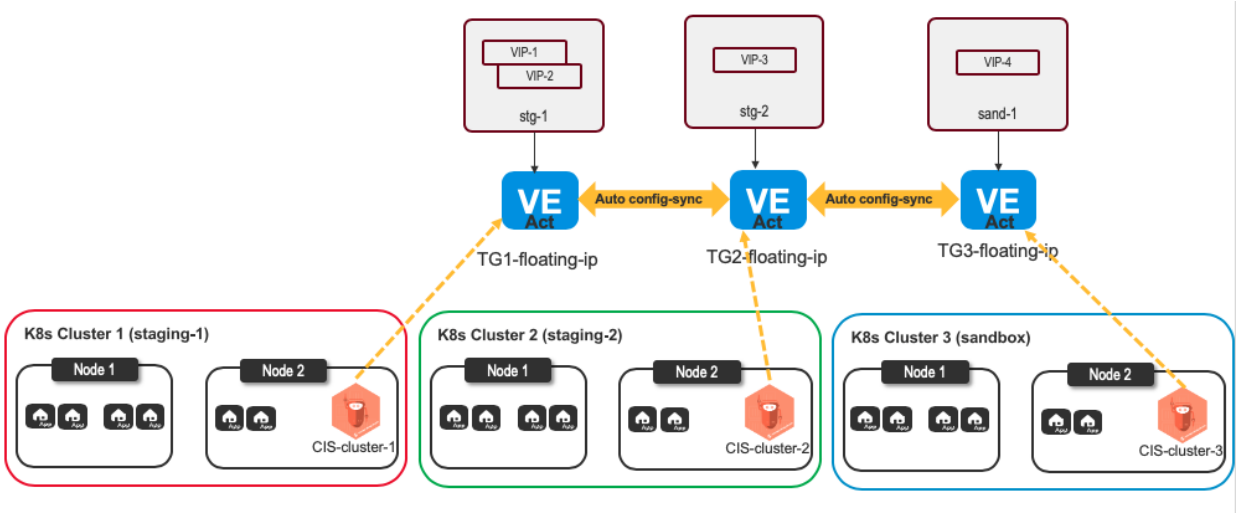
2.2 CIS and ScaleN (N+1) - w/ Auto Config-Sync

Description: The ScaleN architecture allows you to create a redundant system configuration for multiple BIG-IP devices on a network.This guide will focus on the tips and best practices for building this in a lab for testing. For a full guide to the installation, please refer to the official documentation on AskF5 or F5 Cloud Docs.

Prerequisites:

- BIG-IP licenses and basic understanding of the BIG-IP system.
- Existing Kubernetes cluster and basic understanding of the Kubernetes platform.

Sample Diagram:



2.2.1 Configuration tips and caveats

- Ensure proper disaggregation in front of BIG-IP
- Ensure that AS3 Tenant/Partition names *do not* overlap
- **Ensure that AS3 declaration specifies below:**
  - **trafficGroup property**
    - number assignment method example: TG1=prod, TG2=staging
    - *details below*
  - **shareNodes property**
    - To allow Nodeport IPs to be configured in /Common so other partitions can use it
    - *details below*
- **Only Nodeport or potentially Calico BGP could work**
  - auto-sync and Flannel cannot be configured together
- Multi K8s Cluster with 1 CIS deployment per cluster
- BIG-IP in scalen A/A/S with auto config-sync (optional)
- Ensure no more than 3 CIS point to a single VE (max tested)
- Configure HA order to favor a standby before converging TGs on a single VE
- Consider multiple regions of cluster and use GSLB (DNS) for load balancing between regions
- For IPv6 addresses, use `hostAliases` as CIS does not connect right now to an IPv6 address (known issue)

trafficGroup property: You can specify the traffic group associated with any virtual address so that all associated objects float with that traffic group in a ScaleN (N+1) configuration. See F5 Cloud Docs for more details.

You can then reference the `Service_Address` name for the `virtualAddresses` property for your virtual server.

shareNodes property: You can configure `shareNodes` so that multiple tenants can use the same node IP, which gets created in the /Common partition. See F5 Cloud Docs for more details.

serviceMain: If you use a template with a value of http, https, tcp, udp, or l4, you MUST specify an object with the matching Service class Service_HTTP, Service_HTTPS, Service_TCP, Service_UDP, or Service_L4 and name it serviceMain as described in the following Service Class section. See F5 Cloud Docs for more details.

Sample Configuration:

```
kind: ConfigMap
apiVersion: v1
metadata:
name: stg-as3-declaration-demo
namespace: kube-system
labels:
    f5type: virtual-server
    as3: "true"
data:
template: |
    {
    "class": "AS3",
    "action": "deploy",
    "persist": true,
    "declaration": {
        "class": "ADC",
        "schemaVersion": "3.18.0",
        "id": "demoapp",
        "label": "f5-istio",
        "remark": "An HTTP application",
        "stg_tenant": {
        "class": "Tenant",
        "stg_app": {
            "class": "Application",
            "template": "http",
            "stg_svc_addr": {
                "class": "Service_Address",
                "virtualAddress":  "240b:ab11:cd22:a101::10",
                "arpEnabled": false,
                "icmpEcho": "disable",
                "routeAdvertisement": "any",
                "trafficGroup": "/Common/traffic-group-2"
            },
            "serviceMain": {
            "class": "Service_HTTP",
            "virtualAddresses": [{"use": "stg_svc_addr"}],
            "pool": "stg_nginx_pool"
            },
            "stg_nginx_pool": {
            "class": "Pool",
            "monitors": [
                "tcp"
            ],
            "members": [{
                "servicePort": 80,
                "serverAddresses": [],
                "shareNodes": true
            }]
```

(continues on next page)

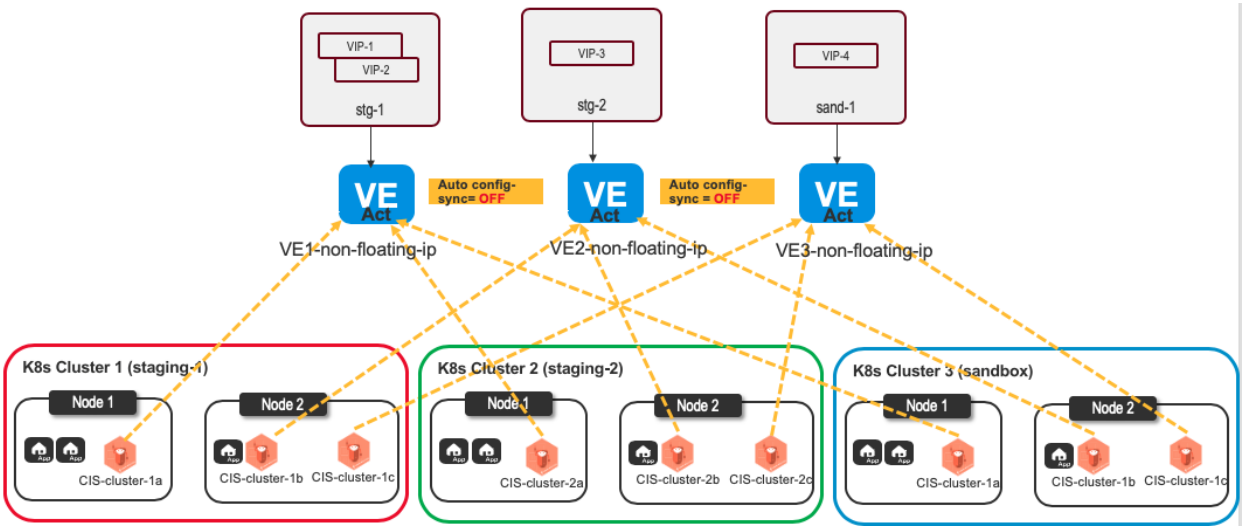
(continued from previous page)

```
                }
                }
            }
            }
        }
```

2.3 CIS and ScaleN (N+1) - w/ out Auto Config-Sync

Description: This design pattern is similar to the previous design except auto config-sync is disabled, therefore multiple CIS instances will be used to configure other devices in the cluster.

Sample Diagram:



2.3.1 Configuration tips and caveats

- Ensure no more than 3 Controllers point to a single VE (max tested)
- **Configure the VE non-floating Self-IP in `--bigip-url`**
  - Since there is one Controller per BIG-IP, no need to configure a floating IP
- Provide a unique metadata.name for each Controller
- Provide a unique `--bigip-url` in each Deployment (each Controller manages a separate BIG-IP device)
- Use the same `--bigip-partition` in all Deployments
- **Do not define multiple Deployment configs in a single manifest.**
  - If you launch multiple BIG-IP Controller instances using a single manifest, they will run on the same Pod. This means that if the Pod goes down, you lose all of your Controllers.

3 CIS Installation and Configuration

Description: This section will cover some best practices, tips, and caveats when installing and configuring F5 Container Ingress Services (CIS) in a Kubernetes environment. For a full guide to these topics, please refer to the official documentation on F5 Cloud Docs.

Prerequisites:

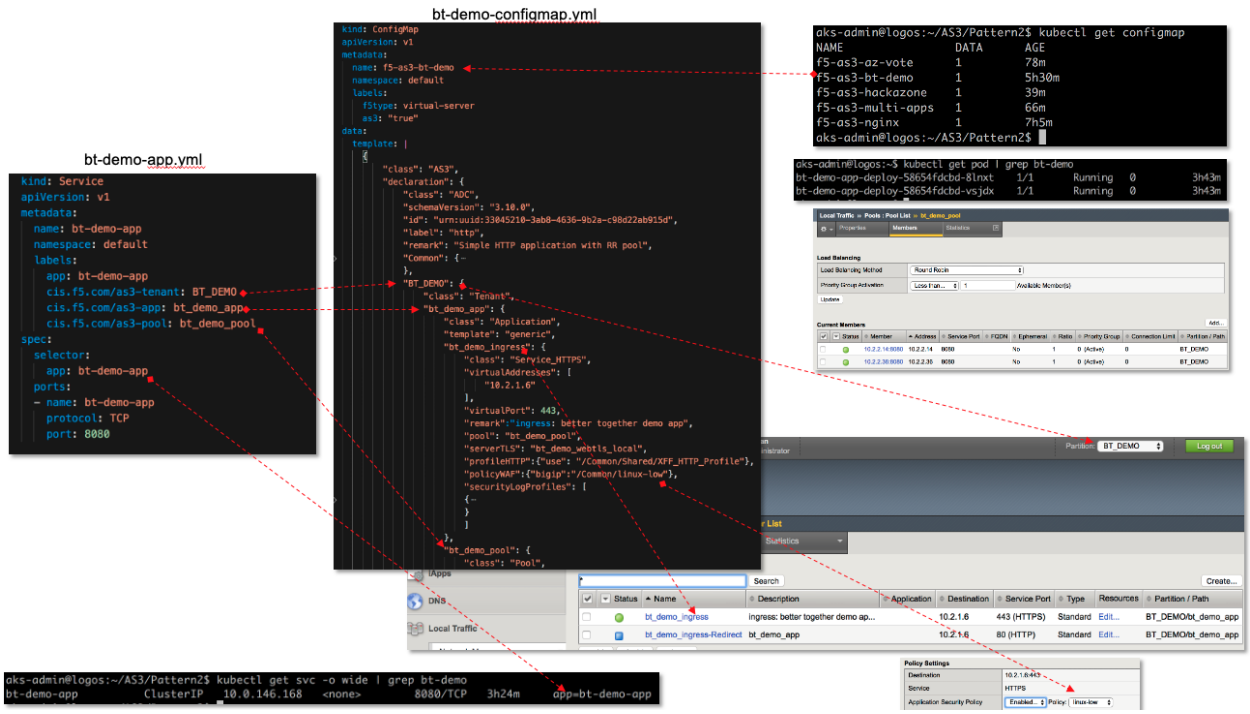
- BIG-IP licenses and basic understanding of the BIG-IP system.
- Existing Kubernetes cluster and basic understanding of the Kubernetes platform.

3.1 CIS and AS3

Description: This section will cover some best practices, tips, and caveats when using AS3 to configure F5 Container Ingress Services (CIS) in a Kubernetes environment. For a full guide to these topics, please refer to the official documentation on F5 Cloud Docs.

Prerequisites: - Basic understanding REST APIs and declarative configuration.

Below is a visualization of the configuration mappings between the AS3 configmap, Kubernetes Service, and the resulting configuration on BIG-IP.



Credit to @foobz for this diagram

3.2 CIS Service Labels

Description: This section will cover some best practices, tips, and caveats when configuring CIS labels in the Kubernetes Service. These labels are used for service discovery on the BIG-IP. For a full guide to these topics, please refer to the official documentation on F5 Cloud Docs.

Prerequisites: - Basic understanding REST APIs and declarative configuration.

3.2.1 Summary

CIS can dynamically discover and update the BIG-IP system's load balancing pool members using Service Discovery. CIS maps each pool definition in the AS3 template to a Kubernetes Service resource using Labels. To create this mapping, add the following labels to your Kubernetes Service:

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    run: nginx
    cis.f5.com/as3-tenant: stg_tenant
    cis.f5.com/as3-app: stg_app
    cis.f5.com/as3-pool: stg_nginx_pool
  name: nginx
  namespace: default
spec:
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 80
  selector:
    run: nginx
  type: NodePort
```

Label	Description
app: <string>	This label associates the service with the deployment. *Important: This label must be included, and resolve in DNS.*
cis.f5.com/as3-tenant: <string>	The name of the **partition** in your AS3 declaration. *Important: The string must not use a hyphen (-) character.*
cis.f5.com/as3-app: <string>	The name of the **class** in your AS3 declaration.
cis.f5.com/as3-pool: <string>	The name of the **pool** in your AS3 Declaration.

3.3 IPv6 Support in CIS and F5 IPAM Controller

Description: This section will cover some best practices, tips, and caveats when configuring CIS and F5 IPAM Controller with manifest files using IPv6 addresses.

3.3.1 Summary

BIG-IP itself supports IPv6 addresses for all objects. As of CIS v2.6.0, you can specify an IPv6 address in the deployment manifest that determines the BIG-IP address to connect to in the `bigip_url` parameter. The new configuration parameter for the deployment manifest is called `enable-ipv6` , which also enables use of IPv6 in custom resources such as VirtualServer, TransportServer and ServiceTypeLB service.

Example for BIG-IP URL in the CIS deployment manifest:

```
"--bigip-url=[2400:c:1:c:0:0:0:114]",
```

Example for VirtualServer CR:

```
apiVersion: "cis.f5.com/v1"
kind: VirtualServer
metadata:
name: cafe-virtual-server
labels:
    f5cr: "true"
spec:
# This is an insecure virtual, Please use TLSProfile to secure the virtual
# check out tls examples to understand more.
host: cafe.example.com
virtualServerAddress: "2002:0:0:0:10:0:0:2"
virtualServerName: "cafe-virtual-server"
pools:
- path: /coffee
    service: svc-2
    servicePort: 80
```

F5 IPAM Controller v0.1.6 and later also supports configuration of IPv6 address ranges.

Example for IPv6 ip-range:

```
--ip-range='{"Test-v4":"10.192.75.113-10.192.75.116","Prod-v4":"10.192.125.
↪30-10.192.125.50","Prod-v6":"2001:db8:5::ffff-2001:db8:6::9"}'
```

For earlier version of CIS, the below workaround is available for the big-ip url:

3.3.2 Workaround

The current workaround is to use `hostAliases` to define a hostname that resolves to the IPv6 address. See below snippet for a sample configuration:

```
apiVersion: apps/v1
kind: Deployment
metadata:
name: k8s-bigip-ctlr-deployment
namespace: kube-system
spec:
```

(continues on next page)

(continued from previous page)

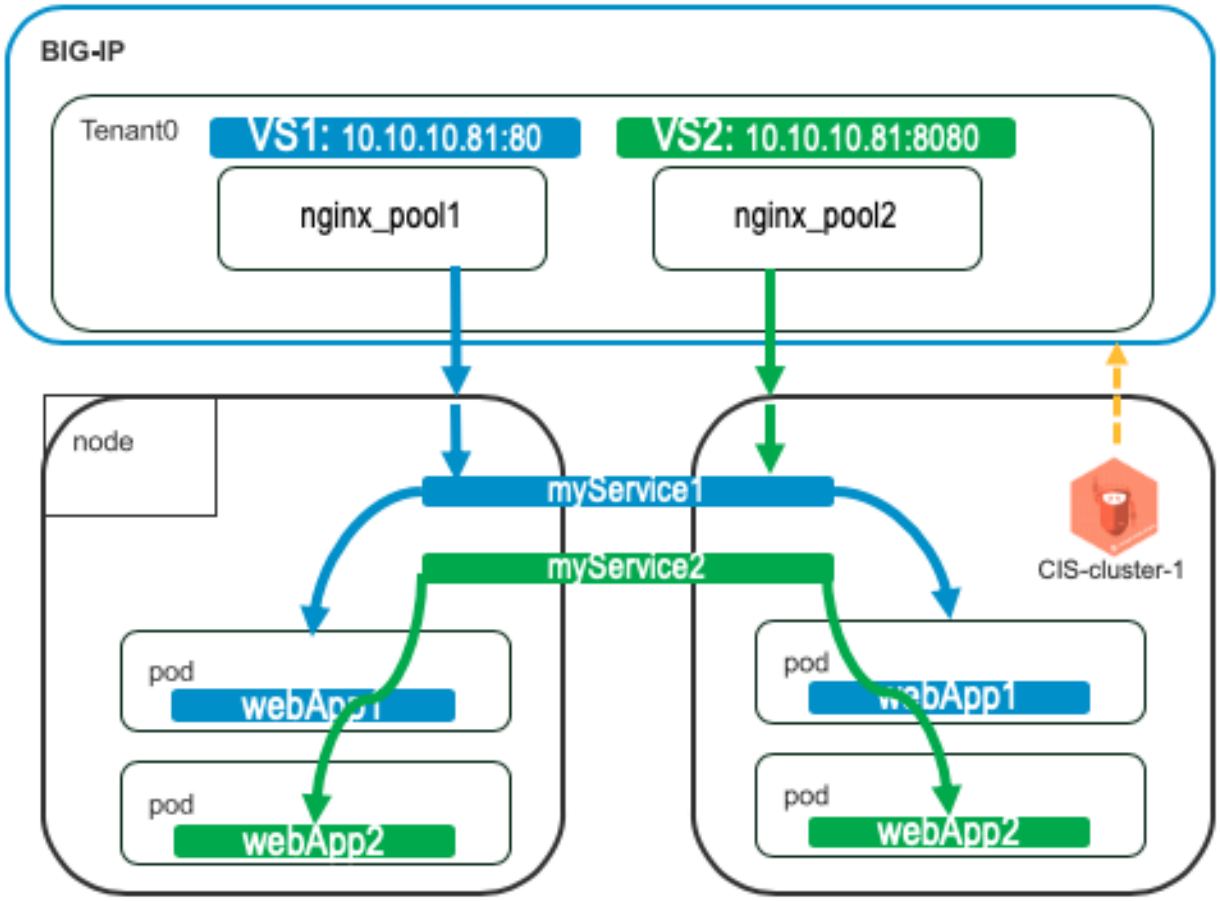
```
# DO NOT INCREASE REPLICA COUNT
selector:
    matchLabels:
    app: k8s-bigip-ctlr
replicas: 1
template:
    metadata:
    name: k8s-bigip-ctlr
    labels:
        app: k8s-bigip-ctlr
    spec:
    # Name of the Service Account bound to a Cluster Role with the required
    # permissions
    hostAliases:
        - ip: "240b:ab11:cd22:a101::10"
        hostnames:
        - "stagingbigip1"
    serviceAccountName: bigip-ctlr
```

3.4 AS3 Configmap: Multiple Apps in Single Partition - Single Virtual Address

Description: This section will cover some best practices, tips, and caveats when configuring multiple apps (virtual servers) in the AS3 declaration. In this scenario, an application owner wants to configure multiple applications that may use different protocols. The tenant/partition will be the same. For more examples, see F5 DevCentral f5-k8s-demo repository.

Prerequisites: - Basic understanding REST APIs and declarative configuration.

Diagrams:



3.4.1 Summary

You can declare multiple applications (virtual servers) in a single partition/tenant.

1. Define one tenant
2. Define one virtual address in a Shared application block
3. Define first application in the tenant block referencing above virtual address using `"use"`
4. Similarly, define second application in the same tenant block

```
kind: ConfigMap
apiVersion: v1
metadata:
name: f5-as3-declaration1
namespace: kube-system
labels:
    f5type: virtual-server
    as3: "true"
data:
template: |
    {
        "class": "AS3",
        "declaration": {
```

(continues on next page)

(continued from previous page)

```
            "class": "ADC",
            "schemaVersion": "3.18.0",
            "id": "testapp",
            "label": "Two HTTP VS with one VIP",
            "remark": "example",
            "Tenant0": {
                "class": "Tenant",
                "Shared": {
                    "class": "Application_Shared",
                    "template":"shared",
                    "serviceAddress0": {
                        "class": "Service_Address",
                        "virtualAddress": "10.1.10.81",
                        "trafficGroup": "/Common/traffic-group-1"
                    }
                },
                "App1": {
                    "class": "Application",
                    "template": "http",
                    "serviceMain": {
                        "class": "Service_HTTP",
                        "virtualAddresses": [{ "use": "/Tenant0/Shared/
↪serviceAddress0"}],
                        "virtualPort": 80,
                        "pool": "nginx_pool1"
                    },
                    "nginx_pool1": {
                        "class": "Pool",
                        "monitors": [
                            "http"
                        ],
                        "members": [
                            {
                                "servicePort": 80,
                                "serverAddresses": [],
                                "shareNodes": true
                            }
                        ]
                    }
                },
                "App2": {
                    "class": "Application",
                    "template": "http",
                    "serviceMain": {
                        "class": "Service_HTTP",
                        "trafficGroup": "traffic-group-1",
                        "virtualAddresses": [{ "use": "/Tenant0/Shared/
↪serviceAddress0"}],
                        "virtualPort": 8080,
                        "pool": "nginx_pool2"
                    },
                    "nginx_pool2": {
                        "class": "Pool",
                        "monitors": [
                            "http"
                        ],
```

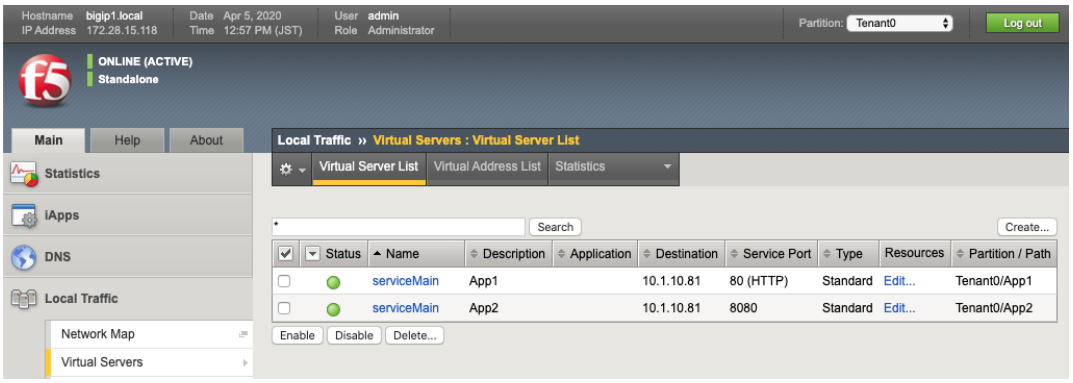
(continues on next page)

(continued from previous page)

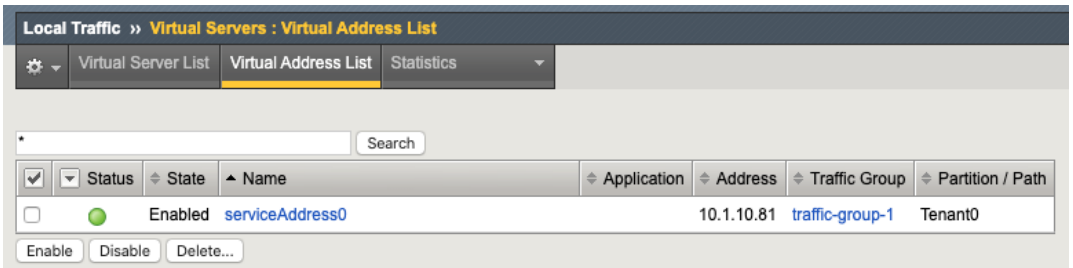
```
                        "members": [
                            {
                                "servicePort": 80,
                                "serverAddresses": [],
                                "shareNodes": true
                            }
                        ]
                    }
                }
            }
        }
    }
```

Confirm BIG-IP Objects:

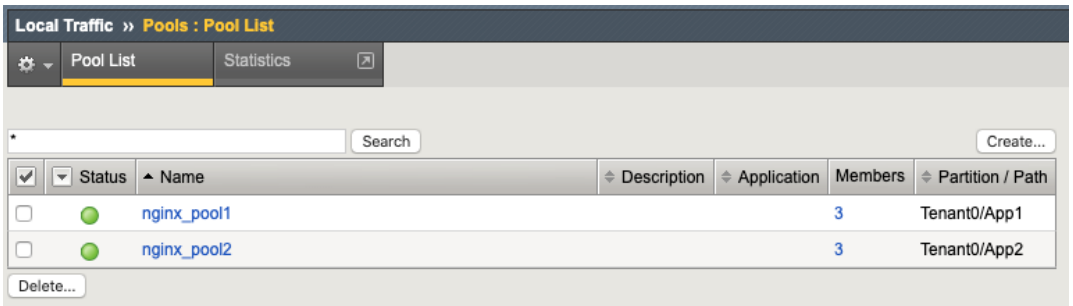
- Two Virtual Servers listening on different ports in the same partition (Tenant0)



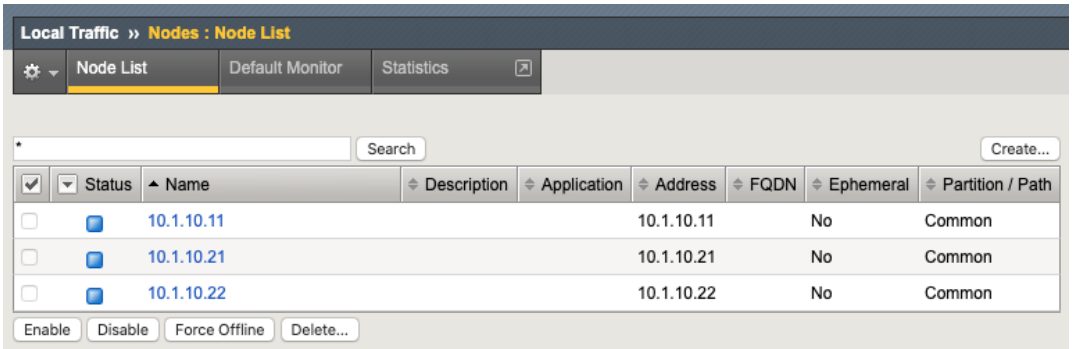
- One Virtual IP in traffic-group-1



- Two pools associated with each Virtual Server



- The nodes (pool member IPs) are automatically discovered and placed in the `/Common` partition

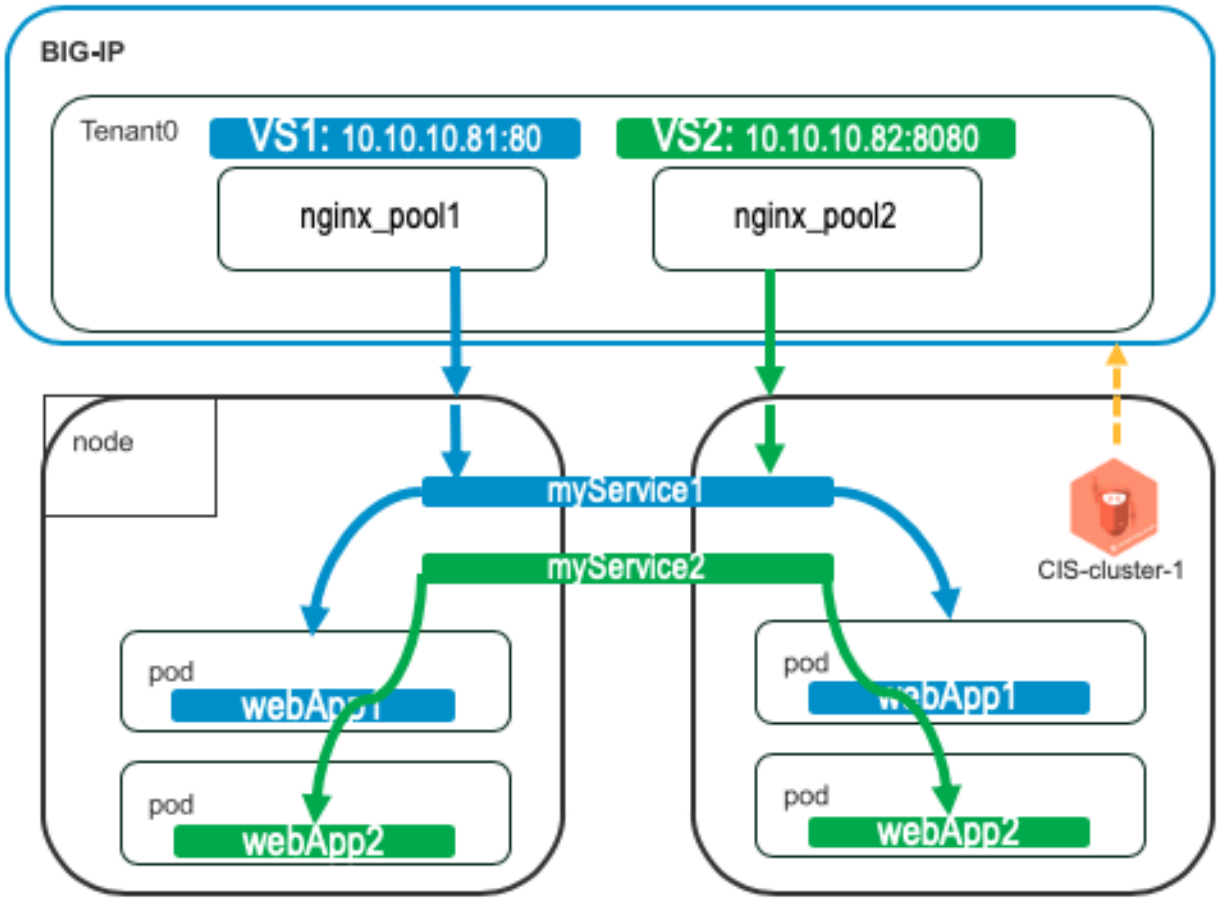


3.5 AS3 Configmap: Multiple Apps in Single Partition - Multiple Virtual Addresses

Description: This section will cover some best practices, tips, and caveats when configuring multiple apps (virtual servers) in the AS3 declaration. In this scenario, an application owner wants to configure multiple applications that may use different protocols and virtual IPs. The tenant/partition will be the same. For more examples, see F5 DevCentral f5-k8s-demo repository.

Prerequisites: - Basic understanding REST APIs and declarative configuration.

Diagram:



3.5.1 Summary

You can declare multiple applications (virtual servers) in a single partition/tenant.

1. Define one tenant
2. Define first application in the tenant block with one virtual address
3. Similarly, define second application with its own virtual address in the same tenant block

```
kind: ConfigMap
apiVersion: v1
metadata:
name: f5-as3-declaration1
namespace: kube-system
labels:
    f5type: virtual-server
    as3: "true"
data:
template: |
    {
        "class": "AS3",
        "declaration": {
            "class": "ADC",
            "schemaVersion": "3.18.0",
            "id": "testapp",
            "label": "Two HTTP VS with one VIP",
            "remark": "example",
            "Tenant0": {
                "class": "Tenant",
                "App1": {
                    "class": "Application",
                    "template": "http",
                    "serviceAddress0": {
                        "class": "Service_Address",
                        "virtualAddress": "10.1.10.81",
                        "trafficGroup": "/Common/traffic-group-1"
                    },
                    "serviceMain": {
                        "class": "Service_HTTP",
                        "virtualAddresses": [{ "use": "/Tenant0/App1/serviceAddress0
↪"}],
                        "virtualPort": 80,
                        "pool": "nginx_pool1"
                    },
                    "nginx_pool1": {
                        "class": "Pool",
                        "monitors": [
                            "http"
                        ],
                        "members": [
                            {
                                "servicePort": 80,
                                "serverAddresses": [],
                                "shareNodes": true
                            }
                        ]
                    }
                },
```

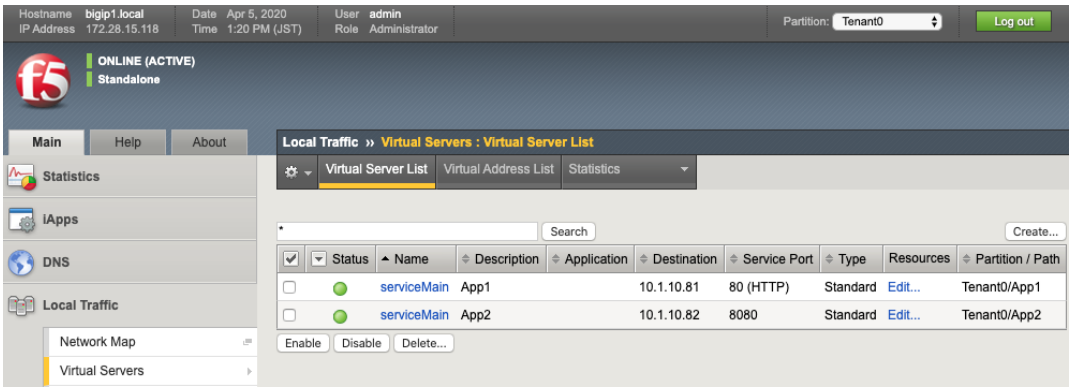
(continues on next page)

(continued from previous page)

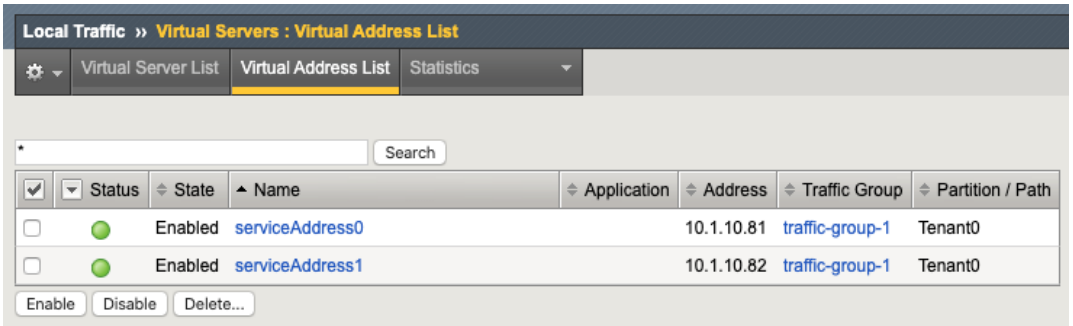
```
                "App2": {
                    "class": "Application",
                    "template": "http",
                    "serviceAddress1": {
                        "class": "Service_Address",
                        "virtualAddress": "10.1.10.82",
                        "trafficGroup": "/Common/traffic-group-1"
                    },
                    "serviceMain": {
                        "class": "Service_HTTP",
                        "trafficGroup": "traffic-group-1",
                        "virtualAddresses": [{ "use": "/Tenant0/App2/serviceAddress1
↪"}],
                        "virtualPort": 8080,
                        "pool": "nginx_pool2"
                    },
                    "nginx_pool2": {
                        "class": "Pool",
                        "monitors": [
                            "http"
                        ],
                        "members": [
                            {
                                "servicePort": 80,
                                "serverAddresses": [],
                                "shareNodes": true
                            }
                        ]
                    }
                }
            }
        }
    }
```

Confirm BIG-IP Objects:

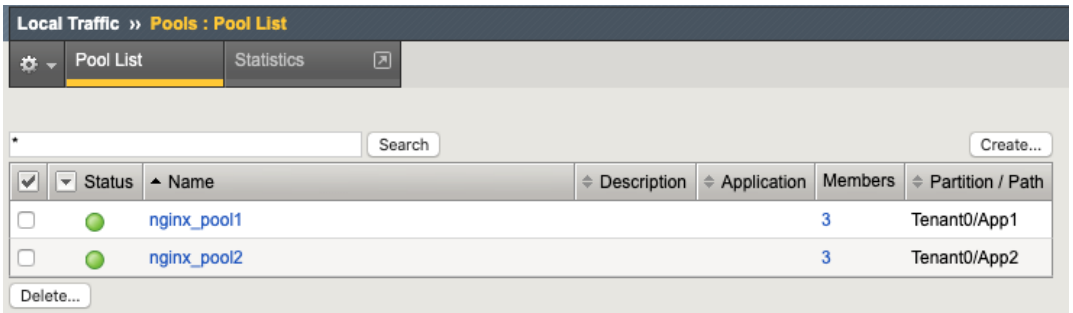
- Two Virtual Servers listening on different ports in the same partition (Tenant0), with different Virtual IPs



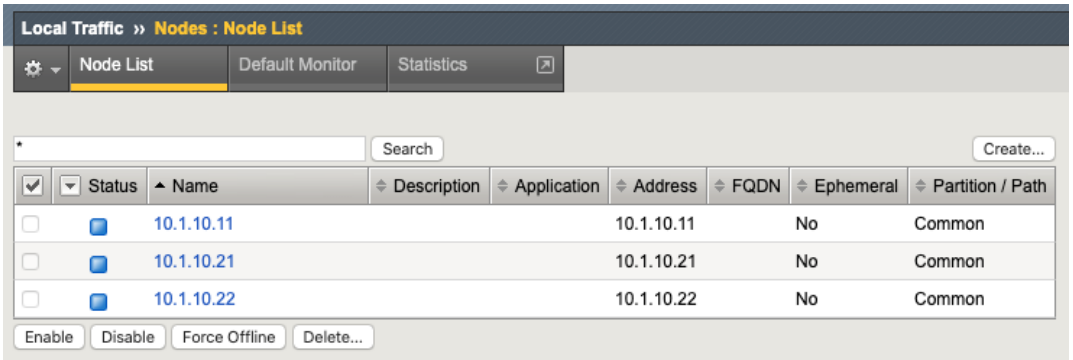
- Two Virtual IPs in traffic-group-1



- Two pools associated with each Virtual Server



- The nodes (pool member IPs) are automatically discovered and placed in the `/Common` partition



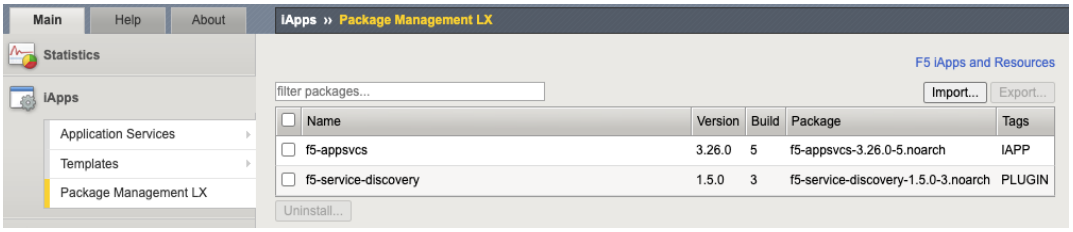
3.6 CIS New Installation

Description: Below are quick instructions for environment setup and installation of F5 CIS and F5 BIG-IP for load balancing and securing traffic into your Kubernetes cluster. For a full guide to these topics, please refer to the official documentation on Clouddocs .

3.6.1 BIG-IP Configuration

Confirm below requirements for deploying CIS

1. AS3 is an extension that allows configuration via declarative API. Version 3.18+ must be installed on your BIG-IP system. Download the rpm file from F5 Networks GitHub. Login to BIG-IP GUI > iApps > Package Management LX > Click "Import. . . ".



2. Create a BIG-IP partition to manage Kubernetes objects. This partition can be created either via the GUI (System > Users > Partition List) or via our TMOS CLI:

```
create auth partition <cis_managed_partition>
```

3. Create a user on BIG-IP with admin access. Login to BIG-IP GUI > System > Users > Click "Create. . . " > Set password > For Partition Access, select Administrator for Role > Click "Add" > Click "Finished".

4. For Cluster mode integration using BGP, confirm that you have the "Routing Bundle" that enables use of zebos.

```
[admin@ip-10-1-1-5:Active:Standalone] ~ # tmsh show sys license | grep -i␣
↪routing
    Routing Bundle, VE

[admin@ip-10-1-1-5:Active:Standalone] ~ # zebos check
=== route domain: 0 ===
nsm is running [23183]
imi is running [23182]
bgpd is running [23184]
```

3.6.2 BIG-IP Optimizations (Optional)

If you have a large cluster or expect to have many Kubernetes tenants, namespaces and app workloads that need to be exposed on BIG-IP, make below optimizations on BIG-IP to support the increased control plane calls.

1. Log into BIG-IP as administrator and make below changes:

```
# https://my.f5.com/manage/s/article/K13996055

tmsh modify sys db provision.extramb value 2048

tmsh modify sys db restjavad.useextramb value true

tmsh modify sys db provision.restjavad.extramb value 1600

tmsh save sys config

curl -s -f -u admin -H "Content-Type: application/json" -d '{
↪"maxMessageBodySize":134217728}' -X POST http://localhost:8100/mgmt/
↪shared/server/messaging/settings/8100

# https://my.f5.com/manage/s/article/K52650034
sysctl -w net.netfilter.nf_conntrack_tcp_timeout_time_wait=30
```

3.6.3 CIS Installation in Kubernetes

1. Create secret used for authentication to F5 BIG-IP from CIS.

```
kubectl create secret generic bigip-login -n kube-system --from-
↪literal=username=cis_usr --from-literal=password=XXX
```

2. Create service account

```
kubectl create serviceaccount bigip-ctlr -n kube-system
```

3. Create cluster role binding for the service account

```
kubectl create clusterrolebinding k8s-bigip-ctlr-clusteradmin --
↪clusterrole=cluster-admin --serviceaccount=kube-system:bigip-ctlr
```

4. Create service account

```
kubectl create serviceaccount bigip-ctlr -n kube-system
```

5. (Optional) Create secret containing repository credentials. Modify registry as needed

```
kubectl create secret generic f5-docker-images --from-file=.
↪dockerconfigjson=/root/.docker/config.json --type=kubernetes.io/
↪dockerconfigjson -n kube-system
```

6. Define deployment for CIS. Note: Below manifest will use Cluster mode, watch all namespaces (default), output debug messages (can change to info as needed), work for IPv4 only environment, and pulls the latest image from docker (you can change to your local registry).

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: k8s-bigip-ctlr-deployment
  namespace: kube-system
spec:
  replicas: 1
  selector:
    matchLabels:
      app: k8s-bigip-ctlr
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: k8s-bigip-ctlr
      name: k8s-bigip-ctlr
    spec:
      serviceAccountName: bigip-ctlr
      containers:
        - args:
            - --bigip-username=$(BIGIP_USERNAME)
            - --bigip-password=$(BIGIP_PASSWORD)
            - --bigip-url=<ip_address-or-hostname>
            - --bigip-partition=k8s
            - --pool-member-type=cluster
            - --insecure=true
```

(continues on next page)

(continued from previous page)

```
            - --agent=as3
            - --log-level=info
            - --custom-resource-mode=true
            - --log-as3-response=true
            - --as3-validation=true
          command:
            - /app/bin/k8s-bigip-ctlr
          env:
            - name: BIGIP_USERNAME
              valueFrom:
                secretKeyRef:
                  key: username
                  name: bigip-login
            - name: BIGIP_PASSWORD
              valueFrom:
                secretKeyRef:
                  key: password
                  name: bigip-login
          image: f5networks/k8s-bigip-ctlr:latest
          imagePullPolicy: IfNotPresent
          name: k8s-bigip-ctlr
      dnsPolicy: ClusterFirst
      #imagePullSecrets:
      #  - name: f5-docker-images
```

7. Apply deployment manifest file for CIS. This will create a pod in kube-system namespace with replica of "1".

```
kubectl apply -f f5-cis-deployment.yaml
```

8. Confirm logs of CIS if any issues

```
kubectl logs <k8s-bigip-ctlr-###> -n kube-system
```

9. Install the F5 CIS CRDs

```
kubectl apply -f https://raw.githubusercontent.com/F5Networks/k8s-bigip-
↪ctlr/master/docs/config_examples/customResourceDefinitions/
↪customresourcedefinitions.yml
```

3.6.4 Expose Application using F5 CIS

1. Create a VirtualServer custom resource (CR). The service parameter is the name of your application service that is of type "ClusterIP"

```
apiVersion: "cis.f5.com/v1"
kind: VirtualServer
metadata:
  name: f5-demo-mysite
  labels:
    f5cr: "true"
spec:
  host: mysite.f5demo.com
  virtualServerAddress: "10.192.75.113"
```

(continues on next page)

(continued from previous page)

```
  pools:
  - monitor:
      interval: 20
      recv: ""
      send: /
      timeout: 31
      type: http
    path: /
    service: f5-demo
    servicePort: 80
```

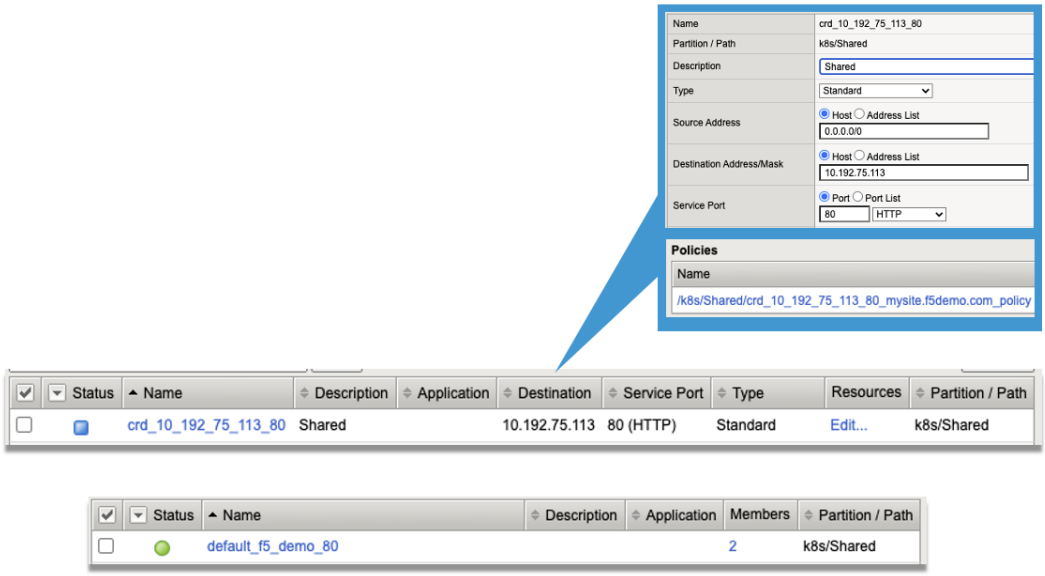
2. Apply the VirtualServer CR. This will create a pod in kube-system namespace with replica of “1”.

```
kubectl apply -f vs-mysite-test.yaml
```

3. Confirm the VirtualServer objects

```
kubectl get vs
```

4. Confirm objects on BIG-IP



5. Confirm access from client to the exposed IP Address on BIG-IP.

4 CIS Troubleshooting

Description: First, refer to the below official documentation on F5 Cloud Docs for a comprehensive list of troubleshooting steps:

https://clouddocs.f5.com/containers/v2/troubleshooting/kubernetes.html

This following section will cover **additional** best practices, tips, and caveats when troubleshooting F5 Container Ingress Services (CIS) in a Kubernetes environment.

4.1 CIS (Controller) Additional Troubleshooting

- AS3 Log level: A new command-line option was introduced in Controller v1.12 that logs the AS3 error response. Add below to the arguments block in the Controller deployment manifest file. The logs will appear in the Controller pod logs

```
"--log-as3-response=true",
```

- BIG-IP restnoded logs: There are two key daemons responsible for iControl REST and iControl LX Extensions. The two logs for these daemons are:

```
/var/log/restjavad.0.log
```

```
/var/log/restnoded/restnoded.log
```

- AS3 Declaration Validator: If you are using Microsoft Visual Studio Code, you can follow instructions below to install a validator for your declaration. This will identify syntax errors and do auto tab complete. Make sure to save the file first as .json extension.

https://clouddocs.f5.com/products/extensions/f5-appsvcs-extension/latest/userguide/validate.html

WE MAKE APPS GO

FASTER.
SMARTER.
SAFER.

F5 Networks, Inc. | f5.com

US Headquarters: 401 Elliott Ave W, Seattle, WA 98119 | 888-882-4447 // Americas: info@f5.com // Asia-Pacific: apacinfo@f5.com // Europe/Middle East/Africa: emeainfo@f5.com // Japan: f5j-info@f5.com