CNF

Release 202202152222

Collaborators

May 28, 2022

CONTENTS

CHAPTER

ONE

LAB ACCESS

CHAPTER

TWO

DEPLOYMENT

2.1 Namespace

Namespaces provide a mechanism for isolating groups of resources within a single cluster. Create a namespace and make it the default context for all kubectl commands.

1. Using kubectl, create the namespace.

```
$ kubectl create ns my-cnf
```

2. Confirm that the namespace has been created.

```
$ kubectl get namespaces
```

3. View the pods in the new namespace, by adding the *–namespace* argument.

```
$ kubectl get pods --namespace my-cnf
```

4. Permanently set the namespace for all subsequent kubectl commands in the *my-cnf* context.

```
$ kubectl config set-context --current --namespace my-cnf

$ kubectl get pods
```

To monitor the Kubernetes cluster, we will use the k9s tool. Open a new SSH window to the Mgmt Jumphost and launch k9s k9s

2.2 Resources

Dependancy resources are required to be created for the F5 CNF.

1. Create two interfaces, corresponding to the subscriber side (ens6) and internet side (ens7).

```
$ kubectl apply -f cni-grpc/cni-macvlan-bridge-ens6-dualstack.yaml
$ kubectl apply -f cni-grpc/cni-macvlan-bridge-ens7-dualstack.yaml
```

2. Create the secrets for the CNF.

```
$ kubectl -n my-cnf apply -f cni-grpc/certs-secret.yaml
$ kubectl -n my-cnf apply -f cni-grpc/keys-secret.yaml
```

(continues on next page)

(continued from previous page)

```
$ kubectl -n my-cnf apply -f cni-grpc/dssm-certs-secret.yaml
$ kubectl -n my-cnf apply -f cni-grpc/dssm-keys-secret.yaml
```

3. Install the custom resources.

```
$ helm install commoncrd --version 0.6.4 crd/f5-spk-crds-common-0.6.4.tgz
$ kubectl apply -f cni-grpc/cnf-vlan-crd.yaml
$ helm install gilancrd --version 0.6.4 crd/f5-spk-crds-n6-gilan-0.6.4.tgz
$ helm install service-proxy --version 0.6.4 crd/f5-spk-crds-service-proxy-0.6.4.tgz
```

2.3 Helm

F5 CNFs support the helm package manager. Create a deplpoyment using a helm chart along with a values file which contains implementation specific details.

1. Install the User Plane CNF

```
$ helm install --values f5-cnf/cnf_1tmm.yaml f5-ctrl f5-cnf/f5ingress-0.90.7.tgz
```

This command creates a new kubernetes pod, named "f5-ctrl" using the Helm chart defined in the tarball (.tgz) file. The deployment can be customised using values from an appropriate YAML configuration file.

Observe the deployment within the SSH session running k9s. A CNF is deployed with several Pods. Note the time for all pods to reach the "running" state.

2.4 Running Config

The kubernetes running config may be viewed and modified.

1. View all pods and services.

```
$ kubectl get pod,svc -o wide
```

   A Kubernetes service is an abstraction of a group of Pods which run the same function. Within a service, a "Selector" is defined as shown in the output above. The service subsequently uses as search criteria to match any Pod which has that label.

2. View the labels assigned to a pod.

```
$ kubectl describe <pod name>
```

Use k9s to inspect the configuration. Use 'd' to see TMM IPs Selected K9s commands (see https://k9scli.io/topics/commands/ for full details) Arrow keys - up down Enter - Select ESC - back d - view configuration (for pod) s - start shell (for container) CNTL-D - exit shell CNTL-C - quit K9s Inspect CNF routing Configuration Routing functions in dataplane pod: f5-tmm-routing is ZebOS instance (dyanmic routing control plane) f5-tmm-tmrouted updates TMM with routes from f5-tmm-routing

Access the Mgmt Jumphost instance running k9s (or launch a new SSH window and run k9s) Use the arrow keys and [ENTER] to select the f5-tmm pod. Use the arrows keys select the f5-tmm-routing acontainer and press [s] to start shell View the routing configuration in Start shell for f5-tmm-routing and view routing configurationusing the following commands imish enable show running-config exit exit exit

2.5 Control Plane

F5's Cloud-Native Engine uses the Distributed Session State Manager (DSSM) to store state for BIG-IP CNFs. The DSSM can be deployed using a Helm Chart.

1. Deploy DSSM HELM Chart

```
$ helm install f5-dssm f5-dssm/f5-dssm-0.11.0.tgz --values f5-dssm/values-f5-dssm.
↪yaml
```

2. DSSM is deployed as a distributed database across 3 pods. Start shell for f5-dssm-db-0 (using K9s, in the same way as done in the previous)
3. DSSM uses a Redis database. You can use redis-cli to work with this database directly. For example, enter redis-cli info to get some basic information.

Deploy F5 Cloud Native Network Functions

CGNAT container running in k8s

CGNAT runs as a container in k8s platforms.

CHAPTER

THREE

POLICIES

3.1 View NAT44

Typical deployments use NAT44 to map private UE addressing to Public IPs. This demo uses NAPT translation – other translation techniques (e.g. PBA) will be shown in future demos.

No Connectivity !

"ping" and other network diagnostics tools will fail until CGNAT policies are activated.

1. Configure UE to use only IPv4

   SSH to Subscriber Client (UE) and configure UE to use only IPv4 on ens6 traffic interface.

   ```
   $ ~ubuntu/ipv4-only-interface.sh
   $ ip -4 a (or ip addr show ens6)
   ```

2. Check the status of connectivity between UE and the test Web Server

   ```
   $ dig web1.cnf.local
   $ ping web1.cnf.local
   $ curl web1.cnf.local
   ```

   Connect via RDP to the Subscriber Client (UE) Desktop to test connectivity using a Web Browser.
   Note: The Subscriber Client credentials are ubuntu/passw0rd

3.2 Apply CGN Config

Apply a bunch of policies.

Note!

If you open a new Web Shell connection, remember to change to the right directory

https://youtu.be/7riGQIJRJgI

1. Connect to the Mgmt Jumphost.

```
$ su - ubuntu
$ cd ~/my-cnf
```

2. Apply the following configuration

```
$ kubectl apply -f cgn-policy/01-hsl-elk.yaml
$ kubectl apply -f cgn-policy/02-seclog-profile.yaml
$ kubectl apply -f cgn-policy/03-cnf-calico-pod-net-static-route.yaml
$ kubectl apply -f cgn-policy/04-cnf-ue-static-route.yaml
```

3. NAT 64

```
$ kubectl apply -f cgn-policy/61-natpolicy1-nat64.yaml
$ kubectl apply -f cgn-policy/62-vs-securefastl4-nat64-http.yaml
$ kubectl apply -f cgn-policy/63-vs-securefastl4-nat64-https.yaml
$ kubectl apply -f cgn-policy/99-cnf-static-route-default-gw.yaml
```

4. NAT 44

```
$ kubectl apply -f cgn-policy/41-natpolicy2-nat44.yaml
$ kubectl apply -f cgn-policy/42-vs-securefastl4-nat44-http.yaml
$ kubectl apply -f cgn-policy/43-vs-securefastl4-nat44-https.yaml
```

5. Test connectivity from the Subscriber Client (UE)

```
$ ping web1.cnf.local
$ curl web1.cnf.local
```

   At this point, "curl" should return HTML content. This includes details of the IP addressing.

6. View the applied NAT44 policy, and verify the translation is performed correctly.

```
$ cat cgn-policy/41-natpolicy2-nat44.yaml
  apiVersion: "k8s.f5net.com/v1"
  kind: F5SPKNatPolicy
  metadata:
    name: "natpolicy2"
  spec:
    sourceTranslation:
      - name: "srctr2"
        type: "dynamic-pat"
        addresses:
          - "10.10.10.128/25"
        port: "5000-5100"
    rule:
      - name: rule2
        ipProtocol: tcp
        sourceAddresses:
          - "10.1.30.16"
        destinationAddresses: []
        destinationPorts: []
        sourceTranslation: "srctr2"
```

3.3 Update NAT44 Policy

3.4 Control Plane CNF

F5's Cloud-Native Engine uses the Distributed Session State Manager (DSSM) to store state for BIG-IP CNFs. The DSSM can be deployed using a Helm Chart. Deploy DSSM HELM Chart: helm install f5-dssm f5-dssm/f5-dssm-0.11.0.tgz –values f5-dssm/values-f5-dssm.yaml

DSSM is deployed as a distributed database across 3 pods. Start shell for f5-dssm-db-0 (using K9s, in the same way as done in the previous) DSSM uses a Redis database. You can use redis-cli to work with this database directly. For example, enter redis-cli info to get some basic information.

STOP! MODULE 4 COMPLETE. Challenge Answer is hot-mouse

3.5 Debug

View Virtual Server statistics directly from CLI. We do this by using kubectl to execute a command directly on the debug container in the tmm pod. kubectl exec -it deployment/f5-tmm -c debug -n my-cnf – tmctl -f /var/tmstat/blade/tmm0 virtual_server_stat -s name,serverside.tot_conns

Use the 'debug' scripts to run multiple kubectl exec commands below: ./debug-tmctl.sh to view multiple statistics ./debug-bdt-cli.sh to see connection table ./debug-config-viewer.sh to view configuration

Use can inspect the scripts with cat to see the kubectl exec commands being used.

3.6 Demonstrate NAT64

For deployment with IPv6-only clients, we can use NAT64. Configure Subscriber Client (UE) to use only IPv6 SSH to Subscriber Client and configure to use only IPv6 on ens6 traffic interface. ~ubuntu/ipv6-only-interface.sh ip -6 a dig AAAA web1.cnf.local curl -6 web1.cnf.local

3.7 Cleanup

Remove the configuration using the commands below kubectl delete -f cgn-policy/01-hsl-elk.yaml kubectl delete -f cgn-policy/02-seclog-profile.yaml kubectl delete -f cgn-policy/61-natpolicy1-nat64.yaml kubectl delete -f cgn-policy/62-vs-securefastl4-nat64-http.yaml kubectl delete -f cgn-policy/63-vs-securefastl4-nat64-https.yaml

kubectl delete -f cgn-policy/41-natpolicy2-nat44.yaml kubectl delete -f cgn-policy/42-vs-securefastl4-nat44-http.yaml kubectl delete -f cgn-policy/43-vs-securefastl4-nat44-https.yaml Verify the policy has been removed (curl no longer works)

3.8 Deploy Firewall Policies

Apply the configuration using the commands below kubectl apply -f fw-policy/01-acl-hsl.yaml kubectl apply -f fw-policy/02-acl-log-profile.yaml kubectl apply -f fw-policy/03-acl-rules.yaml

kubectl apply -f cgn-policy/61-natpolicy1-nat64.yaml

Apply FW rules to NAT 64 Rules kubectl apply -f fw-policy/62-vs-securefastl4-nat64-http-with-fw.yaml kubectl apply -f fw-policy/63-vs-securefastl4-nat64-https-with-fw.yaml View the various FW policies to see how they are defined by K8s CRDs: cat fw-policy/03-acl-rules.yaml cat fw-policy/62-vs-securefastl4-nat64-http-with-fw.yaml

3.9 Remove Firewall Policies

Remove the configuration using the commands below # Remove Firewall rules kubectl delete -f fw-policy/03-acl-rules.yaml kubectl delete -f fw-policy/02-acl-log-profile.yaml kubectl delete -f fw-policy/01-acl-hsl.yaml

kubectl delete -f cgn-policy/61-natpolicy1-nat64.yaml

Remove FW rules to NAT 64 Rules kubectl delete -f fw-policy/62-vs-securefastl4-nat64-http-with-fw.yaml kubectl delete -f fw-policy/63-vs-securefastl4-nat64-https-with-fw.yaml

3.10 Uninstall HELM Charts

SSH to JumpHost and run: helm uninstall f5-dssm -n my-cnf helm uninstall f5-ctrl -n my-cnf

Verify termination of Pods (e.g. using k9s, k9s -n my-cnf)

STOP! MODULE 5 COMPLETE. Challenge Answer is brown-cow

Configure CGNAT and Firewall Policies

CHAPTER

FOUR

CI/CD WITH ARGO

4.1 Create GIT Repo

In UDF, connect directly to the GitLab UI. Login using root/passw0rd

Graphical user interface, application

Description automatically generated

In GitLab, Create a project Graphical user interface, text, application, Teams

Description automatically generated

Then, Create blank project Enter Project Name my-cnf Select Visibility Level Public Click Create project

Graphical user interface, text, application, email

Description automatically generated

On the SSH jumphost, configure Git: git config –global user.name "Administrator" git config –global user.email "admin@example.com" Create a new Git repository and push the contents to the Gitlab server

git init git remote add origin git@gitlab.f5.local:root/my-cnf.git git add . git commit -m "Initial commit" git push -u origin master

Return to the GitLab UI, an verify all files have been pushed into the repository.

Graphical user interface, application

Description automatically generated

4.2 Demonstrate CI/CD NAT44 Policy Update

Re-configure Subscriber Client (UE) to use only IPv4 Re-configure Subscriber Client (UE) to use only IPv4 on ens6 traffic interface ~ubuntu/ipv4-* ip -4 a dig web1.cnf.local curl web1.cnf.local

At this point, "curl" should fail as there is no CGNAT policy deployed. TIP: You can leaving curl running continuously (every 2 seconds) using the watch command. watch curl -s http://web1.cnf.local

Log into ArgoCD Graphical user interface, application

Description automatically generated

Alternatively, Log in via Subscriber Client browser GitLab http://10.1.1.15 root/passw0rd ArgoCD https://10.1.1.7:30422/login admin/passw0rd)

Configure ArgoCD

Select + New App

Graphical user interface, text, application, chat or text message

Description automatically generated

Set-up the following ArgoCD Application:

CGN Policy General Application Name: cgn-policy Project: cgn-policy Source Repository URL: https://gitlab.f5.local/root/my-cnf.git Path: cgn-policy Destination Cluster URL: https://10.1.1.4:6443 Namespace: my-cnf

The Argo application should deploy but will not be synchronized, as shown below. Graphical user interface, text, application, chat or text message

Description automatically generated Click on Sync, then click Synchronize in the new pane that is displayed to synchronise all settings between GitLab and the Kubernetes cluster. It should now be possible to connect from the Client to the Web Server. Explore the Argo UI. Click on cgn-policy to view the configuration and status. Modify NAT44 policy

In the Jumphost SSH session, edit the NAT44 policy (e.g. using vi), then push this to the Gitlab repository. vi ~/my-cnf/cgn-policy/41-natpolicy2-nat44.yaml edit the port range to 3500-3600 A picture containing text

Description automatically generated

git add ~/my-cnf/cgn-policy/41-natpolicy2-nat44.yaml git commit -m "updated nat44 policy in cli" git push origin master

ArgoCD can operate with either Manual or Automatic synchronisation. We are using manual synchronisation. Within the ArgoCD, synchronise as before and verify (using the Subscriber Client) that the change occurs.

4.3 Deploy CNF using ArgoCD

Add the DSSM application in ArgoCD DSSM Application General Application Name: f5-dssm Project: f5-dssm Sync Policy: <automatic> [Prune Resouces] – Selected [Self Heal] - Selected Source Repository URL: https://gitlab.f5.local/root/my-cnf.git Path: f5-dssm Destination Cluster URL: https://10.1.1.4:6443 Namespace: my-cnf Helm Values Files: values-f5-dssm.yaml

Verify using kubectl or k9s that the DSSM function is deployed automatically. Delete the DSSM application in ArgoCD and verify it terminates.

STOP! MODULE 6 COMPLETE. Challenge Answer is blue-kangaroo

Connect to the Mgmt Jumphost. Note! If you open a new Web Shell connection, remember to change to the right directory su – ubuntu cd ~/my-cnf

To ensure the UDF configuration is "clean" following the previous sections, run: cd ~/my-cnf ./99-clean-up.sh ./00-create-namespace.sh ./01-on-board-cnf.sh ./02-deploy-cnf.sh

Sidebar Title

Sidebar Subtitle

Sidebar TESTING - indented lines comprise the body of the sidebar, and are interpreted as body elements. Go to town on a horse. Or ride a scooter

F5 Cloud Native Functions

Unleash the power of Kubernetes on the world and your family

https://youtu.be/7riGQIJRJgI

Your Topic Title

Subsequent indented lines comprise the body of the topic, and are interpreted as body elements.

Fig. 1: Service Proxy for Kubernetes

1. Blue Items

   The first paragraph in an ordered list would be nice to have a gray or grey background.

   ```
   $ kill -9
   ```

   An orderedlistitem helps explain whats going on.

2. Red Items

   The first paragraph in an ordered list would be nice to have a gray background.

   ```
   $ kill -9 86
   ```

   An ordered list item helps explain whats going on.