Juniper Networks | Engineering Simplicity

cSRX Deployment Guide for AWS

Published
2021-07-08

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

cSRX Deployment Guide for AWS
Copyright © 2021 Juniper Networks, Inc. All rights reserved.

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at https://support.juniper.net/support/eula/. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

About This Guide

Use this guide to install and configure the cSRX Container Firewall in AWS using Elastic Kubernetes Service (EKS). This guide also includes basic cSRX container configuration and management procedures.

After completing the installation, management, and basic configuration procedures covered in this guide, refer to the Junos OS documentation for information about further security feature configuration.

1
CHAPTER

Overview

Understanding cSRX Deployment in AWS using Elastic Kubernetes Service (EKS)

SUMMARY

This topic provides you an overview of cSRX Kubernetes Orchestration in AWS Cloud using AWS Elastic Kubernetes Service (EKS).

IN THIS SECTION

Understanding cSRX with Kubernetes

The cSRX Container Firewall is a containerized version of the SRX Series Services Gateway with a low memory footprint. cSRX provides advanced security services, including content security, AppSecure, and unified threat management in the form of a container. By using a Docker container the cSRX can substantially reduce overhead as each container shares the Linux host's OS kernel. Regardless of how many containers a Linux server hosts, only one OS instance is in use. Also, because of the containers' lightweight quality, a server can host many more container instances than virtual machines (VMs), yielding tremendous improvements in utilization. With its small footprint and Docker as a container management system, the cSRX Container Firewall enables deployment of agile, high-density security service.

Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications. With K8s support, cSRX scales out in a cluster running as elastic firewall service with smaller footprint when compared to virtual machines. It groups containers that make up an application into logical units for easy management and discovery. cSRX running in K8s cluster provides advantages such as:

- Runs services with smaller footprint
- Enables faster scale out and scale in of cSRX
- Automated management and controlled workflow

K8s defines a set of building objects that collectively provide mechanisms that orchestrate containerized applications across a distributed cluster of nodes, based on system resources (CPU, memory, or other

custom metrics). K8s masks the complexity of managing a group of containers by providing REST APIs for the required functionalities.

A node refers to a logical unit in a cluster, such as a server, which can either be physical or virtual. In context of Kubernetes clusters, a node usually refers specifically to a worker node. Kubernetes nodes in a cluster are the machines that run the end user applications.

There are two type of nodes in a Kubernetes cluster, and each one runs a well-defined set of processes:

- head node: also called primary, or primary node, it is the head and brain that does all the thinking and makes all the decisions; all of the intelligence is located here.
- worker node: also called node, or minion, it's the hands and feet that conducts the workforce.

The nodes are controlled by the primary in most cases.

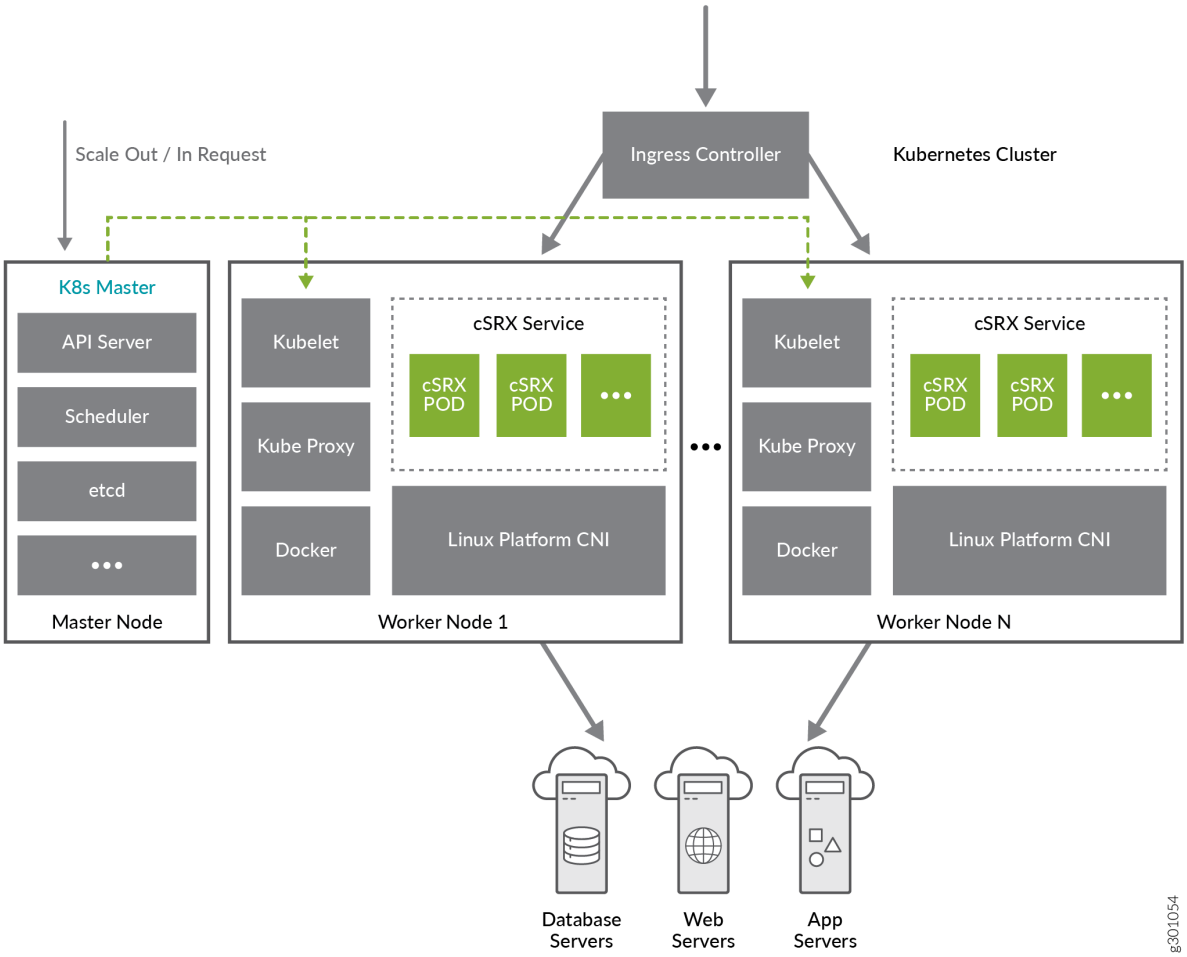
The interfaces between the cluster and you is the command-line tool kubectl. It is installed as a client application, either in the same primary node or in a separate machine.

Kubernetes's objects are:

- Pod
- Service
- Volume
- Namespace
- Replication
- Controller
- ReplicaSet
- Deployment
- StatefulSet
- DaemonSet
- Job

Figure 1 on page 4 illustrates cSRX service in Kubernetes.

Figure 1: cSRX Service in Kubernetes



In K8s deployment, you can use Multus with both Flannel and Weave CNI.

To support Kubernetes Node Port/Ingress controller with cSRX, environment variable CSRX_MGMT_PORT_REORDER allows cSRX to use container management interface as revenue interface. The Kubernetes Node Port/Ingress controller feature with cSRX is only supported with Flannel/Weave CNI. With CSRX_MGMT_PORT_REORDER set to "yes", you can explicitly control the re-configuration of the management port behavior. Like the access to cSRX shell or SD discovery on to the interface attached to cSRX using Multus CNI.

For example, if cSRX is brought up with eth0/eth1/eth2 with CSRX_MGMT_PORT_REORDER=yes, you can use eth2 as the new management interface.

NOTE: The traffic forwarding to this eth2 has to be done through the iptables rules defined explicitly by you.

See "Junos OS Features Supported on cSRX" on page 8 for a summary of the features supported on cSRX.

cSRX Kubernetes Orchestration in AWS Overview

AWS provides Managed Kubernetes (K8s), for short) services as part of their offerings. These managed services benefit you by reducing the dependencies on setting up and operation of the K8s environment. The orchestration and management of the cSRX in a K8s environment using the Multus CNI is already supported. With support for the K8s, you can now deploy, manage and orchestrate the cSRX along with other container workloads in their environment.

The cSRX Container Firewall protects your containerized environments with advanced security services, including content security, intrusion prevention system (IPS), AppSecure, and unified threat management (UTM).

Benefits:

- Automated service provisioning and orchestration
- Distributed and multi-tenancy traffic securing
- Scalable security services with small footprints

Currently, the orchestration and management of the cSRX in a Kubernetes (K8s) environment using the Multus CNI is supported. You can deploy cSRX as Kubernetes Service or Pods. With Kubernetes support, you can deploy, manage, and orchestrate, scale out and scale in cSRX in a cluster that provides an elastic firewall service to application containers along with other container workloads in AWS environment.

For more information, see cSRX Deployment Guide Kubernetes.

AWS provides managed K8s for short services as part of their offerings. With these managed services you can benefit by reducing dependencies on setting up and operation of the K8s environment. Customers also need to be provided with an option for deploying a containerized Firewall (cSRX) to secure their workloads in the public cloud on public cloud platform. While companies migrating to container workloads rely on K8s for management and orchestration of the containers, services provided by the AWS (and GCP and Microsoft Azure) are increasing in demand for their ease of use and low maintenance.

AWS provides two orchestration services for containers: **Elastic Container Service (ECS)** and **Elastic Kubernetes Service (EKS)**.

Elastic Kubernetes Service (EKS): This is a fully managed Kubernetes service. An open source Kubernetes adaptation and fully supports the open source version. EKS is Amazon managed service that helps in running Kubernetes application on AWS cloud. EKS helps in setting up Kubernetes control plane on multiple zones providing high-availability, EKS has the capability to detect and replace unhealthy control plane instances with automated version upgrades and patches as when required. EKS is fully integrated with Elastic Container Registry (ECR) which holds container images, Identity and Access Management (IAM) roles for authentication, AWS VPC for network isolation and Elastic Load Balancing for load distribution.

You can deploy and manage cSRX on the AWS cloud using Elastic Kubernetes Services (EKS) orchestration for cluster management with bring your own license (BYOL) licensing model.

Amazon Elastic Kubernetes Service

IN THIS SECTION

Amazon Elastic Kubernetes Service (Amazon EKS) gives you the flexibility to start, run, and scale Kubernetes applications in the AWS cloud or on-premises. Amazon EKS helps you provide highly-available and secure clusters and automates key tasks such as patching, node provisioning, and updates.

EKS runs upstream Kubernetes and is certified Kubernetes conformant for a predictable experience. You can easily migrate any standard Kubernetes application to EKS without needing to refactor your code.

EKS makes it easy to standardize operations across every environment. You can run fully managed EKS clusters on AWS. You can have an open source, proven distribution of Kubernetes wherever you want for consistent operations with Amazon EKS. You can host and operate your Kubernetes clusters on-premises and at the edge and have a consistent cluster management experience with Amazon EKS.

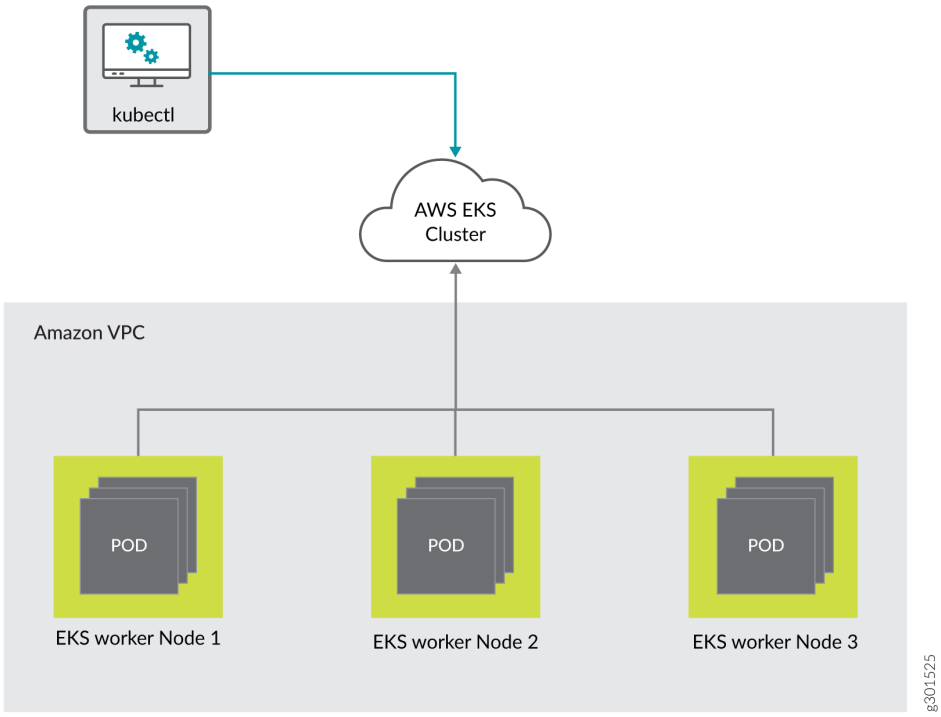
You can completely utilize the open-source Kubernetes functionality with their Elastic Kubernetes Solution (EKS) on AWS cloud. All latest Kubernetes updates are available on EKS framework.

cSRX is supported only on EKS with EC2 instances. EKS is fully integrated with Amazon cloud watch, Autoscaling groups, AWS Identity and Access Management (IAM) and Amazon Virtual Private Cloud (VPC) enabling seamless environment to monitor and load balance the cloud application.

AWS with EKS provides highly scalable control plane which will be running on two different zones to provide high availability support. EKS is completely compatible with open-source Kubernetes and any standard Kubernetes application can be easily migrated to EKS.

Figure 2 on page 7 illustrates AWS EKS abstraction architecture.

Figure 2: AWS EKS Abstraction Architecture



AWS proprietary Multus with flannel CNI is supported for EKS cluster deployments.

Benefits

The cSRX also integrates with other next-generation cloud orchestration tools such as Kubernetes.

The cSRX adds security enforcement points where none have existed before, offering the most comprehensive network security for Kubernetes deployments.

- Provides faster boot time and is very cost effective.
- Supports small footprint to deliver highly agile, advanced security services in a container form factor.

cSRX supports easy, flexible, and highly scalable deployment options covering various customer use cases, including application protection, and microsegmentation through a Docker container management solution.

The cSRX deployed as a service in a deployment object, will allow for scale-up and scale down of the cSRX on demand. It functions as a firewall, protecting workloads deployed in the cluster with the configuration of rich advanced services.

Some deployments require highly agile and lightweight security VNF that can scale massively. For such deployments VM based VNF is not a scalable solution and requires container based security VNF.

- Supports network function service chains, allowing high availability as well as containerized security that scales in individual network functions as needed.
- Provides management flexibility with NETCONF and Security Director to support integration with third-party management and cloud orchestration tools like Kubernetes.

  Also, with EKS, the latest security patches are applied to your cluster's control plane to ensure security of your cluster.

Junos OS Features Supported on cSRX

IN THIS SECTION

cSRX provides Layer 4 through 7 secure services in a containerized environment.

Supported SRX Series Features on cSRX

Table 1 on page 9 provides a high-level summary of the feature categories supported on cSRX and any feature considerations.

To determine the Junos OS features supported on cSRX, use the Juniper Networks Feature Explorer, a Web-based application that helps you to explore and compare Junos OS feature information to find the right software release and hardware platform for your network. See Feature Explorer.

Table 1: Security Features Supported on cSRX

Security Features	Considerations
Application Tracking (AppTrack)	Understanding AppTrack
Application Firewall (AppFW)	Application Firewall Overview
Application Identification (AppID)	Understanding Application Identification Techniques
Basic Firewall Policy	Understanding Security Basics
Brute force attack mitigation	
DoS/DDoS protection	DoS Attack Overview DoS Attack Overview
Intrusion Prevention System (IPS)	For SRX Series IPS configuration details, see: Understanding Intrusion Detection and Prevention for SRX Series
IPv4 and IPv6	Understanding IPv4 Addressing Understanding IPv6 Address Space
Jumbo Frames	Understanding Jumbo Frames Support for Ethernet Interfaces
SYN cookie protection	Understanding SYN Cookie Protection
Malformed packet protection	

Table 1: Security Features Supported on cSRX *(Continued)*

Security Features	Considerations
Unified Threat Management (UTM)	Includes support for all UTM functionality on the cSRX platform, such as: • Antispam • Sophos Antivirus • Web filtering • Content filtering For SRX Series UTM configuration details, see: Unified Threat Management Overview For SRX Series UTM antispam configuration details, see: Antispam Filtering Overview
User Firewall	Includes support for all user firewall functionality on the cSRX platform, such as: • Policy enforcement with matching source identity criteria • Logging with source identity information • Integrated user firewall with active directory • Local authentication For SRX Series user firewall configuration details, see: Overview of Integrated User Firewall
Zones and Zone based IP spoofing	Understanding IP Spoofing

2

CHAPTER

Deployment

Deployment of cSRX on AWS Cloud Using Elastic Kubernetes Services (EKS) for Orchestration

SUMMARY

cSRX deployment on AWS can be achieved as plain docker container on EC2 instance using Amazon Elastic Kubernetes Service (Amazon EKS). The cluster management is done by Kubernetes, assisted by AWS and all Kubernetes commands work as is in case of EKS for container creation and management. This topic provides you details on how you can deploy cSRX on AWS cloud using Elastic Kubernetes Services (EKS) for Orchestration.

IN THIS SECTION

Deploy cSRX on AWS Cloud Using Elastic Kubernetes Services (EKS)

This topic provides you details to deploy the cSRX on AWS cloud.

1. As a prerequisite, install AWS CLI, eksctl, and kubectl packages. For more information, see Getting started with Amazon EKS.
2. Create cluster on EKS using the following CLI command:

```
# eksctl create cluster --name <cluster_name> --version 1.17 --region us-west-2 --nodegroup-name
```

```
<node_group_name> --node-type t3.medium --nodes 2 --nodes-min 1 --nodes-max 3 --ssh-access --ssh-public-key ~/.ssh/id_rsa.pub --managed --asg-access
```

3. Monitor the cluster status using the eksctl commands listed below:

```
# ubuntu@ip-172-31-0-168:~$ eksctl get cluster
NAME            REGION
csrx-eks-cluster        us-west-2
```

4. Verify the cluster created. Cluster with instance type of t3.medium and 2 worker nodes is created.

```
# kubectl get nodes
NAME                                          STATUS   ROLES    AGE     VERSION
ip-192-168-10-52.us-west-2.compute.internal   Ready    <none>   7d21h   v1.17.9
ip-192-168-33-89.us-west-2.compute.internal   Ready    <none>   7d21h   v1.17.9
```

5. Start a cSRX pod on the EKS cluster using the following .yaml file. Use this yaml file as reference and run the kubectl command to deploy cSRX pod. Use the cSRX image available on AWS marketplace to spawn cSRX containers.

```
# kubectl create -f csrx.yaml
```

6. Verify the deployment using the kubectl command below:

```
# kubectl get deployment csrx
 NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
 csrx5                 1/1     1            1           2m
```

Sample File for cSRX Deployment

This topic provides you sample file for deploying cSRX in AWS cloud using AWS EKS orchestration.

```
vim csrx.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: csrx-byol
  labels:
    app: csrx-byol
spec:
  replicas: 2
  selector:
    matchLabels:
      app: csrx-byol
  template:
    metadata:
      name: csrx-byol
```

```
    labels:
      app: csrx-byol
    annotations:
      k8s.v1.cni.cncf.io/networks: br-51@eth1, br-52@eth2
  spec:
    serviceAccountName: csrxpod
    containers:
    - name: csrx-byol
      securityContext:
         privileged: true
      image: <csrx-image> ## replace image name with repo:tag
      ports:
        - containerPort: 80
      env:
      - name: CSRX_SIZE
        value: "large"
      - name: CSRX_HUGEPAGES
        value: "no"
      - name: CSRX_PACKET_DRIVER
        value: "interrupt"
      - name: CSRX_FORWARD_MODE
        value: "routing"
      - name: CSRX_AUTO_ASSIGN_IP
        value: "yes"
      - name: CSRX_MGMT_PORT_REORDER
        value: "yes"
      - name: CSRX_TCP_CKSUM_CALC
        value: "yes"
      - name: CSRX_JUNOS_CONFIG
        value: "/var/jail/csrx_config"
      - name: CSRX_LICENSE_FILE
        value: "/var/jail/.csrx_license"
      volumeMounts:
      - name: disk
        mountPath: "/dev"
      - name: config
        mountPath: "/var/jail"
    volumes:
    - name: disk
      hostPath:
        path: /dev
        type: Directory
    - name: config
```

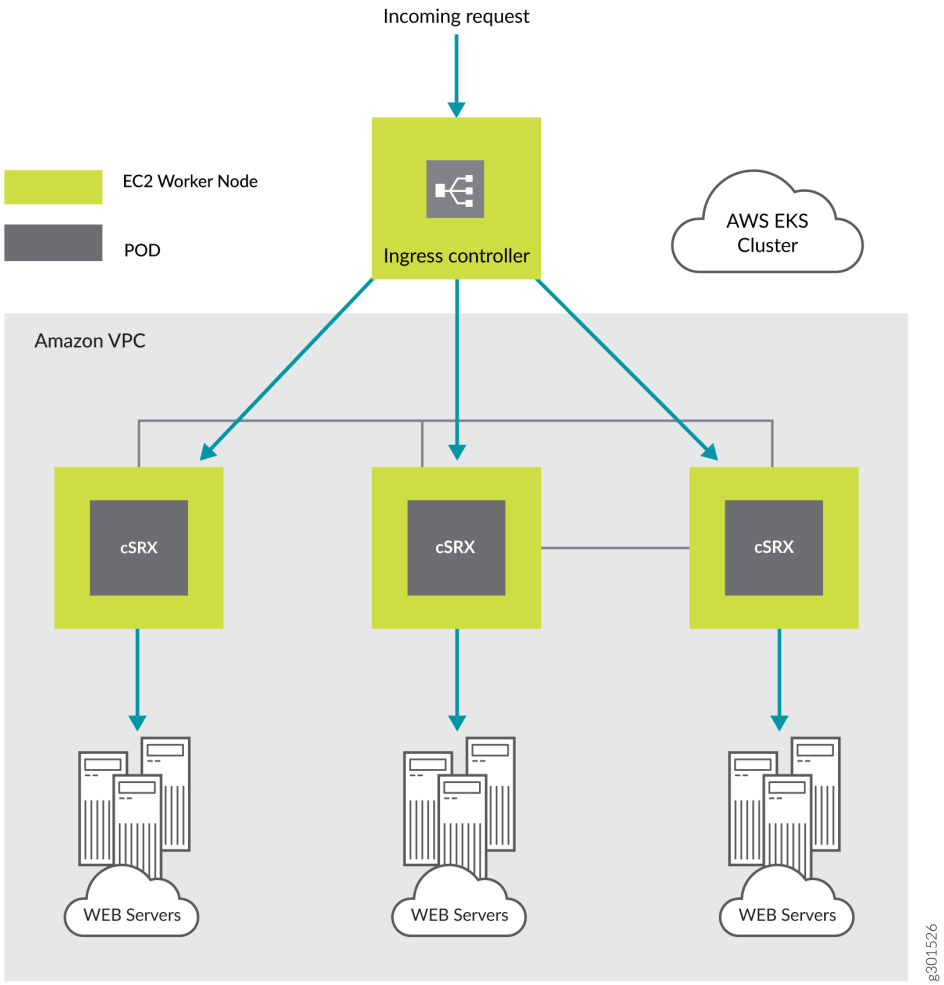
```
        configMap:
          name: cm-byol
          items:
          - key: csrx_config
            path: csrx_config
          - key: csrx_license
            path: .csrx_license
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app: csrx-byol
  name: csrx-byol
spec:
  selector:
      app: csrx-byol
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

cSRX as a Service with Ingress Controller on AWS EKS

The cSRX can be deployed as a service using a Network Load Balancer with NGINX Ingress Controller on Amazon EKS. The cSRX deployed as a service in a deployment object allows you to scale up and scale down by distributing the traffic among different cSRX PODs. Also, cSRX functions as a firewall, protecting workloads deployed in the cluster with rich advanced security services.

Figure 3 on page 16 illustrates AWS EKS ingress controller.

Figure 3: AWS EKS Ingress Controller



To deploy the cSRX as Ingress controller on AWS EKS:

1. Define and deploy cSRX as K8s POD or as ReplicaSet. This type of deployment is the standard K8s to define and to manage resource. Also, allows you to deploy cSRX container on specified work nodes, update or rollback based on your request.
2. Use Kubectl and YAML templates to define and to deploy cSRX related resource on command line. K8s API server can process the request from other applications.
3. Expose cSRX as K8s service with load balancing. EKS supports Kubernetes Network Load Balancer (NLB) and EKS specific Application Load Balancer (ALB).
4. The cSRX POD is identified with predefined selectors and exposed with supported load balancer. The load balancer is the NGINX ingress controller and AWS NLB as external load balancer.

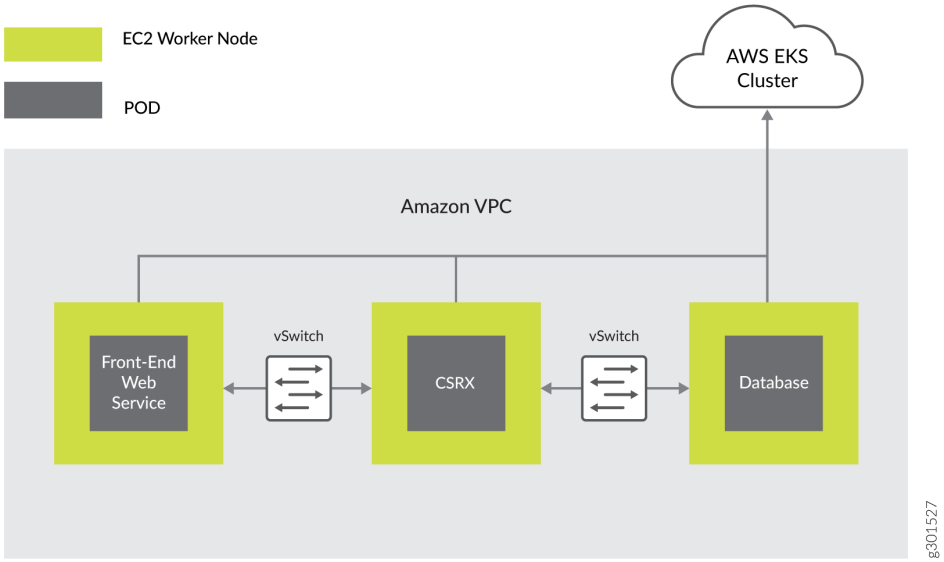
5. Connect cSRX container to the external network using Multus with flannel CNI. cSRX requires at least three interfaces (1 management port and 2 revenue ports).

Microsegmentation with cSRX in AWS

With micro-segmentation (East and the West firewall) application interacting in the same EKS, VPC is secured with the supported application layer security provided by cSRX. Multus-CNI and flannel is used to support multiple interfaces per POD for micro-segmentation. Multus-CNI and flannel leverages the Linux native CNI support of bridge and the MAC VLAN to connect to external interfaces.

Figure 4 on page 17 illustrates AWS EKS microsegmentation with cSRX in AWS.

Figure 4: AWS EKS Microsegmentation



3

CHAPTER

Licensing

Licensing for cSRX on AWS Marketplace

- cSRX is available with 60 days free trial eval license (S-CSRX-A1 SKU). The eval license in cSRX expires after 60 days.
- AWS supports Bring Your Own License (BYOL) licensing model. The BYOL license model allows you to customize your license, subscription and support to fit your needs. You can purchase BYOL from Juniper Networks or Juniper Networks authorized reseller.
- The cSRX Container Firewall software features require a license to activate the feature. To understand more about cSRX Container Firewall licenses, see
  - Supported Features on cSRX.
  - Juniper Agile Licensing Guide.
  - Flex Software License for cSRX.
- To add, delete, and manage licenses, see Managing cSRX Licenses.