RabbitMQ 3 Node Cluster Setup | Helm Chat

I’m assuming that kind, kubectl, docker and helm are already installed in your system. If not then please installed it first as it’s a prerequisite for this task.

After installation follow the below steps:

Step: 1

I’m using Kind for Kubernetes so as a first step we need to create a kind config file.

kind-multi-node-cluster.yaml

kind: Cluster

apiVersion: kind.x-k8s.io/v1alpha4

nodes:

  - role: control-plane

    extraPortMappings:

      - containerPort: 30000

        hostPort: 30000

        protocol: TCP

  - role: worker

    extraMounts:

      - hostPath: /home/linux/helm/worker1

        containerPath: /mnt/disks/worker1

  - role: worker

    extraMounts:

      - hostPath: /home/linux/helm/worker2

        containerPath: /mnt/disks/worker2

  - role: worker

    extraMounts:

      - hostPath: /home/linux/helm/worker3

        containerPath: /mnt/disks/worker3

After creating this file, run the below command to create kubernetes 3 node cluster.

kind create cluster --config kind-multi-node-cluster.yaml

Above command will create 3 docker containers that works as a node for your cluster.

A black background with white text

Description automatically generated

Step: 2

Now, you have to create a chart with below command.

helm create rabbit-helm

By default this command will create a chart for installing nginx. Now we have to update those files as per below.

values.yaml

container:

  image:

    name: rabbitmq:3.12-management

  ports:

    amqp:

      containerPort: 5672

    management:

      containerPort: 15672

fullnameOverride: ""

ingress:

  enabled: false

nameOverride: ""

nodeSelector: {}

persistentVolume:

  accessMode: ReadWriteOnce

  enabled: true

  nodes:

  - hostname: kind-worker

    path: /mnt/disks/worker1

  - hostname: kind-worker2

    path: /mnt/disks/worker2

  - hostname: kind-worker3

    path: /mnt/disks/worker3

  reclaimPolicy: Delete

  size: 10Gi

  storageClass: local-storage

podAnnotations: {}

podSecurityContext: {}

rabbitmq:

  addressType: dns

  cluster\_partition\_handling: autoheal

  clustering:

    enabled: true

  config:

    cluster\_formation:

      k8shost: kubernetes.default.svc.cluster.local

      node\_cleanup\_only\_log\_warning: true

      peer\_discovery\_backend: rabbit\_peer\_discovery\_classic\_config

  erlangCookie: lgRJdhxgCe9OmYf

  log\_console: true

node1: rabbit@rabbitmq-release-rabbitmq-0.rabbitmq-release-internal

  node2: rabbit@rabbitmq-release-rabbitmq-1.rabbitmq-release-internal

  node3: rabbit@rabbitmq-release-rabbitmq-2.rabbitmq-release-internal

  password: admin@123

  username: admin

replicaCount: 3

probes:

  liveness:

    path: /api/healthchecks/node

    port: 15672

    scheme: HTTP

    initialDelaySeconds: 120

    periodSeconds: 100

    timeoutSeconds: 50

    failureThreshold: 20

    successThreshold: 1

  readiness:

    path: /api/healthchecks/node

    port: 15672

    scheme: HTTP

    initialDelaySeconds: 120

    periodSeconds: 100

    timeoutSeconds: 50

    failureThreshold: 20

    successThreshold: 1

resources:

  limits:

    cpu: "2"

    memory: 2Gi

  requests:

    cpu: "1"

    memory: 1Gi

securityContext: {}

service:

  headless: true

  internal:

    port: 5672

    type: ClusterIP

  management:

    port: 15672

    type: ClusterIP

serviceAccount:

  annotations: {}

  create: true

  name: ""

autoscaling:

  enabled: false

  maxReplicas: 100

  minReplicas: 1

  targetCPUUtilizationPercentage: 80

affinity: {}

tolerations: []

templates/configmap.yaml

apiVersion: v1

kind: ConfigMap

metadata:

  name: {{ .Release.Name }}-config

data:

  rabbitmq.conf: |

    # Enable Kubernetes peer discovery

    #cluster\_formation.peer\_discovery\_backend = rabbit\_peer\_discovery\_k8s

    cluster\_formation.k8s.host = {{ .Values.rabbitmq.config.cluster\_formation.k8shost }}

    cluster\_formation.node\_cleanup.only\_log\_warning = {{ .Values.rabbitmq.config.cluster\_formation.node\_cleanup\_only\_log\_warning }}

    cluster\_formation.peer\_discovery\_backend = {{ .Values.rabbitmq.config.cluster\_formation.peer\_discovery\_backend }}

    cluster\_formation.classic\_config.nodes.1 = {{ .Values.rabbitmq.node1 }}

    cluster\_formation.classic\_config.nodes.2 = {{ .Values.rabbitmq.node2 }}

    cluster\_formation.classic\_config.nodes.3 = {{ .Values.rabbitmq.node3 }}

    # Cluster auto-healing

    cluster\_partition\_handling = {{ .Values.rabbitmq.cluster\_partition\_handling }}

    # Logs and management

    log.console = {{ .Values.rabbitmq.log\_console }}

templates/persistentvolume.yaml

{{- $releaseName := .Release.Name -}}

{{- range $index, $node := .Values.persistentVolume.nodes }}

apiVersion: v1

kind: PersistentVolume

metadata:

  name: {{ $releaseName }}-pv-{{ $index }}

  labels:

    app: {{ $releaseName }}

spec:

  capacity:

    storage: {{ $.Values.persistentVolume.size }}

  volumeMode: Filesystem

  accessModes:

  - {{ $.Values.persistentVolume.accessMode }}

  persistentVolumeReclaimPolicy: {{ $.Values.persistentVolume.reclaimPolicy }}

  storageClassName: {{ $.Values.persistentVolume.storageClass }}

  local:

    path: {{ $node.path }}

  nodeAffinity:

    required:

      nodeSelectorTerms:

      - matchExpressions:

          - key: kubernetes.io/hostname

            operator: In

            values:

              - {{ $node.hostname }}

---

{{- end }}

templates/secrets.yaml

apiVersion: v1

kind: Secret

metadata:

  name: {{ .Release.Name }}-secrets

type: Opaque

data:

  erlang-cookie: {{ .Values.rabbitmq.erlangCookie | b64enc }}

  rabbitmq-username: {{ .Values.rabbitmq.username | b64enc }}

  rabbitmq-password: {{ .Values.rabbitmq.password | b64enc }}

templates/service.yaml

apiVersion: v1

kind: Service

metadata:

  name: {{ .Release.Name }}-management

spec:

  type: {{ .Values.service.management.type }}

  ports:

    - port: {{ .Values.service.management.port }}

      targetPort: {{ .Values.service.management.port }}

  selector:

    app: rabbitmq

---

apiVersion: v1

kind: Service

metadata:

  name: {{ .Release.Name }}-internal

spec:

  type: {{ .Values.service.internal.type }}

  ports:

    - port: {{ .Values.service.internal.port }}

      targetPort: {{ .Values.service.internal.port }}

  selector:

    app: rabbitmq

templates/statefulset.yaml

apiVersion: apps/v1

kind: StatefulSet

metadata:

  name: {{ .Release.Name }}-rabbitmq

spec:

  replicas: {{ .Values.replicaCount }}

  selector:

    matchLabels:

      app: rabbitmq

  serviceName: {{ .Release.Name }}-internal

  template:

    metadata:

      labels:

        app: rabbitmq

    spec:

      initContainers:

        - name: set-erlang-cookie-permissions

          image: busybox

          command:

            - sh

            - -c

            - |

              cp /secrets/.erlang.cookie /cookie/.erlang.cookie

              chmod 400 /cookie/.erlang.cookie

              chown 999:999 /cookie/.erlang.cookie

          volumeMounts:

            - name: rabbitmq-erlang-cookie

              mountPath: /secrets

            - name: cookie-dir

              mountPath: /cookie

      containers:

        - name: rabbitmq

          image: {{ .Values.container.image.name }}

          ports:

            - name: amqp

              containerPort: {{ .Values.container.ports.amqp.containerPort }}

            - name: management

              containerPort: {{ .Values.container.ports.management.containerPort }}

          resources:

            requests:

              memory: {{ .Values.resources.requests.memory }}

              cpu: {{ .Values.resources.requests.cpu }}

            limits:

              memory: {{ .Values.resources.limits.memory }}

              cpu: {{ .Values.resources.limits.cpu }}

          livenessProbe:

            exec:

              command:

                - /bin/bash

                - '-ec'

                - rabbitmqctl status

            initialDelaySeconds: {{ .Values.probes.liveness.initialDelaySeconds }}

            periodSeconds: {{ .Values.probes.liveness.periodSeconds }}

            timeoutSeconds: {{ .Values.probes.liveness.timeoutSeconds }}

            failureThreshold: {{ .Values.probes.liveness.failureThreshold }}

            successThreshold: {{ .Values.probes.liveness.successThreshold }}

          env:

            - name: RABBITMQ\_PLUGINS

              value: "rabbitmq\_management"

            - name: RABBITMQ\_DEFAULT\_USER

              valueFrom:

                secretKeyRef:

                  name: {{ .Release.Name }}-secrets

                  key: rabbitmq-username

            - name: RABBITMQ\_DEFAULT\_PASS

              valueFrom:

                secretKeyRef:

                  name: {{ .Release.Name }}-secrets

                  key: rabbitmq-password

            - name: RABBITMQ\_ERLANG\_COOKIE

              valueFrom:

                secretKeyRef:

                  name: {{ .Release.Name }}-secrets

                  key: erlang-cookie

            - name: RABBITMQ\_USE\_LONGNAME

              value: "true"

            - name: POD\_NAME

              valueFrom:

                fieldRef:

                  fieldPath: metadata.name

            - name: RABBITMQ\_NODENAME

              value: "rabbit@$(POD\_NAME).rabbitmq-release-internal"

            - name: RABBITMQ\_CLUSTER\_DISCOVERY

              value: "k8s"

            - name: RABBITMQ\_CLUSTER\_DISCOVERY\_K8S\_HOST

              value: "kubernetes.default.svc.cluster.local"

            - name: RABBITMQ\_CLUSTER\_DISCOVERY\_K8S\_SERVICE\_NAME

              value: "{{ .Release.Name }}-internal"

            - name: RABBITMQ\_CLUSTER\_DISCOVERY\_K8S\_ADDRESS\_TYPE

              value: "hostname"

            - name: RABBITMQ\_CONFIG\_FILE

              value: "/etc/rabbitmq/rabbitmq.conf"

          volumeMounts:

            - name: rabbitmq-data

              mountPath: /var/lib/rabbitmq

            - name: rabbitmq-config

              mountPath: /etc/rabbitmq/rabbitmq.conf

              subPath: rabbitmq.conf

            - name: cookie-dir

              mountPath: /var/lib/rabbitmq/.erlang.cookie

              subPath: .erlang.cookie

      volumes:

        - name: cookie-dir

          emptyDir: {}

        - name: rabbitmq-config

          configMap:

            name: {{ .Release.Name }}-config

        - name: rabbitmq-erlang-cookie

          secret:

            secretName: {{ .Release.Name }}-secrets

            items:

              - key: erlang-cookie

                path: .erlang.cookie

  volumeClaimTemplates:

    - metadata:

        name: rabbitmq-data

      spec:

        accessModes: ["ReadWriteOnce"]

        resources:

          requests:

            storage: {{ .Values.persistentVolume.size }}

        storageClassName: {{ .Values.persistentVolume.storageClass }}

That’s it... You are good to go. This is all you need to do to deploy 3 node RabbitMQ cluster.

Step: 3

Now run the below command to deploy our custom helm chart.

helm install rabbitmq-release ./rabbit-helm/

A black screen with white text

Description automatically generated

Run `helm ls` to verify it.

A black screen with white text

Description automatically generated

Run `kubectl get all` to get all resources of current namespace.

A black screen with white text

Description automatically generated

Now, to access the RabbitMQ management, you have to do a port forward as we have service of ClusterIP type and kind don’t support accessing cluster applications without port forwarding.

Run `kubectl port-forward svc/rabbitmq-release-management 15672:15672` to access your rabbitmq management on `http://localhost:15672`.

A black screen with white text

Description automatically generated

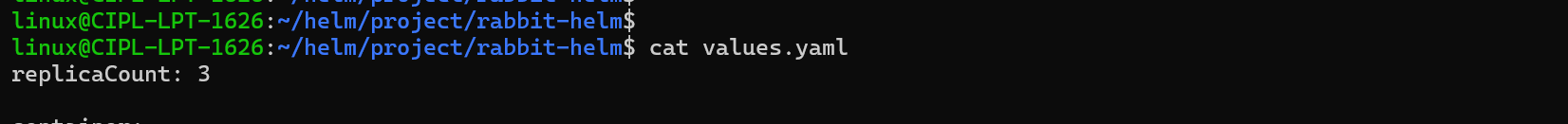
After inserting your credentials, you will be redirected to the dashboard. As you can see in the dashboard all 3 nodes are joined.

A screenshot of a computer

Description automatically generated

How to scale the RabbitMQ cluster

Just increase the count in your values.yaml file and run ` helm upgrade rabbitmq-release . -f values.yaml `to scale the cluster.



Troubleshoot common issues

**1. Pod Not Starting (CrashLoopBackOff)**

**Symptoms:**

* Pods fail to start or are in CrashLoopBackOff status.
* Look for errors in the logs using:

kubectl logs <rabbitmq-pod-name>

**Solutions:**

* **Check for permission issues:** Ensure that the RabbitMQ user creation scripts are working and that the necessary secrets (e.g., rabbitmq-username, rabbitmq-password) are created.
* **Verify resource limits:** Ensure that the resource requests/limits are appropriate. If RabbitMQ is memory-intensive, consider adjusting the limits.
* **Check PVC storage:** If persistent storage is enabled, ensure the PVC has enough space or isn’t in an error state.

**2. Readiness/Liveness Probes Failing**

**Symptoms:**

* Pods are stuck in Unhealthy state.

**Solutions:**

* **Increase probe timeout:** If RabbitMQ takes time to start, increase the readiness/liveness probe timeout in values.yaml:

livenessProbe:

  timeoutSeconds: 60

  periodSeconds: 20

  failureThreshold: 5

readinessProbe:

  timeoutSeconds: 60

  periodSeconds: 20

  failureThreshold: 5

* **Check RabbitMQ Logs:** RabbitMQ may still be starting and needs more time. You can adjust the probes or look into logs:

kubectl logs <rabbitmq-pod-name>

**3. Persistent Volume Claims (PVC) Issues**

**Symptoms:**

* RabbitMQ pods fail to mount the PVC, causing startup failures.

**Solutions:**

* **Check PVC status:** Use kubectl get pvc to check the status of your PVC.
* **Recreate PVC if needed:**

kubectl delete pvc <pvc-name>

helm upgrade --install rabbitmq -f values.yaml

**4. User Not Created (Authentication Issues)**

**Symptoms:**

* Unable to log in with the RabbitMQ username/password.

**Solutions:**

* **Ensure rabbitmq-credentials Secret is correctly set:** Verify that the secret is created with correct keys (rabbitmq-username and rabbitmq-password).
* **Manual User Creation:** If users are not created automatically, you can create them manually:

kubectl exec -it <rabbitmq-pod-name> -- rabbitmqctl add\_user myuser mypassword

kubectl exec -it <rabbitmq-pod-name> -- rabbitmqctl set\_user\_tags myuser administrator

kubectl exec -it <rabbitmq-pod-name> -- rabbitmqctl set\_permissions -p / myuser ".\*" ".\*" ".\*"

It’sss Cleaning Time

Now, after work is done, you can delete your resources by running `helm uninstall rabbitmq-release`

