# Install the Couchbase Service Broker on Open-Source Kubernetes

## Prerequisites

This page details the platform and software requirements for running the Service Broker.

### **Kubernetes Cluster**

In general, any Kubernetes distribution based on a supported version should work.

Supported Kubernetes Version:

* Kubernetes 1.17+

### **Service Catalog**

The Service Broker is recommended for use with the [Kubernetes Service Catalog](https://kubernetes.io/docs/concepts/extend-kubernetes/service-catalog/).

Installation instructions can be found in the [official documentation](https://kubernetes.io/docs/tasks/service-catalog/).

As the Service Broker is standards based, any version of the Service Catalog supporting the Open Service Broker API version 2.13+ is supported.

### CAO

Visit this page (<https://docs.couchbase.com/operator/current/install-kubernetes.html>) to install Couchbase Autonomous Operator.

## Package Downloads

Download Couchbase Service Broker [package](https://www.couchbase.com/downloads) and unpack on the same computer where you normally run kubectl.

The Couchbase Service Broker package contains YAML configuration files and command-line tools that you will use to install the Couchbase Service Broker and Couchbase Clusters.

## Install Couchbase Service Broker

Important:

After you unpack the download, the resulting directory will be titled something like couchbase-service-broker\_x.x.x. Make sure to cd into this directory before you run the commands in this guide.

**Install the CRD**

The Service Broker is configured using a Kubernetes Custom Resource that allows type checking of your configurations and must be installed first:

$ kubectl create -f crd.yaml

**Install a configuration**

$ kubectl create -f servicebrokerconfig.yaml

## Install the service broker service

## $ kubectl create -f broker.yaml

## The ServiceAccount the Service Broker runs as is bound to the Role we created in the previous step. A Secret contains the TLS configuration and bearer token for authentication as the Service Broker is secure by default. The Deployment creates the Service Broker and ensures it is highly available and a Service makes it discoverable by the Kubernetes Service Catalog in the next step.

## Check the status of the service broker

## $ kubectl get deployments

NAME READY UP-TO-DATE AVAILABLE AGE

couchbase-operator 1/1 1 1 33m

couchbase-operator-admission 1/1 1 1 37m

couchbase-service-broker 1/1 1 1 32m

## Register the service broker with the service catalog

## The final step is to tell the Kubernetes Service Catalog about our Service Broker.

## $ kubectl create -f clusterservicebroker.yaml

## Create Couchbase Cluster

## Provision Couchbase Cluster Service:

## $ svcat provision csb --class couchbase-osb-service --plan csb-basic --param password=password --wait

You will end up with a 3 node cluster (by default), with a Bucket, an Administrator user with the password you provided.

## Bind to the provisioned service

## $ svcat bind csb

## Name: csb

## Namespace: default

## Status:

## Secret: csb

## Instance: csb

## 

## Parameters:

## No parameters defined

This will create a user and allow access to the bucket.  Connection string, username, password and CA certificate will be in the secret, ready to be used by a client of some variety.

**$ kubectl get secrets csb**

NAME TYPE DATA AGE

csb Opaque 5 7m58s

To access the couchbase cluster UI console,

**$ kubectl port-forward couchbase-instance-winhhoku-0000 8091**

Go to http://localhost:8091

login with username as Administrator & password as password

# Sample example – Deploying couchbase cluster locally on minikube using Couchbase Service Broker

1. **Install docker (Note: for minikube with 8GB memory, assign 8 GB memory to Docker)**

[**https://hub.docker.com/editions/community/docker-ce-desktop-mac/**](https://hub.docker.com/editions/community/docker-ce-desktop-mac/)

wait for the installation to complete, verification, then access permissions and the whale will appear in the status bar!

1. **Install kubectl (**[**https://kubernetes.io/docs/tasks/tools/install-kubectl/**](https://kubernetes.io/docs/tasks/tools/install-kubectl/)**)**

curl -LO "https://storage.googleapis.com/kubernetes-release/release/**$(**curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt**)**/bin/darwin/amd64/kubectl"

chmod +x ./kubectl

sudo mv ./kubectl /usr/local/bin/kubectl

kubectl version –client

1. **Install Homebrew (**[**https://brew.sh/**](https://brew.sh/)**)**

/bin/bash -c "$(curl -fsSL <https://raw.githubusercontent.com/Homebrew/install/master/install.sh>)"

1. **Install helm**

$ brew install helm

1. **Install Minikube (**[**https://minikube.sigs.k8s.io/docs/start/**](https://minikube.sigs.k8s.io/docs/start/)**)**

$ brew install minikube

1. **Create Kubernetes Cluster Using minikube**

$ minikube start --cpus 2 --memory 8192

1. **Install Service Catalog**

$helm repo add svc-cat <https://svc-catalog-charts.storage.googleapis.com>

$ helm search repo service-catalog

$ kubectl create namespace catalog

$ helm install catalog svc-cat/catalog --namespace catalog

$ brew update

$ brew install kubernetes-service-catalog-client

1. **Install CAO (**[**https://docs.couchbase.com/operator/current/install-kubernetes.html**](https://docs.couchbase.com/operator/current/install-kubernetes.html)**)**

**Download the CAO package**

**Extract the downloaded package**

**$ sudo tar xf couchbase-autonomous-operator-kubernetes\_2.1.0-macos-x86\_64.zip**

**$ cd couchbase-autonomous-operator-kubernetes\_2.1.0-macos-x86\_64**

**$ kubectl create -f crd.yaml**

**Warning: apiextensions.k8s.io/v1beta1 CustomResourceDefinition is deprecated in v1.16+, unavailable in v1.22+; use apiextensions.k8s.io/v1 CustomResourceDefinition**

customresourcedefinition.apiextensions.k8s.io/couchbaseautoscalers.couchbase.com created

customresourcedefinition.apiextensions.k8s.io/couchbasebackuprestores.couchbase.com created

customresourcedefinition.apiextensions.k8s.io/couchbasebackups.couchbase.com created

customresourcedefinition.apiextensions.k8s.io/couchbasebuckets.couchbase.com created

customresourcedefinition.apiextensions.k8s.io/couchbaseclusters.couchbase.com created

customresourcedefinition.apiextensions.k8s.io/couchbaseephemeralbuckets.couchbase.com created

customresourcedefinition.apiextensions.k8s.io/couchbasegroups.couchbase.com created

customresourcedefinition.apiextensions.k8s.io/couchbasememcachedbuckets.couchbase.com created

customresourcedefinition.apiextensions.k8s.io/couchbasereplications.couchbase.com created

customresourcedefinition.apiextensions.k8s.io/couchbaserolebindings.couchbase.com created

customresourcedefinition.apiextensions.k8s.io/couchbaseusers.couchbase.com created

**$ bin/cbopcfg generate admission | kubectl create -f -**

serviceaccount/couchbase-operator-admission created

clusterrole.rbac.authorization.k8s.io/couchbase-operator-admission created

clusterrolebinding.rbac.authorization.k8s.io/couchbase-operator-admission created

secret/couchbase-operator-admission created

deployment.apps/couchbase-operator-admission created

service/couchbase-operator-admission created

**Warning: admissionregistration.k8s.io/v1beta1 MutatingWebhookConfiguration is deprecated in v1.16+, unavailable in v1.22+; use admissionregistration.k8s.io/v1 MutatingWebhookConfiguration**

mutatingwebhookconfiguration.admissionregistration.k8s.io/couchbase-operator-admission created

**Warning: admissionregistration.k8s.io/v1beta1 ValidatingWebhookConfiguration is deprecated in v1.16+, unavailable in v1.22+; use admissionregistration.k8s.io/v1 ValidatingWebhookConfiguration**

validatingwebhookconfiguration.admissionregistration.k8s.io/couchbase-operator-admission created

**$ bin/cbopcfg generate operator --scope cluster | kubectl create -f -**

serviceaccount/couchbase-operator created

clusterrole.rbac.authorization.k8s.io/couchbase-operator created

clusterrolebinding.rbac.authorization.k8s.io/couchbase-operator created

deployment.apps/couchbase-operator created

service/couchbase-operator created

1. **Install Couchbase Service Broker**

**$ kubectl create -f crd.yaml**

Warning: apiextensions.k8s.io/v1beta1 CustomResourceDefinition is deprecated in v1.16+, unavailable in v1.22+; use apiextensions.k8s.io/v1 CustomResourceDefinition

customresourcedefinition.apiextensions.k8s.io/servicebrokerconfigs.servicebroker.couchbase.com created

**$ kubectl create -f servicebrokerconfig.yaml**

role.rbac.authorization.k8s.io/couchbase-service-broker created

servicebrokerconfig.servicebroker.couchbase.com/couchbase-service-broker created

**$ kubectl create -f broker.yaml -- wait**

serviceaccount/couchbase-service-broker created

rolebinding.rbac.authorization.k8s.io/couchbase-service-broker created

secret/couchbase-service-broker created

deployment.apps/couchbase-service-broker created

service/couchbase-service-broker created

clusterservicebroker.servicecatalog.k8s.io/couchbase-service-broker created

**$ kubectl create -f clusterservicebroker.yaml**

clusterservicebroker.servicecatalog.k8s.io/couchbase-service-broker created

**$ svcat provision csb --class couchbase-osb-service --plan csb-basic --param password=password –-wait**

Waiting for the instance to be provisioned...

    Name:           csb

    Namespace:      default

    Status:         Ready - The instance was provisioned successfully @ 2020-11-23 10:05:25 +0000 UTC

    DashboardURL:   <https://couchbase-instance-c08r083o.default:18091>

    Class: 8522e991-07bc-4225-a859-1eec1e333153

Plan: ec0f2c9b-0277-46d7-985f-ba1fbf3b068d

Parameters:

    password: password

You will end up with a 3 node cluster (by default), with a Bucket, an Administrator user with the password you provided.

You can then bind to it:

**$ svcat bind csb**

Name:        csb

    Namespace:   default

    Status:

    Secret:      csb

    Instance:    csb

  Parameters:

    No parameters defined

**$ kubectl get secrets csb**

NAME TYPE DATA AGE

csb Opaque 5 7m58s

To access the couchbase cluster UI console,

**$ kubectl port-forward couchbase-instance-winhhoku-0000 8091**

Go to http://localhost:8091

login with username as Administrator & password as password