

**Collaboard Install**

on AWS EKS cluster

|  |  |
| --- | --- |
| Title: | Collaboard installation on AWS EKS |
| Abstract: | This document details the architecture, sizing and deployment of Collaboard application on an AWS EKS cluster. |
| Status\* | Draft |
| Document Owner: | IBV |
| Document Author: | Noel Hermans |
| \*Draft, In progress, Reviewed, Authorized, Approved, Completed | |

|  |  |  |
| --- | --- | --- |
| **Document Control** | | |
| Document Distribution: | Internal: | All dev Teams |
|  | External: |  |
| Document Authorization: | Authorizer(s): | Gian Paolo Santopaolo |

|  |  |  |  |
| --- | --- | --- | --- |
| **Change History** | | | |
| Date | Ver | Changed By | Details |
| 19/1/2021 | 1.0 | N Hermans | Initial Draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Index

[Overview 3](#_Toc62400551)

[Introduction 3](#_Toc62400552)

[Architecture 3](#_Toc62400553)

[Sizing 4](#_Toc62400554)

[Kubernetes manifests/objects 5](#_Toc62400555)

[Installation runbook 6](#_Toc62400556)

[Prerequisites 6](#_Toc62400557)

[High level instructions 6](#_Toc62400558)

[Detailed instructions (runbook) 6](#_Toc62400559)

[Deploy AWS VPC infrastructure 6](#_Toc62400560)

[Deploy AWS EKS infrastructure 7](#_Toc62400561)

[Provision AWS RDS MSSQL instance 7](#_Toc62400562)

[Provision AWS ElastiCache Redis instance 11](#_Toc62400563)

[Create AWS EFS shared file system 14](#_Toc62400564)

[Set the kubernetes cluster admin credentials 15](#_Toc62400565)

[Deploy EFS CSI driver for kubernetes 16](#_Toc62400566)

[Deploy application containers/manifests in EKS 17](#_Toc62400567)

[Update DNS 18](#_Toc62400568)

# Overview

This document details the architecture, sizing and deployment of Collaboard application on an AWS EKS cluster. This document must be used when a new AWS EKS (Kubernetes) cluster will be deployed

in any AWS account.

Our engineers can answer further questions.

# Introduction

Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that you can use to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes. Kubernetes is an open-source system for automating the deployment, scaling, and management of containerized applications.

Amazon EKS runs Kubernetes control plane instances across multiple Availability Zones to ensure high availability. Amazon EKS automatically detects and replaces unhealthy control plane instances, and it provides automated version updates and patching for them.

The Amazon EKS cluster we have deployed for the Collaboard application is integrated with the following AWS services :

* Amazon ECR for storing the CB container images
* Elastic Load Balancing for load distribution
* IAM for authentication
* Amazon VPC for isolation
* Amazon EFS for shared file system
* Amazon RDS for hosting the MSSQL database
* Amazon EC2 instances for running the container workloads
* Amazon ElastiCache with Redis

# 

# Architecture

A screenshot of a computer

Description automatically generated with medium confidence

## Sizing

The minumum sizing is based on EKS high available control plane or cluster + 2x EC2 instances of type t3 medium. This minimum configuration must be used only in test/developments evenvironments.

The following table must be used to deploy a production cluster:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Container name** | **vCPU** | **RAM (GB)** | **Service** | **Action Type** |
| cb\_proxy | 2 | 2 |  |  |
| cb\_frontend | 1 | 0.5 |  |  |
| cb\_api | 1 | 1.5 |  |  |
| cb\_auth | 1 | 0.5 |  |  |
| mft\_web | 1 | 1.5 |  |  |
| cb\_licensing | 1 | 0.5 |  |  |
| cb\_copyworker | 1 | 0.5 |  |  |
| cb\_worker1 | 1 | 0.5 |  |  |
| cb\_worker2 | 1 | 0.5 |  |  |
| mft\_cleanerservice | 1 | 0.5 |  |  |
| mft\_mergeservice | 1 | 1.5 |  |  |
| cb\_canvasshotter | 1 | 2 |  |  |
| cb\_fileconverter | 1 | 2 |  |  |
| mft\_storageoperationsservice | 1 | 0.5 |  |  |
| **Total** | 15 | 14.5 |  |  |

Corresponding EC2 instances types (**for each AZ**) :

Min: 1x a1.xlarge 4vCPU/8GiB

Max: 4x a1.xlarge 4vCPU/8GiB

Desired: 2x a1.xlarge 4vCPU/8GiB

This calculation is based on the instance information [https://www.ec2instances.info/](https://docs.amazonaws.cn/en_us/eks/latest/userguide/efs-csi.html) considering the optimal configuration for vCPU/RAM and cost per hour.

To consider: AWS Spot or reserved instances instead of on demand instances to reduce the cost of this infrastrucure.

## Kubernetes manifests/objects

When we implement the Kubernetes manifests we need to prepare for the following details :

|  |  |  |  |
| --- | --- | --- | --- |
| **Container name** | **Service** | **Image:Tag** | **Storage Volume(s)** |
| cb\_proxy | proxy | cb\_proxy:latest | - |
| cb\_frontend | frontend | cb\_frontend:latest | - |
| cb\_api | api | cb\_api:latest | yes |
| cb\_auth | auth | cb\_auth:latest | - |
| mft\_web | webapi | mft\_web:latest-stage | yes |
| cb\_licensing | licensing | cb\_licensing:latest | - |
| cb\_copyworker | copyworker | cb\_copyworker:latest | yes |
| cb\_worker1 | worker1 | cb\_worker1:latest | yes |
| cb\_worker2 | worker2 | cb\_worker2:latest | yes |
| mft\_cleanerservice | cleanerservice | mft\_cleanerservice:latest-stage | yes |
| mft\_mergeservice | mergeservice | mft\_mergeservice:latest-stage | yes |
| cb\_canvasshotter | canvasshotter | cb\_canvasshotter:latest | yes |
| cb\_fileconverter | fileconverter | cb\_fileconverter:latest | yes |
| mft\_storageoperationsservice | storageoperationsservice | mft\_storageoperationsservice:latest-stage | yes |

Notes:

1. CPU and Memory limits are defined in the sizing section.
2. Images are stored in ECR and have a base repository path of :

475945325705.dkr.ecr.eu-central-1.amazonaws.com/ibv\_collaboard

1. URL : [https://www.collaboard.app/](https://www.terraform.io/downloads.html) which are configured in DNS pointing to the AW Elastic Load Balancer.
2. Environment variables :

For all containers with a volume attached :

**CB\_FILES** : /var/mft/\_cbfiles\_

**CB\_TEMP** : /var/mft/\_temp\_

For cb\_proxy container (path to the SSL certificates) :

**SSL\_CERTS :** /etc/nginx/certs

# Installation runbook

## Prerequisites

* Terraform >= 0.12 : you can download it at : [https://www.terraform.io/downloads.html](https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-files.html)
* Access to AWS with admin privileges
* AWS Console and API access
* AWS API access configured (see for more details : [https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-files.html](https://www.ec2instances.info/))
* Clone git repo (in Azure ? - TBD) git clone https://gitxxx/cb-eks/aws-eks-cluster.git

## High level instructions

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Action** | **Action Type** | **Duration** |
| 1 | Deploy AWS VPC network infrastructure | script | 10m |
| 2 | Deploy AWS EKS infrastructure | script | 15m |
| 3 | Provision AWS RDS MSSQL instance | Manual (GUI) | 15m |
| 4 | Provision AWS ElastiCache Redis instance | Manual (GUI) | 15m |
| 5 | Create AWS EFS shared file system | Manual (CLI) | 10m |
| 6 | Set the kubernetes cluster admin credential | Manual (CLI) | 1m |
| 7 | Deploy EFS CSI driver for kubernetes | script | 15m |
| 8 | Deploy application containers/manifests | script | 15m |
| 9 | Update DNS | Manual | 10m |

## Detailed instructions (runbook)

### Deploy AWS VPC infrastructure

| **Step** | **Description** | **Commands** |
| --- | --- | --- |
|  | Inittialize terraform in the directory vpc | cd vpc  terraform init |
|  | Make a dry-run to check which resources will be created | terraform plan |
|  | Deploy the infrastructure | terraform apply |

### Deploy AWS EKS infrastructure

| **Step** | **Description** | **Commands** |
| --- | --- | --- |
|  | Inittialize terraform in the directory eks | cd eks  terraform init |
|  | Make a dry-run to check which resources will be created | terraform plan |
|  | Deploy the infrastructure | terraform apply |

### Provision AWS RDS MSSQL instance

| **Step** | **Description** | **Commands** |
| --- | --- | --- |
|  | Login into AWS console with your account | Graphical user interface, application  Description automatically generated |
|  | In the services type “RDS” | **Graphical user interface, application  Description automatically generated** |
|  | Create DB subnetgroup | This is the list of subnets the RDS is conected to. The RDS will be installed in the same VPC (and subnets) as the EKS is installed.  Graphical user interface, application  Description automatically generated  After clicking “Create DB Subnet Group” you need the details :  Enter “Subnet group details” :  *Name : <name of the DB>*  *Description : <description of the DB>*  *VPC : make sure you select the VPC in which the RDS will be installed*  Choose “Add Subnets” :  *Availability Zones : select one of the AZ e.g. “eu-central-1a”*  *Subnets : Select here the all the subnets in the public\_az*    Press “Create” button to create the subnetgroup |
|  | Click create Database | Graphical user interface, text, application, email  Description automatically generated |
|  | Choose Microsoft SQL Server  “Express Edition” | Text  Description automatically generatedText  Description automatically generated |
|  | Specify DB details | You can leave the defaults except for :  Table  Description automatically generated  Choose db.t3.small or db.t3.medium (depending on the resource requirements |
|  | Scroll down to “Settings” | Enter instance details like name, account and password :  Graphical user interface, text, application, email  Description automatically generated  Click “Next” button |
|  | Configure advanced settings : “Network & security” | Select VPC , make sure you select the VPC where the EKS cluster has been installed.  Select subnetgroup created in step 3 “Create subnetgroup”  Select Public accessibility : Yes  Select Create new VPC security group  Graphical user interface, text, application, email  Description automatically generated |
|  | Press “Create database” button at the bottom of the screen | A picture containing shape  Description automatically generated |
|  | Update Database security group | The security group should only contain the IP address of the Azure pipeline which will initialize the DB and the EKS VPC subnet. Update the inbound rules accordingly  Graphical user interface, application  Description automatically generated  The IP : 46.140.50.50/32 is the Azure pipeline  The IP subnet : 10.20.0.0/16 is the CIDR of the EKS VPC |

### Provision AWS ElastiCache Redis instance

| **Step** | **Description** | **Commands** |
| --- | --- | --- |
|  | Login into AWS console with your account | Graphical user interface, application  Description automatically generated |
|  | In the services type “Redis” | **Graphical user interface, application  Description automatically generated** |
|  | Create subnetgroup | Graphical user interface, text, application  Description automatically generated  After clicking the Create Subnet Group :  Graphical user interface, text, application  Description automatically generated  Enter “Subnet group details” :  *Name : <name of the redis-cache>*  *Description : <description of the redis-cache>*  *VPC : make sure you select the VPC in which the Redis-cache will be installed*  Choose “Add Subnets” :  *Availability Zones : select one of the AZ e.g. “eu-central-1a”*  *Subnets : Select here the all the subnets in the public\_az*    Press “Create” button to create the subnetgroup |
|  | Create Redis-cache | Graphical user interface, text, application  Description automatically generated |
|  | Create your Amazon ElastiCache cluster | Select Cluster Engine as “Redis”  Choose location : “Amazon Cloud”  Enter “Redis settings” :  Graphical user interface, text, application, email  Description automatically generated  Enter in the Redis settings :  *Name : <name of the redis-cache>*  *Description : <description of the redis-cache>*  All other settings can be default except :  *Node type : cache.t3.medium* |
|  | Enter the rest of the details | Advanced Redis settings :  *Subnet group : enter here the subnetgroup create in step 3*  Security settings :  *Speficy the securty that should be used by Redis, if not exist, please create one*  Backup and maintenance settings can be left at default.  Press *A blue square with white text  Description automatically generated with low confidence* button in the bottom of the screen to provision the Redis-Cache |
|  | Update Redis security group | The inbound rules are very similar to RDS except that the port will be 6379.  Graphical user interface, text, application  Description automatically generated |

### Create AWS EFS shared file system

| **Step** | **Description** | **Commands** |
| --- | --- | --- |
|  | Create Security Group for EFS | *Command :*  **aws ec2 create-security-group --region eu-central-1 --group-name efs-mount-sg --description "Amazon EFS for EKS, SG for mount target" --vpc-id vpc-03aae752d0015cccd**  *Response :*  {  "GroupId": "sg-0b03101da5564cbf9"  }  Only variable that must be changed is the –vpc-id. You need to use the VPC-id that was created by the terraform scripts to deploy the VPC infrastructure. |
|  | Update security group for inbound rules | *Command :*  **aws ec2 authorize-security-group-ingress --group-id sg-0b03101da5564cbf9 --region eu-central-1 --protocol tcp --port 2049 --cidr 10.20.0.0/16**  Variables that must be changed in this command :  –group-id : You need to use the security-group id that was created above.  --cidr: this is the CIDR of the VPC where the EKS cluster was created. |
|  | Create the EFS file system | *Command :*  **aws efs create-file-system --creation-token creation-token --performance-mode generalPurpose --throughput-mode bursting --region eu-central-1 --tags Key=Name,Value=EKS-EFS-FileSystem –encrypted**  *Response :*  {  "SizeInBytes": {  "ValueInIA": 0,  "ValueInStandard": 0,  "Value": 0  },  "Name": "EKS-EFS-FileSystem",  "CreationToken": "creation-token",  "Encrypted": true,  "Tags": [  {  "Value": "EKS-EFS-FileSystem",  "Key": "Name"  }  ],  "CreationTime": 1610978015.0,  "PerformanceMode": "generalPurpose",  "FileSystemId": "fs-8784a5df",  "NumberOfMountTargets": 0,  "LifeCycleState": "creating",  "KmsKeyId": "arn:aws:kms:eu-central-1:475945325705:key/5b2c6e02-8080-4a36-b5b0-a906bfaf8835",  "OwnerId": "475945325705",  "ThroughputMode": "bursting"  } |
|  | Get subnet-ids from (EC2) instances.  The EC2 instances are refering to the workers nodes that were deployed as part of the EKS cluster. | *Command :*  **aws ec2 describe-instances --filters Name=vpc-id,Values=vpc-03aae752d0015cccd --query 'Reservations[\*].Instances[].SubnetId'**  *Response :*  [  "subnet-0d5fda4b150b8c02a",  "subnet-093d0e7fbe8f189b4"  ]  Only varialble that must be changed is the –filters vpc-id Values. You need to use the VPC-id that was created by the terraform scripts to deploy the VPC infrastructure. |
|  | Create the EFS mount target for all AZ’s | *Command (1 for each AZ) :*  **aws efs create-mount-target --file-system-id fs-8784a5df --subnet-id subnet-0d5fda4b150b8c02a --security-group sg-0b03101da5564cbf9 --region eu-central-1**  **aws efs create-mount-target --file-system-id fs-8784a5df --subnet-id subnet-0a493ead44df93f90 --security-group sg-0b03101da5564cbf9 --region eu-central-1**  **aws efs create-mount-target --file-system-id fs-8784a5df --subnet-id subnet-093d0e7fbe8f189b4 --security-group sg-0b03101da5564cbf9 --region eu-central-1**  Variables that must be changed in these commands :  –security-group : this is the security group id what was created in step 1  --file-system-id : this is the file system id what was created in step 3 (creation of EFS file system) in the Response |

### Set the kubernetes cluster admin credentials

Command :

**aws eks --region eu-central-1 update-kubeconfig --name cb-eks-cluster**

Test :

**kubectl get nodes**

Output :

NAME STATUS AGE VERSION

ip-10-20-71-99.eu-central-1.compute.internal Ready 6d4h v1.18.9-eks-d1db3c

ip-10-20-84-99.eu-central-1.compute.internal Ready 6d4h v1.18.9-eks-d1db3c

### Deploy EFS CSI driver for kubernetes

As the volumes needs to be shared between most of the containers, we can only use AWS EFS.

To make EFS based volumes available for the kubernetes cluster we need first to install the EFS CSI drivers for kubernetes. This is based on : [https://docs.amazonaws.cn/en\_us/eks/latest/userguide/efs-csi.html](https://www.collaboard.app/)

| **Step** | **Description** | **Commands** |
| --- | --- | --- |
|  | Clone the Amazon EFS Container Storage Interface (CSI) driver | **git clone https://github.com/kubernetes-sigs/aws-efs-csi-driver.git** |
|  | Navigate to the multiple\_pods example directory | **cd aws-efs-csi-driver/examples/kubernetes/multiple\_pods/** |
|  | Retrieve your Amazon EFS file system ID that was created in the section “**Create AWS EFS shared file system**” | **aws efs describe-file-systems --query "FileSystems[\*].FileSystemId" --output text**  Output :  fs-8784a5df |
|  | Create the specs/pv.yaml manifest | apiVersion: v1  kind: PersistentVolume  metadata:  name: efs-pv  spec:  capacity:  storage: 5Gi  volumeMode: Filesystem  accessModes:  - ReadWriteMany  persistentVolumeReclaimPolicy: Retain  storageClassName: efs-sc  csi:  driver: efs.csi.aws.com  volumeHandle: **fs-8784a5df** |
|  | Deploy storageclass, PV,PVC manifests | **kubectl apply -f specs/pv.yaml**  **kubectl apply -f specs/claim.yaml**  **kubectl apply -f specs/storageclass.yaml** |
|  | Check manifests have been deployed correctly | **kubectl get sc,pv,pvc**  Output :  NAME PROVISIONER RECLAIMPOLICY VOLUMEBINDINGMODE ALLOWVOLUMEEXPANSION AGE  storageclass.storage.k8s.io/efs-sc **efs.csi.aws.com** Delete Immediate false 5d4h  storageclass.storage.k8s.io/gp2 (default) kubernetes.io/aws-ebs Delete WaitForFirstConsumer false 6d4h  NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS REASON AGE  persistentvolume/**efs-pv** 5Gi **RWX** Retain Bound default/cb-efs-claim efs-sc 5d4h  NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE  persistentvolumeclaim/**cb-efs-claim** Bound efs-pv **5Gi** **RWX** efs-sc 5d4h |

### Deploy application containers/manifests in EKS

| **Step** | **Description** | **Commands** |
| --- | --- | --- |
|  | Manifests are located in k8s directory | cd k8s  kubectl create -f \*.yaml |
|  | Create service and Load Balancer (ELB)  for cb\_proxy container | kubectl create -f service\_loadbalancer\_proxy\_without\_CERT.yaml |
|  | Check all pods are running | kubectl get pods |

### Update DNS

Create or update a DNS record :

DNS CNAME record \*.collaboard.app with target = load-balancer-endpoint

How to get the load-balancer-endpoint ?

**kubectl svc lb-cb-proxy-service**

Output:

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

lb-cb-proxy-service LoadBalancer 172.20.123.84 **a4656bc2c3d2e4a44ab63625ad6da855-1967873529.eu-central-1.elb.amazonaws.com** 443:32059/TCP 5d5h

Result :

CNAME \*.collaboard.app a4656bc2c3d2e4a44ab63625ad6da855-1967873529.eu-central-1.elb.amazonaws.com