Scale MobilityDB on AWS cloud User's Manual

COLLABORATORS			
	TITLE : Scale MobilityDB on AWS cloud User's Manual		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Sid Ahmed BOUZOUIDJA	August 2, 2021	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Abstract

MobilityDB is an open source moving object database system. It is based on PostgreSQL and and built on top of PostGIS extension. It provides an effective functionality and new data type in order to store, analyze data trajectory. As MobilityDB deal with a large dataset and it queries is very complex and expensive in term of cpu and memory comsuming. This behavior cannot be supported by a single host machine. There is a way to make the MobilityDB queries more powerful and takes less execution time is to scaling up across a cluster of nodes. We have thinks on the On-premises cluster of machine but this is not enough according to the rapid grows of technologies and 5G network deployment sooner that will makes the MobilityDB dataset drastically big. In this respect the effective way to scale up the MobilityDB is to create our cluster on cloud services and using an orchestration as a control plane that manage the cluster. We have used another tool which is a Citus Data to gain more performance by partitioning tables on shards in single node. All the deployment manifests and configuration files, docker images and the requirements to deploy MobilityDB on AWS is located in this Github repository.

Chapter 1

Introduction

There are many cloud services around the world as AWS, Azure, Google cloud, IBM cloud, Saleforce and so on. In this study we will deploy the MobilityDB on AWS cloud services. The AWS cloud provides a large services, sometime it difficult to make a choice between them in order to adjust it to our application. In order to scaling MobilityDB, we need to keep in mind of two kind of resources. The volume storage and the computation units. In addition to those two resources we need an orchestrator that manage cluster and ensure the availability. Again the AWS provides different orchestrator like ECS for Elastic Container Service, an EKS for Elastic Kubernetes Service, ECR for Elastic Container Registry. The orchestrator used in this study is the EKS one. There is another tool that may increase the MobilityDB performance, this one is away from the cloud services. The citus Data for distribution. Their idea is to partitioning tables on different shards. This mechanism allow the query plan to rooting rapidly the queries to the right data shard based on the hash value. In following we will see the different kind of orchestrator that may support our deployment and than describe the architecture to deploying the scale of MobilityDB.

1.1 Different possibilities to scale MobilityDB on AWS

As we have mentioned, we have several posibilities to deploy MobilityDB on AWS cloud. In order to take advantages of what AWS services provides, we have made a reflection in our choice between ECS (Elastic Container Service) and EKS (Elastic Kubernetes Service). This two environement is very similar and both are linked to AWS services. The only reason to choose the EKS is the portability. If we want for example to migrate our scale MobilityDB environement from AWS to Azure in the future, it will be easy or vice versa.

1.2 Elastic Container Service and Fargate

The ECS is a containers orchestration or a control plane that manage the containers within the EC2 instances. This mode is pretty good because it do not manage the hosting infrastructure. If you want to delegate the host management to AWS services, you can use the Fargate service. The Fargate is a provision server or the capacity provider that provide you resources according to the container demands (CPU and memory). In another words it create automatically a server using the container demand. The advantage is we pay only what our containers are consuming.

1.3 Elastic Kubernetes Service

The EKS is a AWS service that allow us to manage a Kubernetes cluster on AWS ecosystem. The advantage of using the EKS instead the ECS is a portability of your Kubernetes cluster that mean if we want to migrate our cluster to AWS it will be very easy. Even to migrate it to Azure or Google Cloud infrastructure. Another advantage is the popularity of Kubernetes with a large community. The EKS is a control plane that can scheduling and orchestrating a cluster. When you create a EKS, AWS provision a Master node in the background linked with all AWS services as CloudWatch for monitoring, Elastic Load Balancer for load balancing, IAM for users and permissions and VPC for networking. Using the EKS service you can replicate the Master

node on other availability zone and regions (link to regions aws) In our deployment guide we have used the region Europe(Paris) eu-west-3, the 3 means that are 3 availability zones. After creating the Cluster(master node) we need to create a workers nodes with EC2 instances and join them to the cluster. EKS service allow us to semi manage the workers nodes using the node group option. If we want to fully manage our workers nodes, again we can use the AWS Fargate provision.

1.4 Citus cluster using the AWS EC2 instance

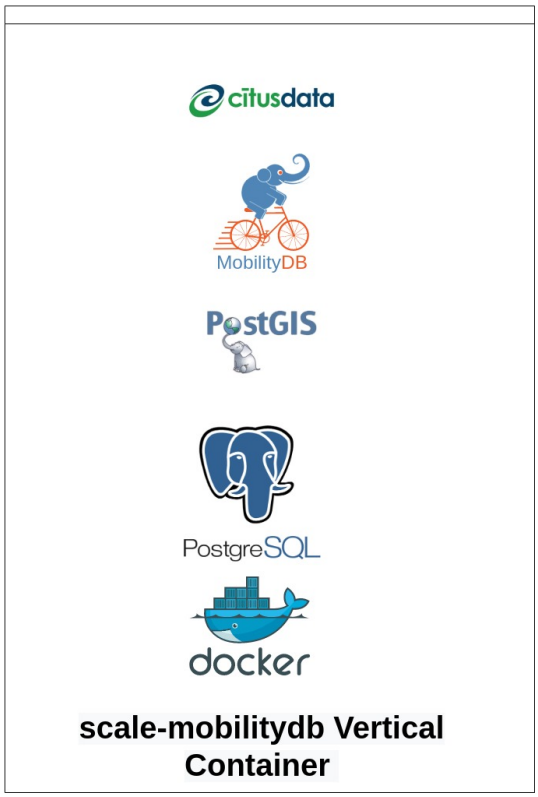
There is another way to scale our MobilityDB through the Citus cluster using Citus docker image. The idea is simply to create a group of EC2 instances, choose one of them as a Master node and join the worker nodes to the same Master node. Citus docker version provides 3 types of node, the Master, worker and the manager that has as role listening to new worker nodes in the same subnet in order to join them automatically to the Master node. This kind of deployment does not benefit from the AWS services comparing EKS or ECS.

1.4.1 EKS deployment architecture

In this part we will show you the different layers and components of our deployment in order to scale the MobilityDB using EKS service.

1.4.2 EKS vertical container view

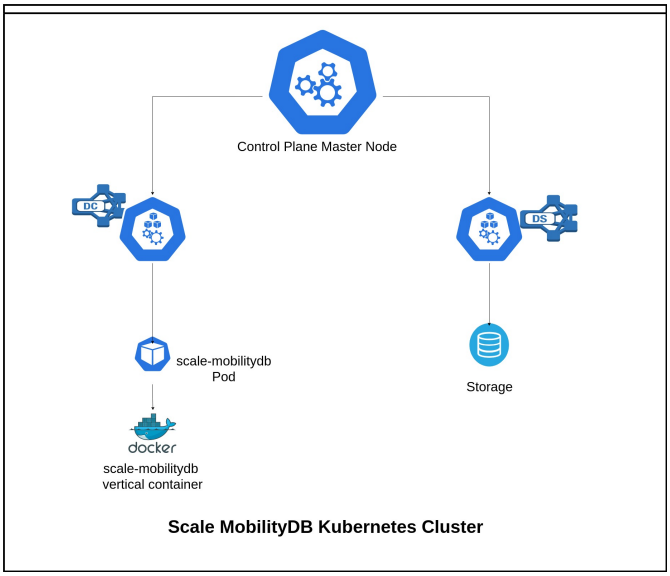
As you can see in the following figure we have prepared a docker image that contains the MobilityDB environment and Citus environment built on top of it.



1.4.3 Kubernetes Cluster view

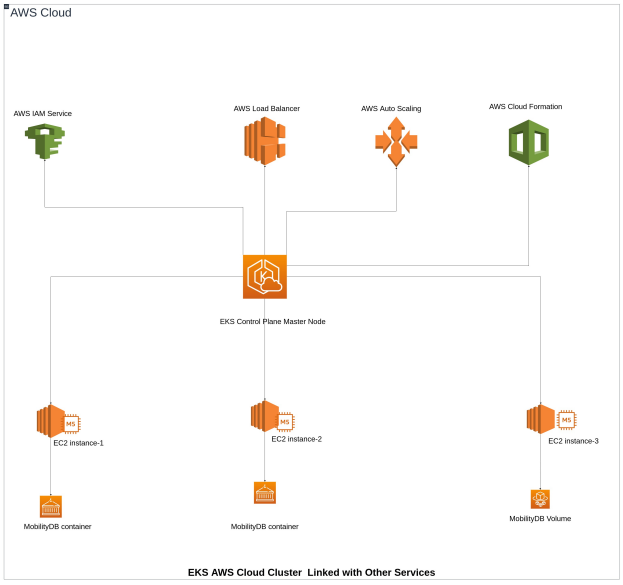
After preparing our MobilityDB scale, we can easily deploy it on worker node in Kubernetes cluster using the kubectl command. In the following figure we will show you a small Kubernetes cluster that make MobilityDB scaling. We have a control Plane or the Master node and two workers nodes. The worker in the right it can be seen as storage node, the DS stand for dense storage.

The worker in the left it can be seen as compute node, the DC stand for dense compute. As you can see a pod is created within the worker node using the scale-mobilityDB container prepared. Once you have configure this Kubernetes architecture, you can deploy it in any cloud service platform that provide Kubernetes. Maybe you are asking a question why there is a storage node?. In my opinion the data need to be loaded within a cluster before using the MobilityDB queries. Else another AWS services may be explored as EMR for Elastic MapReduce and AWS Apache Spark or AWS S3 for Simple Storage Service to make web-scale computing easier.



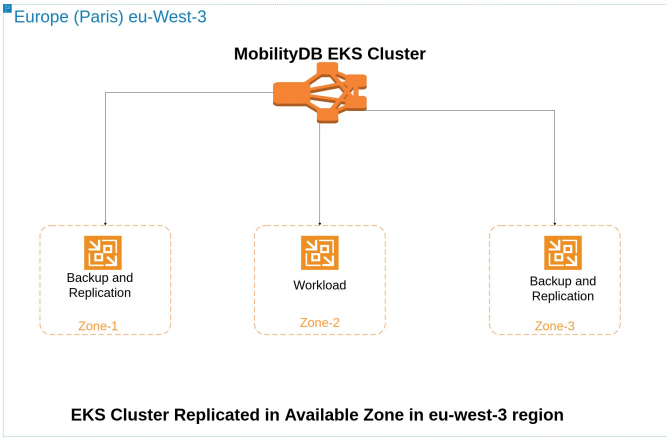
1.4.4 EKS control plane view

In the following figure we have a Kubernetes cluster created on AWS EKS service using eksctl command. THere is a basics parameters needs to pass on eksctl command to create your cluster as which region want to deploy, how many replication in availability zone, the number of node and type of node. Once the EKS cluster is created, all the configuration between nodes and control plane as connectivity, default volume used, EC2 instance creation is automatically set up in the background. In addition the link to other AWS components is set with you cluster like the IAM service for Identity and Access management and Cloud Formation service used to manage the life cycle of AWS resources. We can configure other AWS services with the EKS like Load Balancer used to distribute incoming traffic across multiple targets, such EC2 instances, containers, IP addresses.



1.4.5 EKS cluster replication view

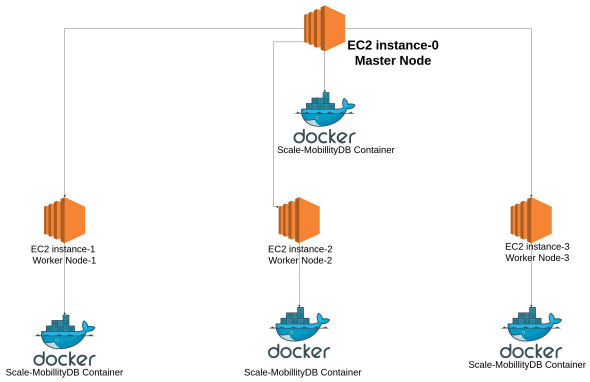
In this figure we have our MobilityDB environment replicated in all available zones in the Europe (Paris) eu-West-3 region. We can also deploy our application in other regions as Europe (Frankfurt) eu-central-1 or Europe (Milan) eu-south-1



EKS Cluster Replicated in Available Zone in eu-west-3 region

1.4.6 Scaling using Citus Cluster view

In this part we defined a simple cluster of machine (EC2 instances) created by hand on AWS cloud as on premise version. So we have deployed the scale-mobilitydb in all nodes and we choose one from them as Master and then make the others as worker by joining them to the Master by the sql command citus_add_node('IP/host of new worker', 5432);



Scaling MobilityDB using Citus daya cluster