



# Lab1: Cloudhub 2.0 Features, capabilities, and Architecture

* Mulesoft has recently announced the CloudHub 2.0 to simplified the developer experience for deploying and running the application in the containerized environment based on the Kubernetes
* CloudHub 2.0 is a fully managed, containerized service for deploying and running the Mulesoft Application. With Cloud 2.0, developers can easily deploy and run the application in the resilient and easily scalable environment

## Cloudhub 2.0 features:

* Applications can be deployed in private or shared space

**shared space:** Deploying the applications in multi-tenant mode, means all the tenant of the mulesoft is sharing the same space for deploying their applications, it doesn’t mean all the data will be shared across the tenants

Ex: If am a tenant I can see the data of only my applications but not the others

* Container-based application isolation

1. Whenever we are deploying the applications in the shared space they will be isolated with each other, each application will be running in the separate container, we cannot deploy more than one application in the one container
2. Incase if we dont want to run the application in the multi-tenant mode, If we want to run the application in the private space or in a single-tenant mode then we need to choose the private space

* CloudHub 2.0 is available in the 12 regions globally
* Application can be deployed on multiple replicas to enable high availability and fault tolerance

1. It means we are deploying the two instances of the application in case one instance goes down we still have a second instance running till the previous instance automatically gets recovered, but we still have a one instance availability
2. Fault tolerance, whenever we are deploying multiple replicas, it is distributed across multiple availability zone incase one of the availability zone goes down still second replica running in some availability zone, and your application reflash will be successfully fulfilled. In case any availability zone goes down the replica will try to deploy himself or deploy itself in other availability zone

* More granular vCores option. The maximum vCores can be allocated per replica is vCores

1. If we need total 8 vCores for one replica, we can create a two replicas for that application allocate 4,4 vCores to both replicas

* Application can be deployed on Clustered mode. Minimum 2 replicas required for clustering

1. Basically for enabling the anypoint clustering for your application we need two replicas

* Support for the rolling update and Recreate while application deployment

1. If we want the application to deploy in zero down time you can make easy of rolling update. If we want to recreate it, it will remove the application and redeploy it

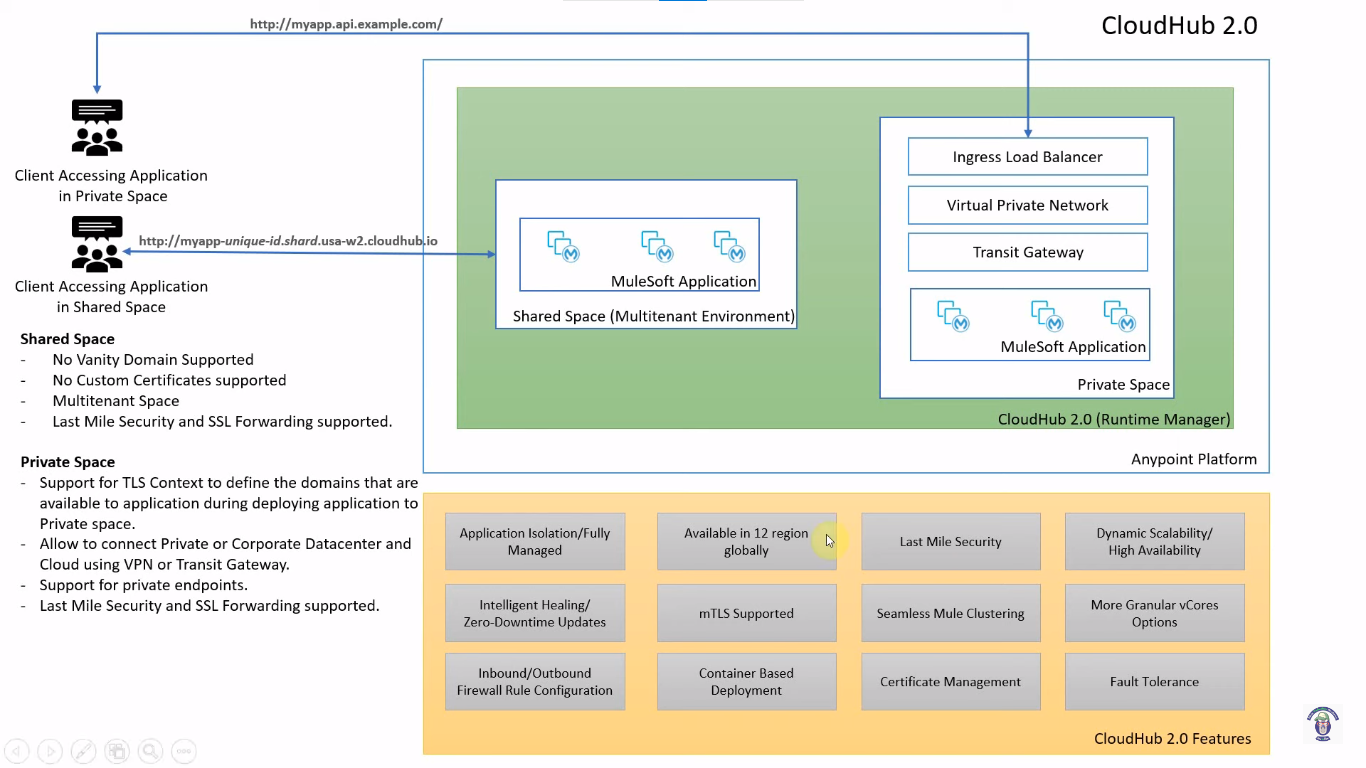
Incase of rolling update, it will keep the older version of application till new version of application is not deployed. So, our request will not filled

* Easily/Dynamic scalable. Horizontal scaling can be easily achieved by increasing the number of replica and Verical scaling can be easily achieved by increasing vCores size
* Supports features like intelligent healing, zero downtime updates etc

Intelligent healing means if the replica got crashed, it tries to recover itself automatically, it will try to restart or it will try to deploy a new replica automatically

* CloudHub2.0 stores upto 200MB of log data per config, or upto 30 days whichever limit is reached first





## Shared space:

Shared space enables you to run the mule instance in the multitenant environment. CloudHub 2.0 provides one shared space in each region to deploy your Mulesoft application

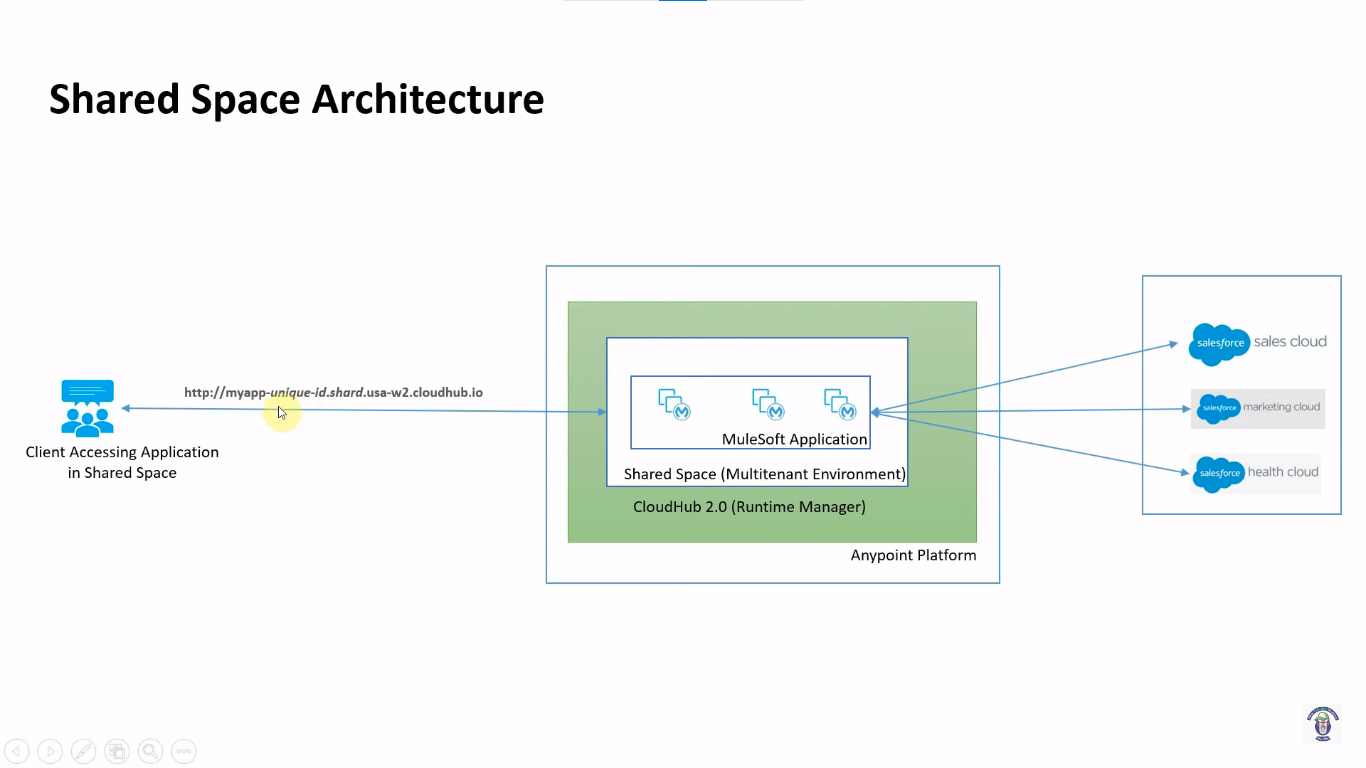
**When to use Shared space:**

* You don’t have requirement for using vanity domain and can continue to use cloudhub.io
* No custom certificate requirements
* Your application deployed in shared space don’t want to access resources or services or databases located in the On-Premise Datacenter or Private or PublicCloud
* You don’t have requirement of running application isolation from public cloud

**When not use shared space:**

* You want to run your application in single-tenant environment or in isolated mode
* You have requirement of the vanity domain
* Need of custom certificates
* Need of private endpoints
* You need to connect to the resources or services, or databases located in the on-premise data center or private or public cloud

Example: Let us consider I had deployed one application in the cloudhub 2.0 and that application has to connect with the database and that database is located in the corporate data center, so we cannot directly access the database from the cloudhub 2.0, first we required a private space(nothing but a anypoint Virtual Private Cloud(VPC)). Once you deploy the application into VPC, we need to setup a VPN tunnel between cloudhub and corporate data center to connect or access the database located in the corporate data center

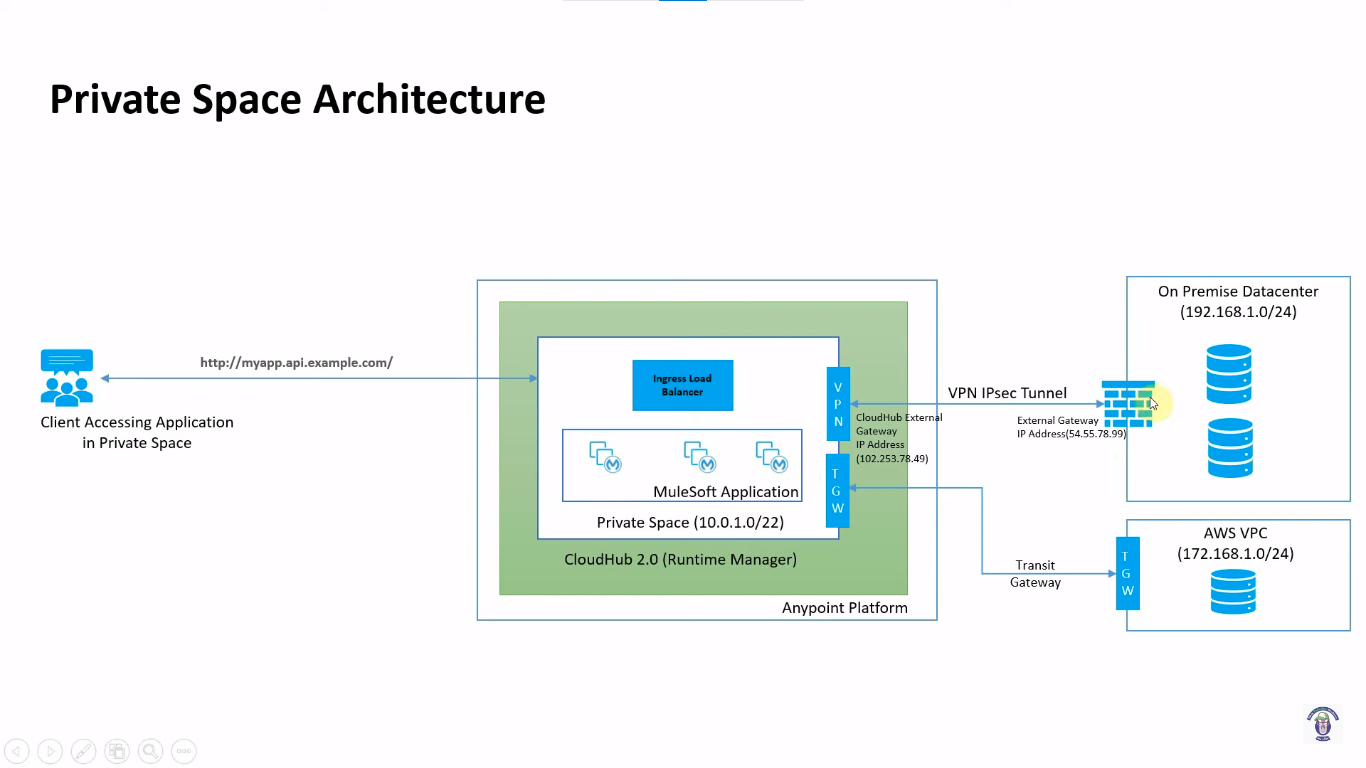


## Private Space:

Private Space is like Anypoint Virtual Private Cloud in Cloud Hub. Private space is virtual, private and isolate space in the CloudHub 2.0 to run your mulesoft application. You can create multiple private space in single or multiple regions in CloudHub 2.0. One of the mandatory requirement for creating the private space is range of IP addresses for the apps in your private space to use

**When to use private space:**

* You want to connect one or more resources or services, or databases located in the on-premise data center or private or public cloud. You can use anypoint VPN or Transit Gateway. There is no support for AWS direct connect and VPC peering in CloudHub 2.0
* You can enable private endpoints to accepts traffic from private locations
* You can configure the custom certificates and your own vanity domains
* You can configure inbound as well as outbound firewall rules



## Features did not support by Cloudhub 2.0:

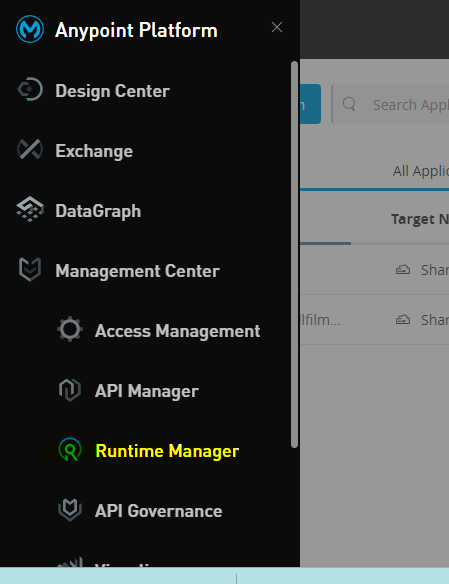
* Anypoint Datagraph
* VPC peering and AWS direct connect’
* Persistent VM queues
* TLS1.0
* Built-in and custom notifications(Cloudhub connector)
* URL rewriting

## Why Cloudhub 2.0?

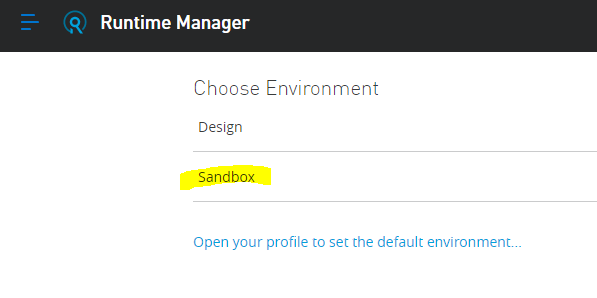
* CloudHub 2.0 is using Kubernetes based architecture, every single applications is deployed in lightweight container, so it can easily scale up and down easily.
* No need of managing dedicated load balancer as CloudHub 2.0 provides built-in ingress load balancer with auto-scaling capabilities in the private space
* CloudHub 2.0 provides option to configure inbound as well as outbound firewall. This will enable us to block or allow traffic from specific ports
* Provides inbuilt security policies that secured the services and sensitive data with encrypted secrets.

# Lab2: Deploying Mulesoft Application on CloudHub 2.0:

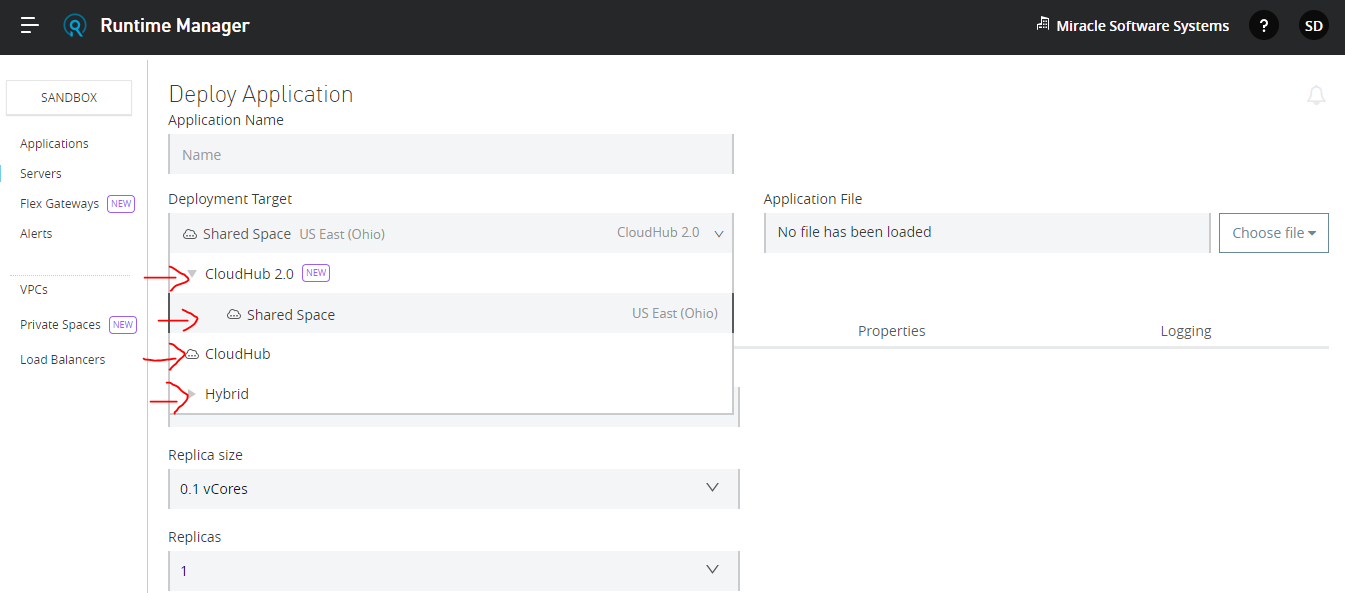
**Step1:** First we need to go to the Runtime Manager



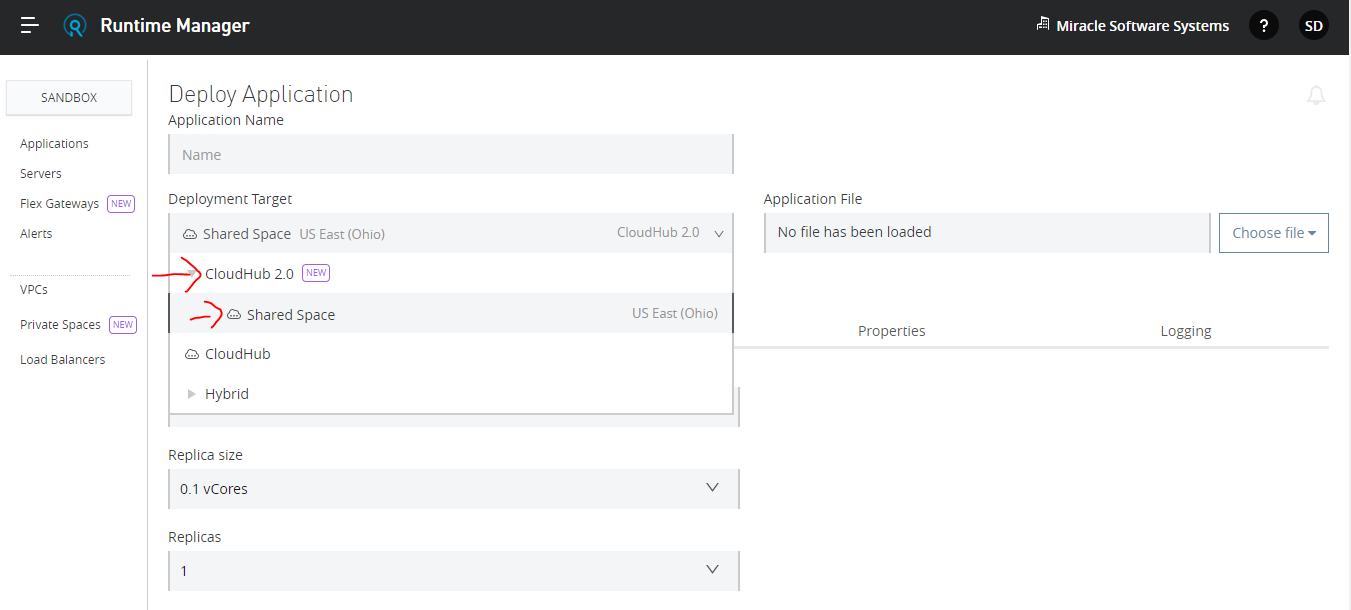
**Step2:** Choose the environment and select Sandbox



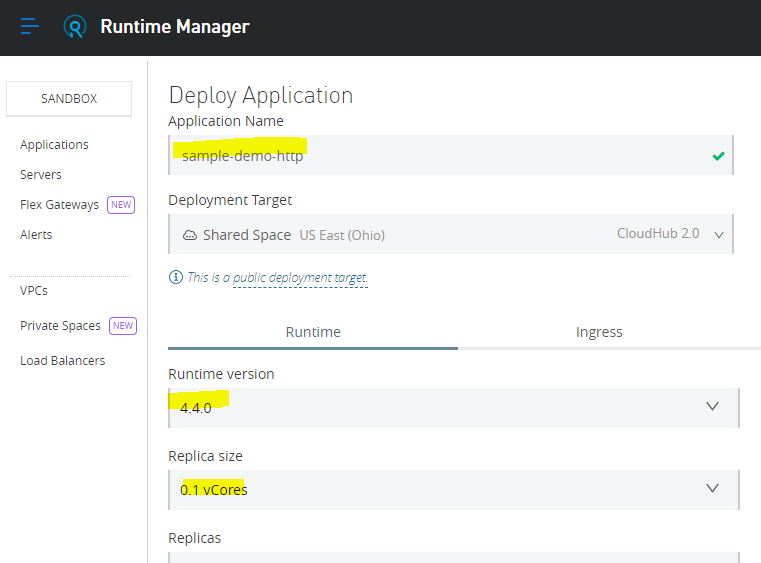
**Step3:** We can see here multiple deployment targets



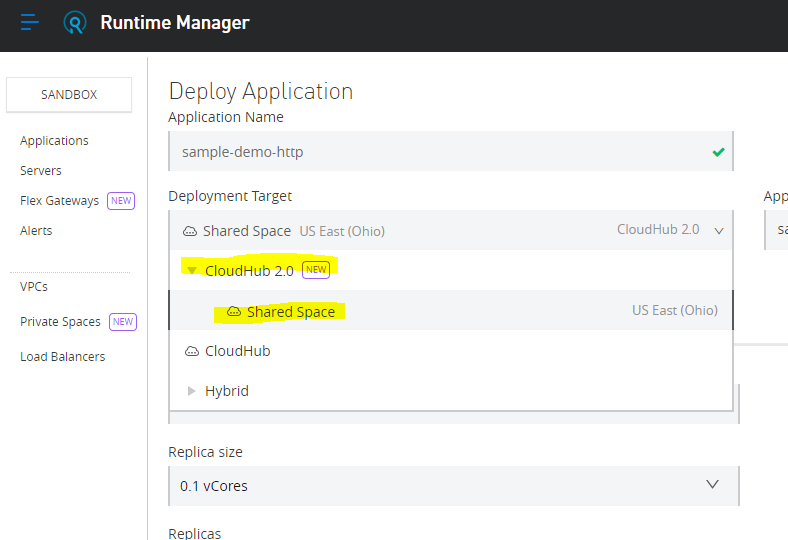
**Step4:** For now, we need to deploy the application into CloudHub 2.0 for trial account it has only one region, by default it has 12 regions



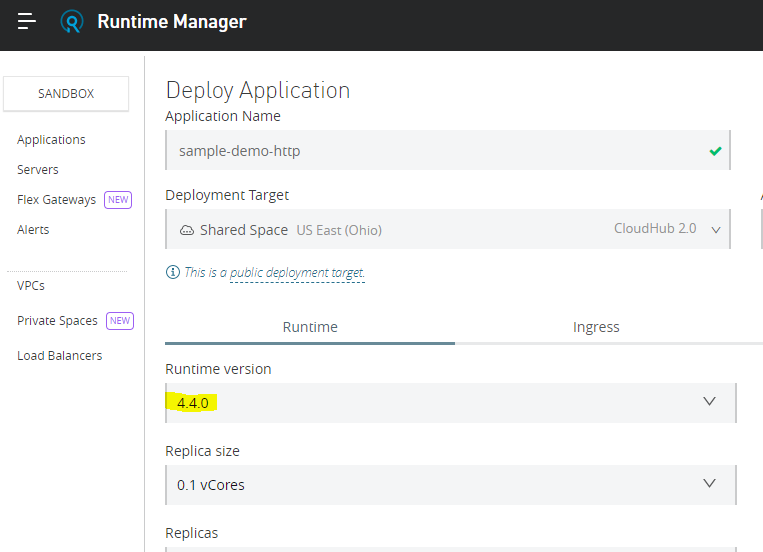
**Step5:** Here we need to give the application name



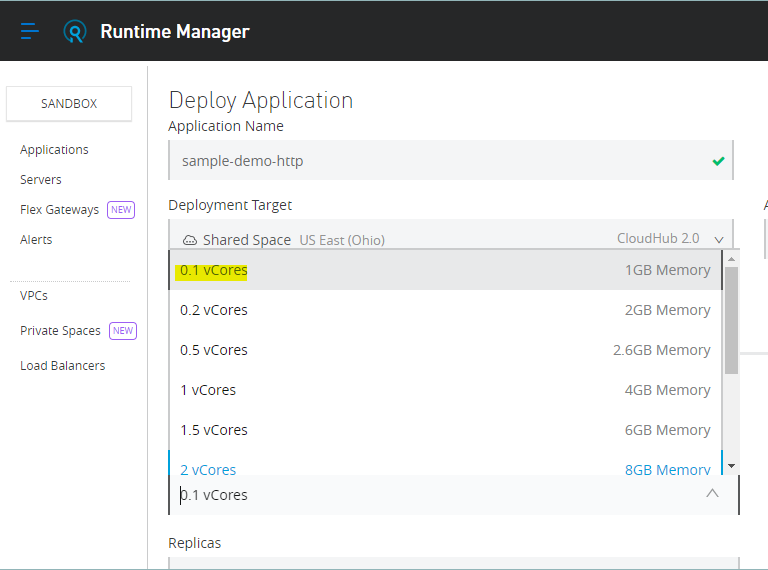
**Step6:** We need to select the deployment target as Cloudhub 2.0



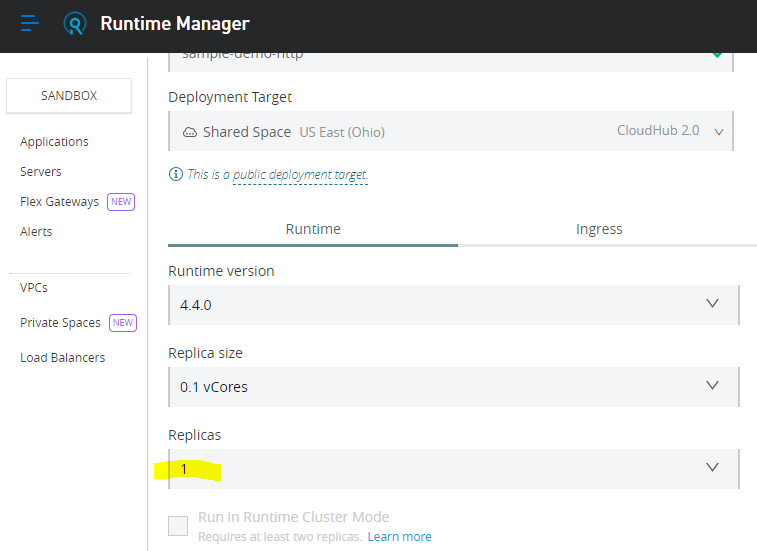
**Step7:** Here we need to choose the Runtime version



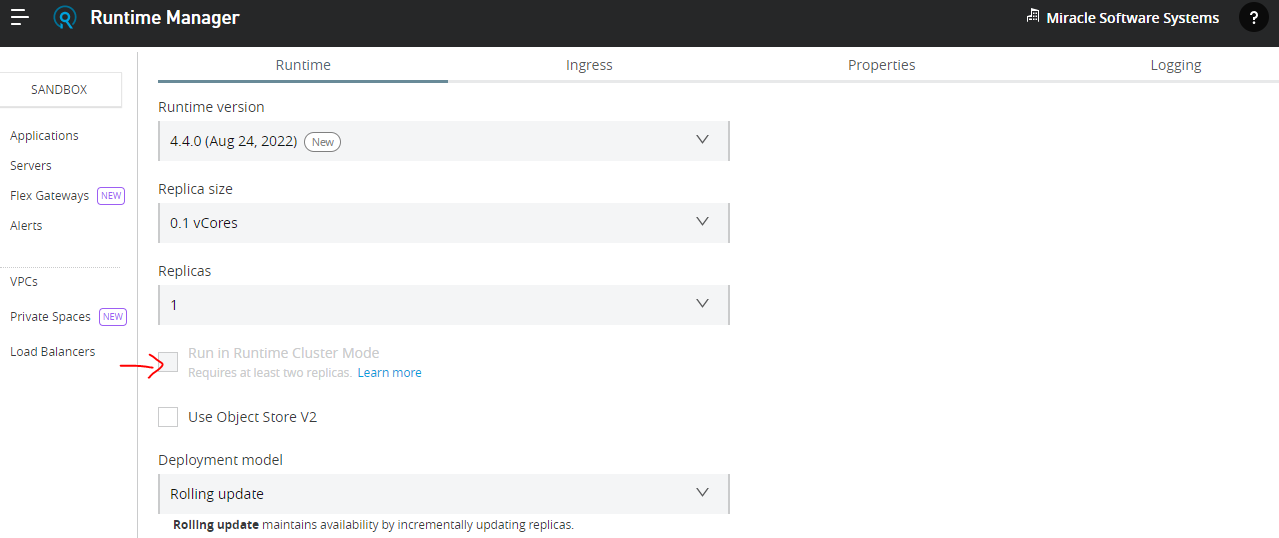
**Step8:** Here we need to select the replica size, for one replica we need to give max of 4vCores



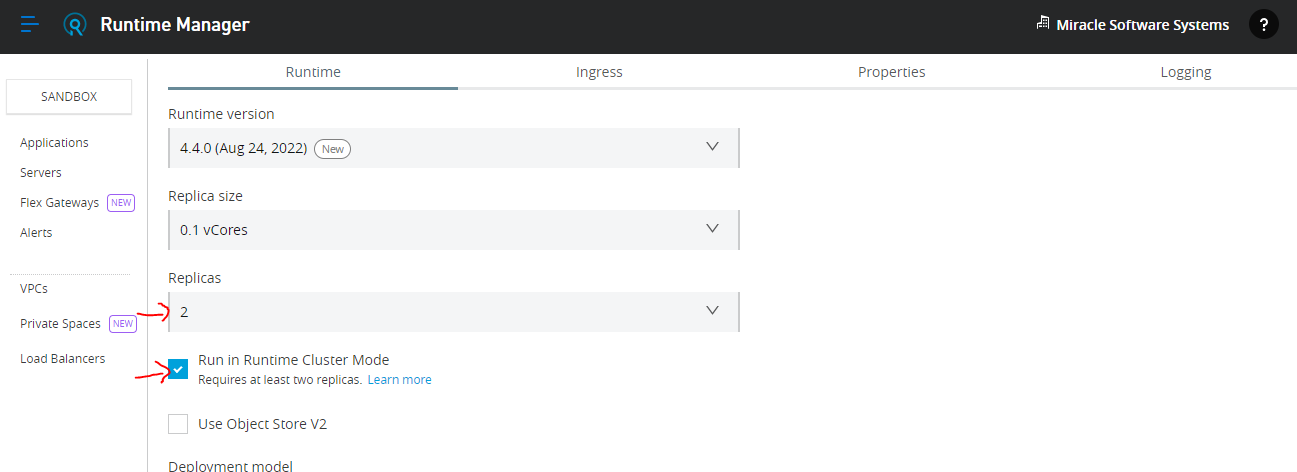
In this example, we are selecting one replica



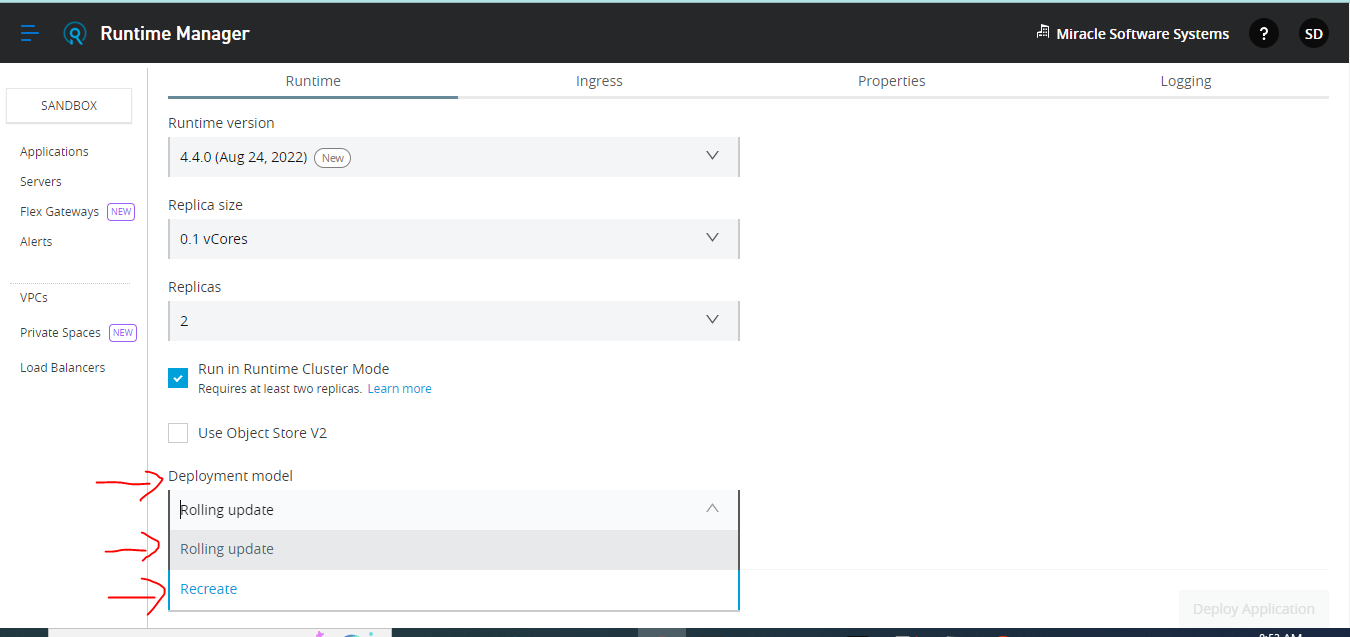
To enable this option, we need at least two replicas



After giving the replicas as 2 the option(Run in Runtime Cluster mode) will be enabled

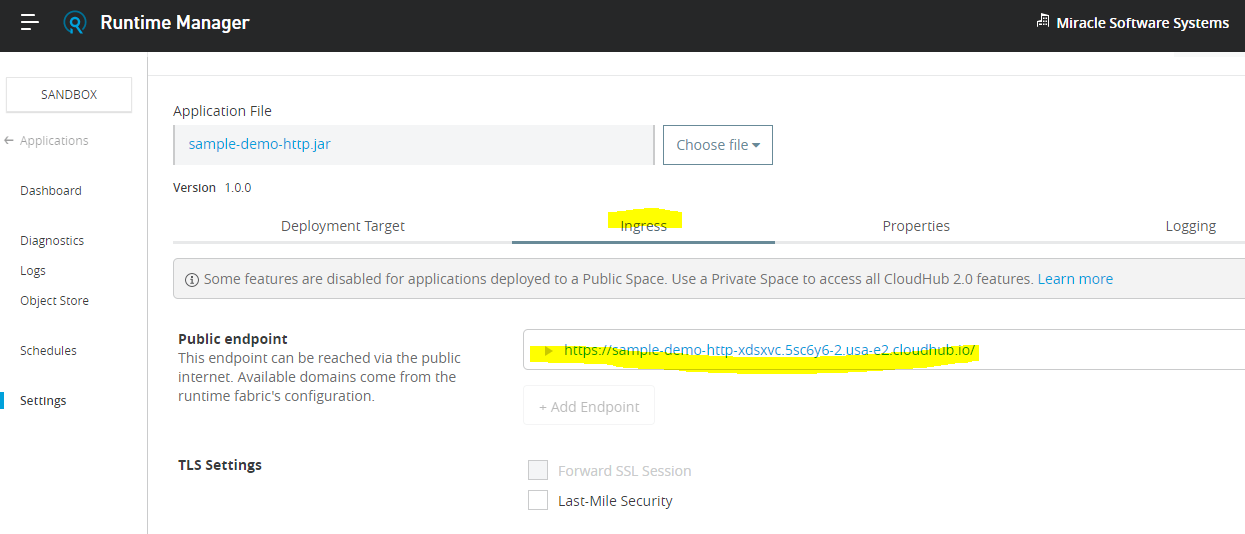


**Step9:** Here we need to choose the deployment model

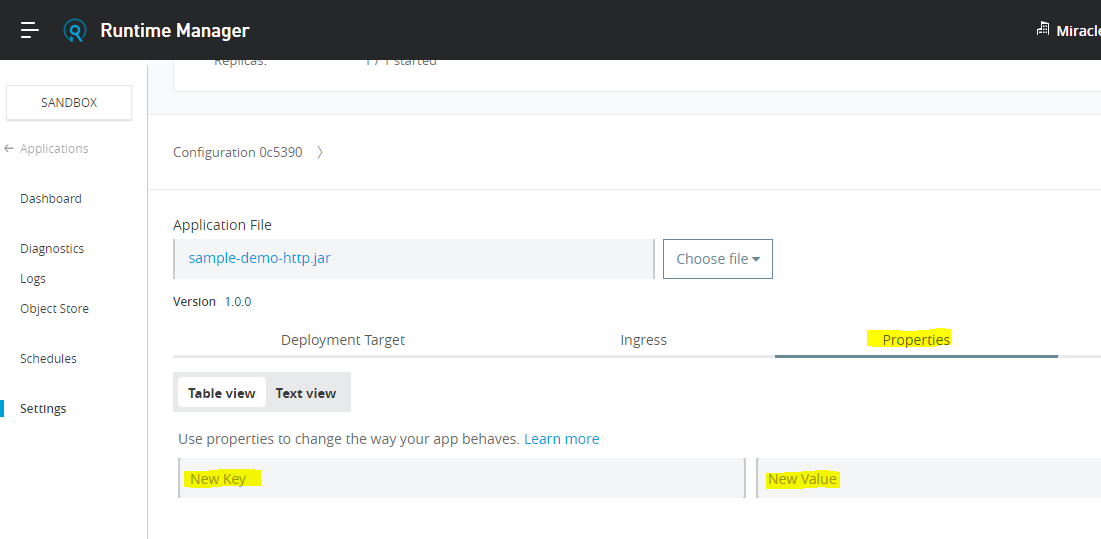


Rolling update means incase you are doing redeployment, it would happen with zero downtime updates, In case of recreate It will remove and stop the application and try to install the new version of the application.

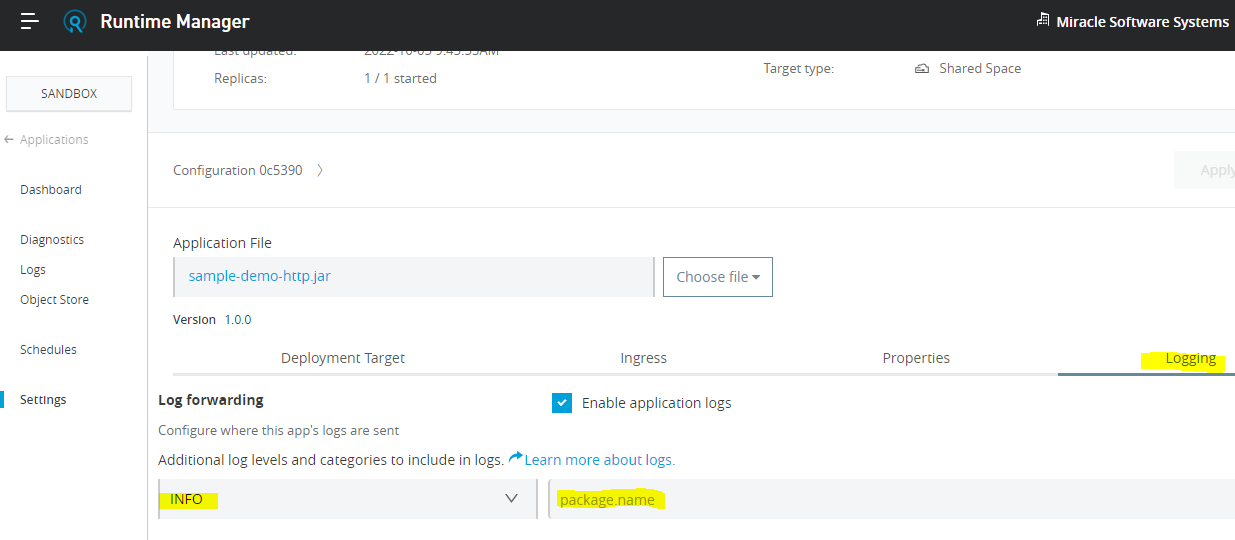
**Ingress:** Once you deploy an application It will give the endpoint which is public that we can use to access the application



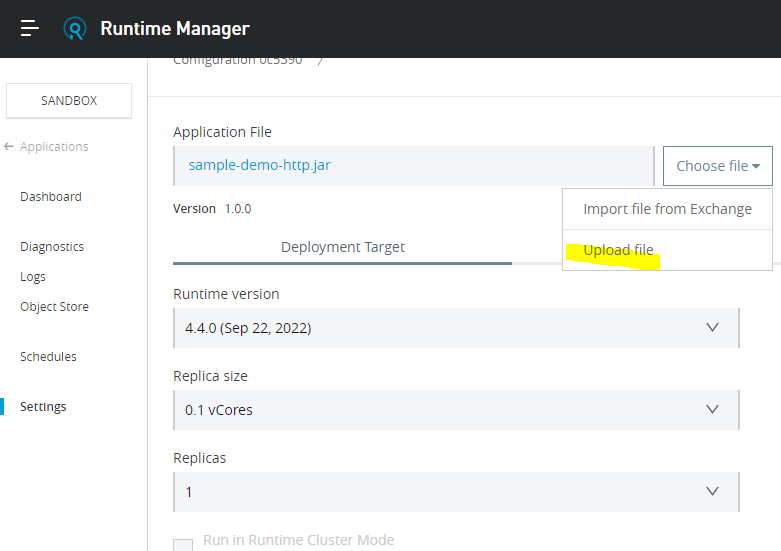
**If we have properties like runtime properties like key value pair , then we need to define here**

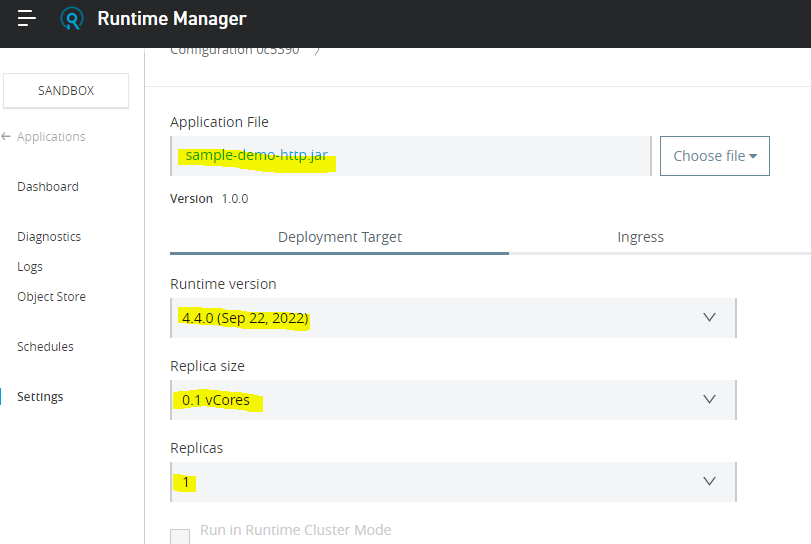


**If we want to apply Logging, we can apply here**

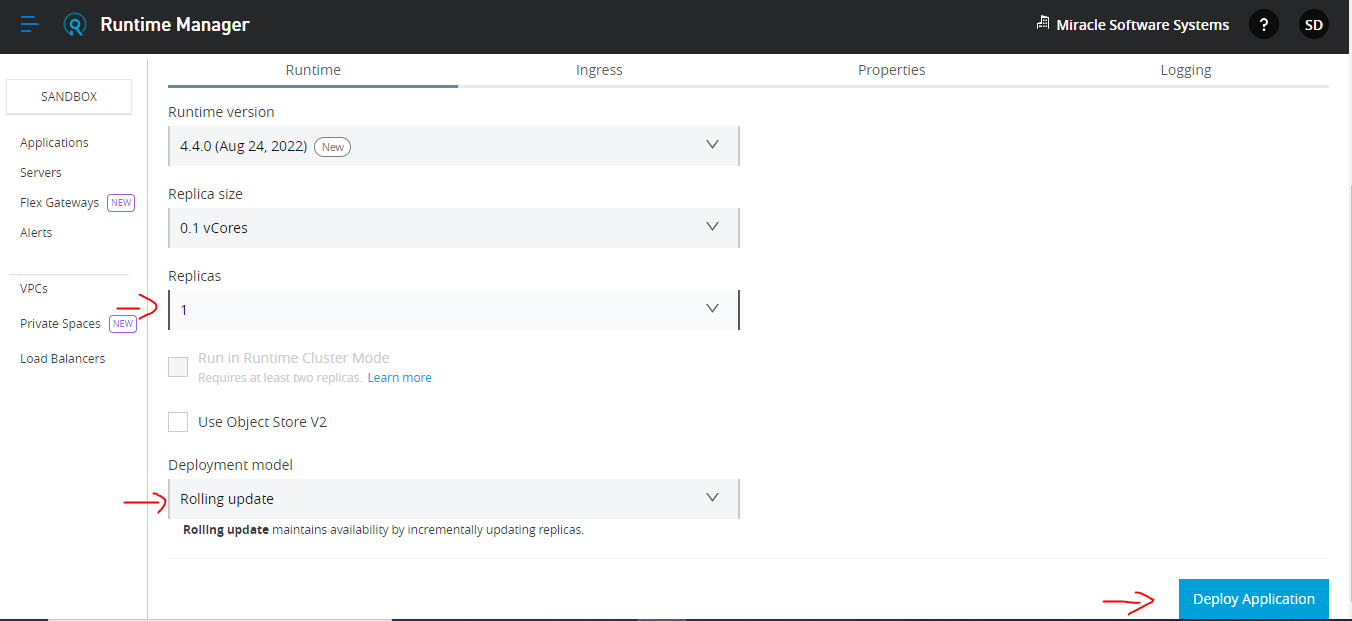


**Step10:** Here we need to choose the file

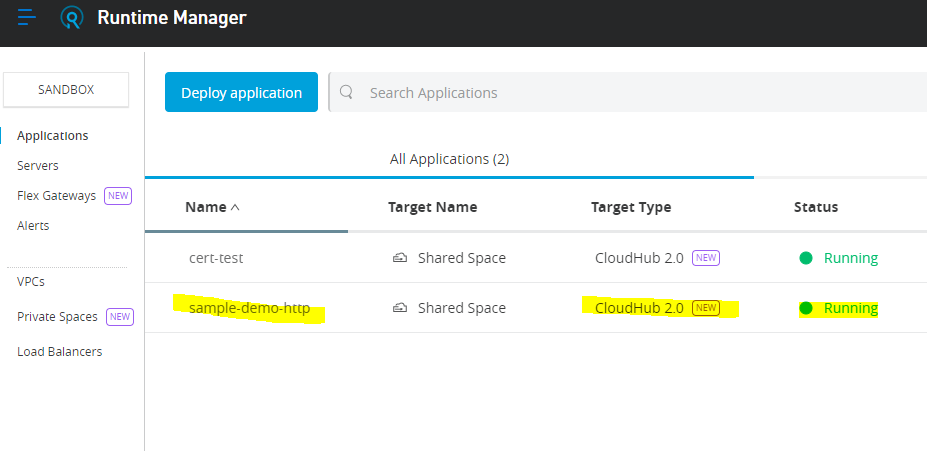




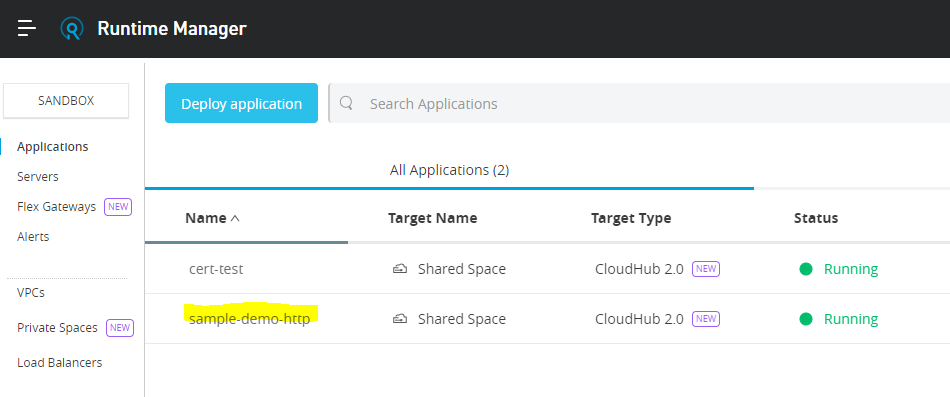
**Then we deploy the application into the CloudHub 2.0**



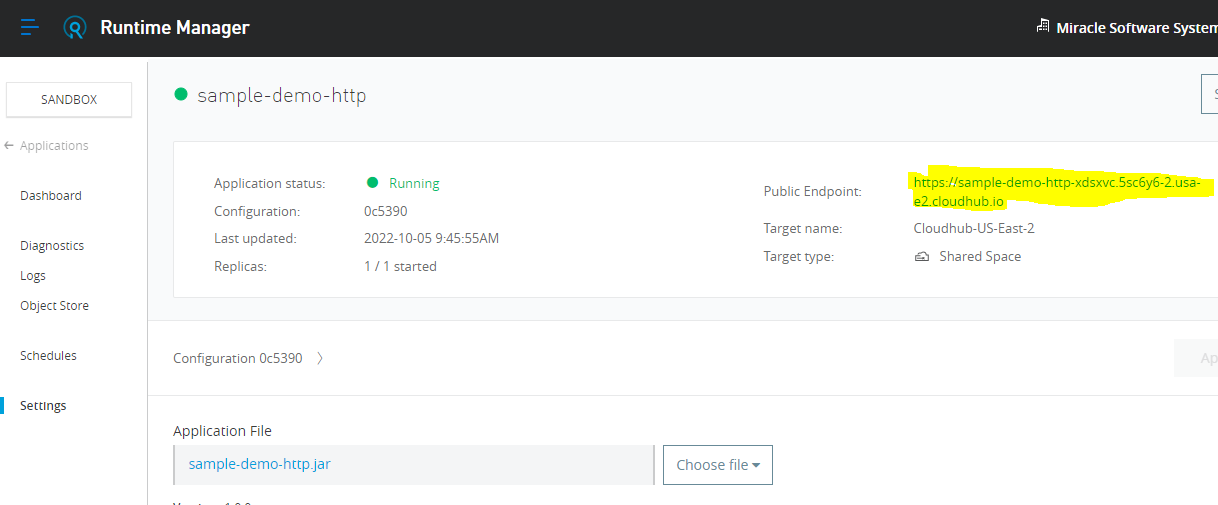
**Step11:** After successfully deployed into the Cloudhub2.0 it will show like this



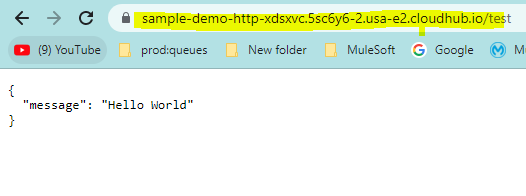
**Step12:** If we need to access this application click on the application name



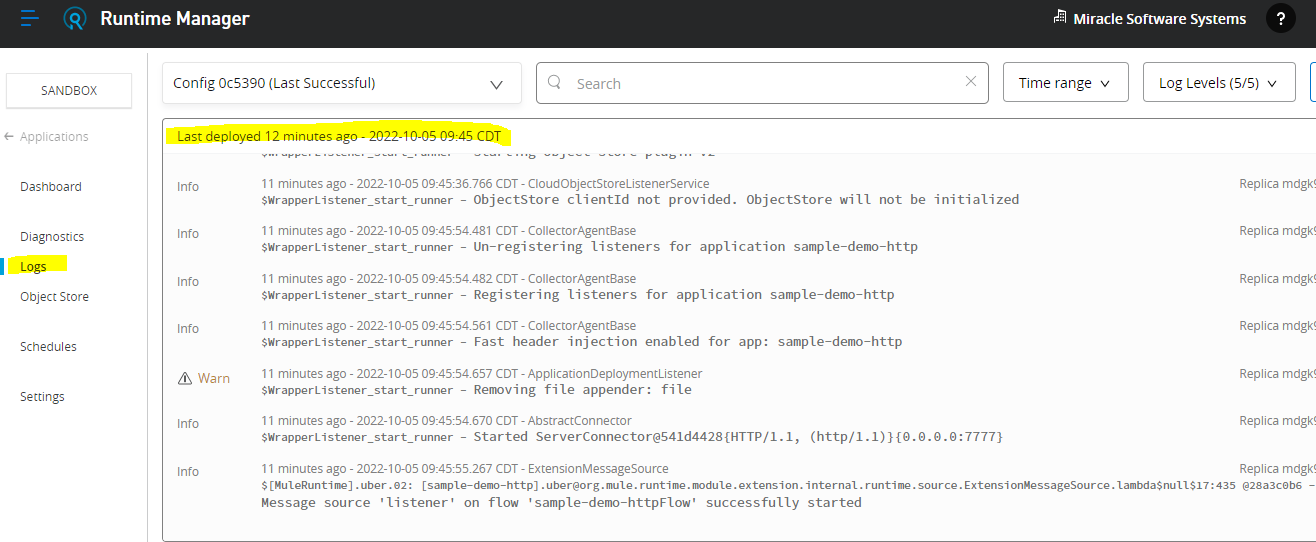
**Step13:** Here we can see the endpoint, by using this endpoint we can access the application. The endpoint consists of the application name, and some generated code along with the region



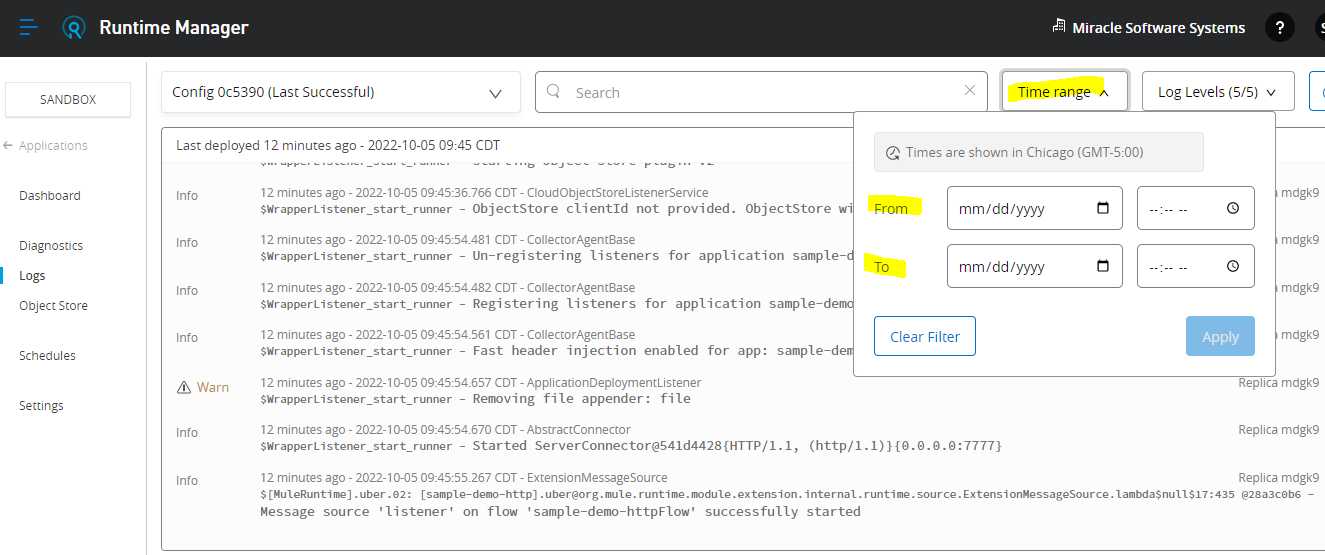
**Step14:** By using the testpoint we can get the response from the application



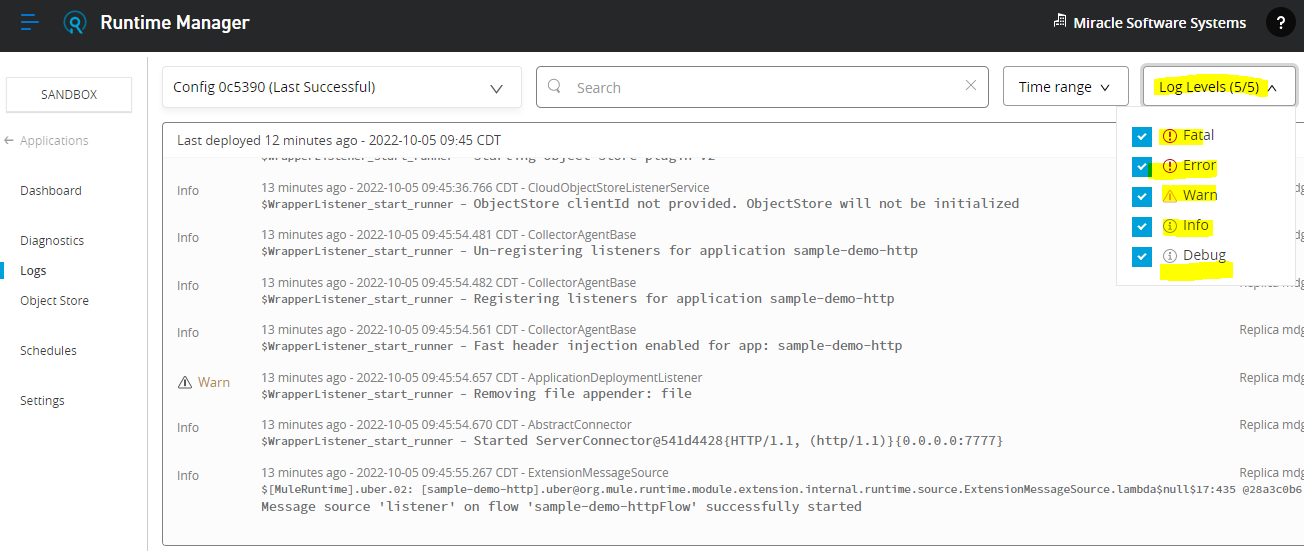
**Step15: We can check the logs here**



**Step16: We can filter the log levels based on the time**

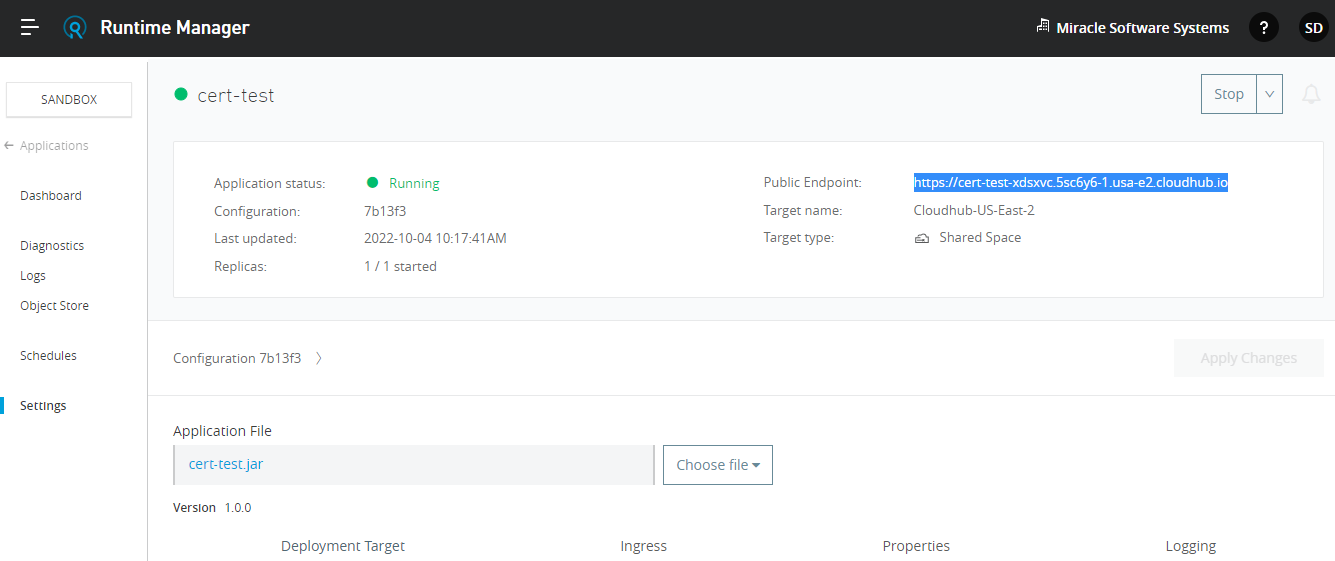


**Step17: We can also filter based on the log levels**

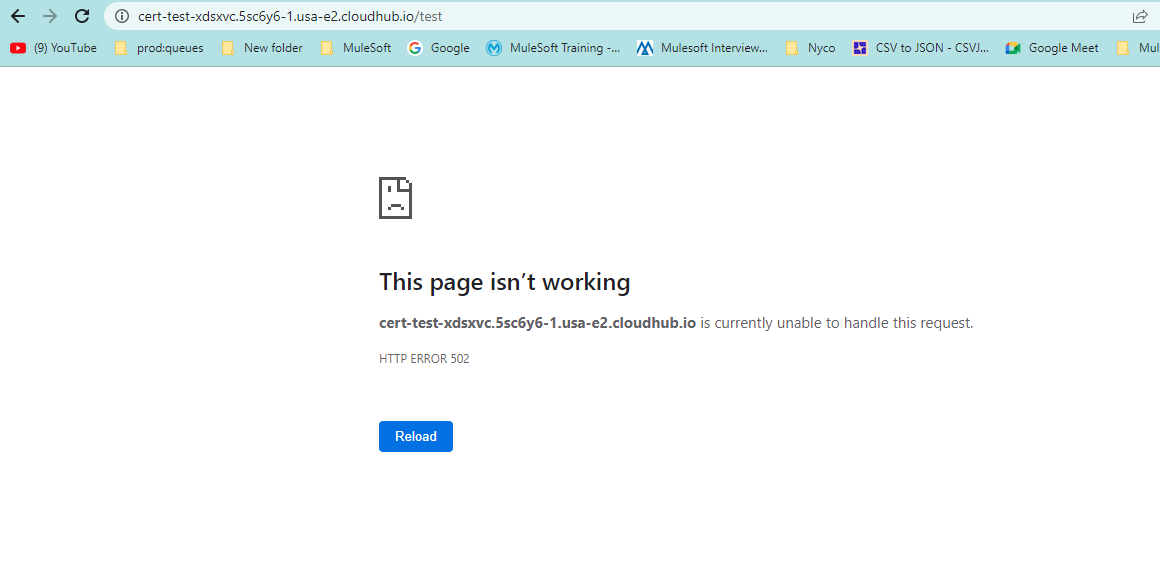


# Lab3: How to enable last mile security in CloudHub 2.0

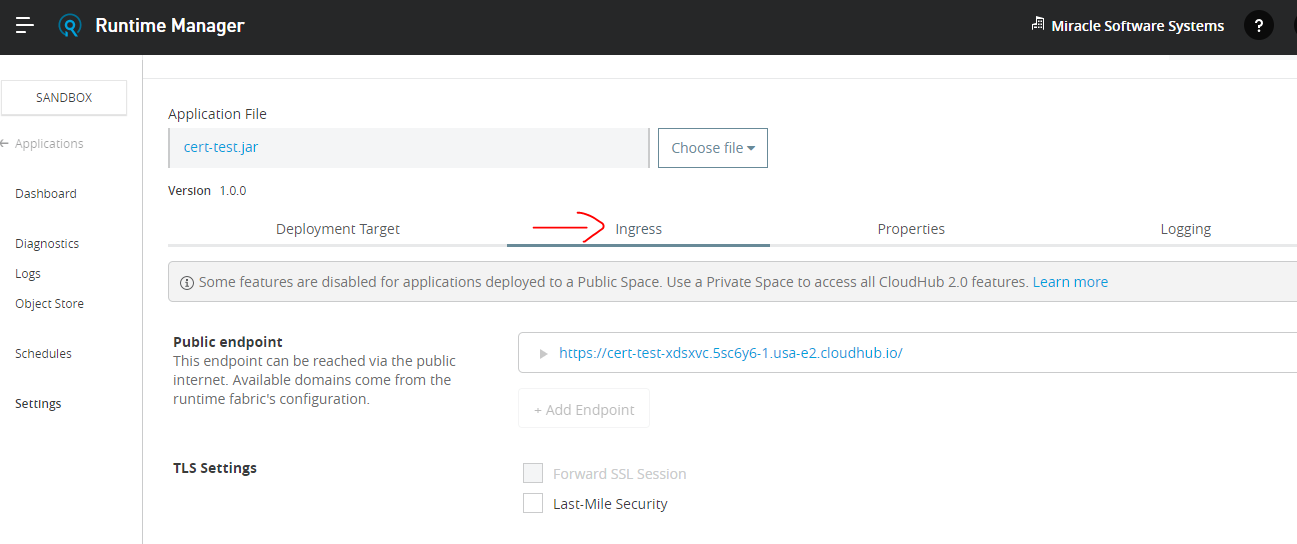
Port number 8081 is same for both the HTTP and HTTPS in cloudhub 2.0



**Step1:** After testing it was showing failure because of the last mile security option was not enabled, The application is failed because it is having the HTTP listener

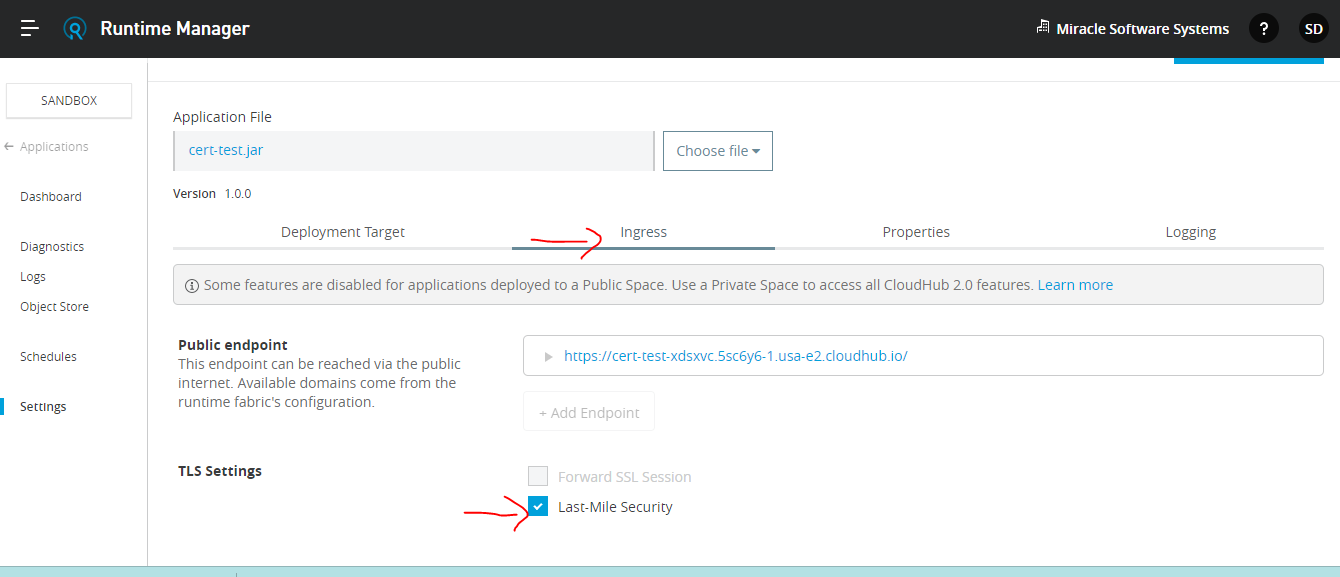


**Step2:** Ingress had tried to forward the traffic on the application on the HTTP port but my listener is listening on the HTTPS port 8081, for cloudhub application, we always use port 8081

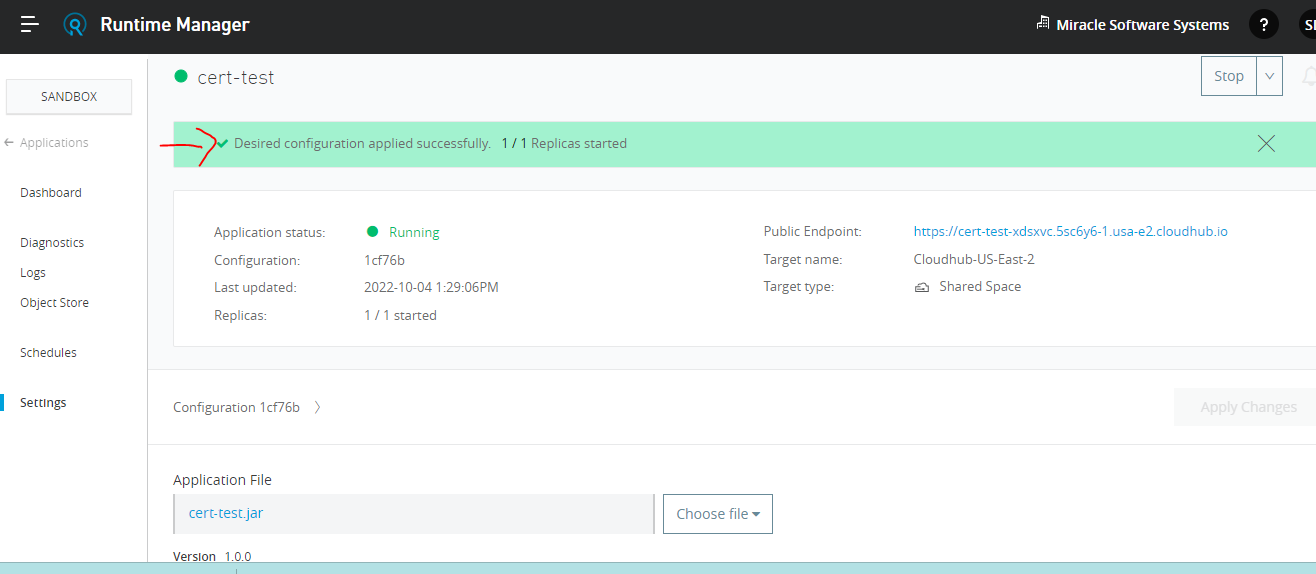


**Step3:** Why we need to enable the last mile security means because to understand the ingris, whatever the application we deployed the cloud hub 2.0 that having the HTTP listener so you have to forward the traffic on HTTP listener , so the request will come on ingris from the HTTPS the traffic will be forwarded from the HTTPS

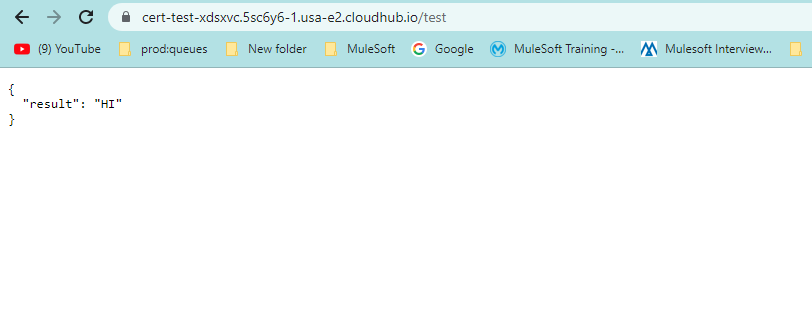
**Step4:** We need to enable the option Last Mile Security



**After we made the changes successfully**



**Step5:** After testing from the endpoint we will get the desired result.



# Lab4: Deployment model(Rolling Update and Recreate)

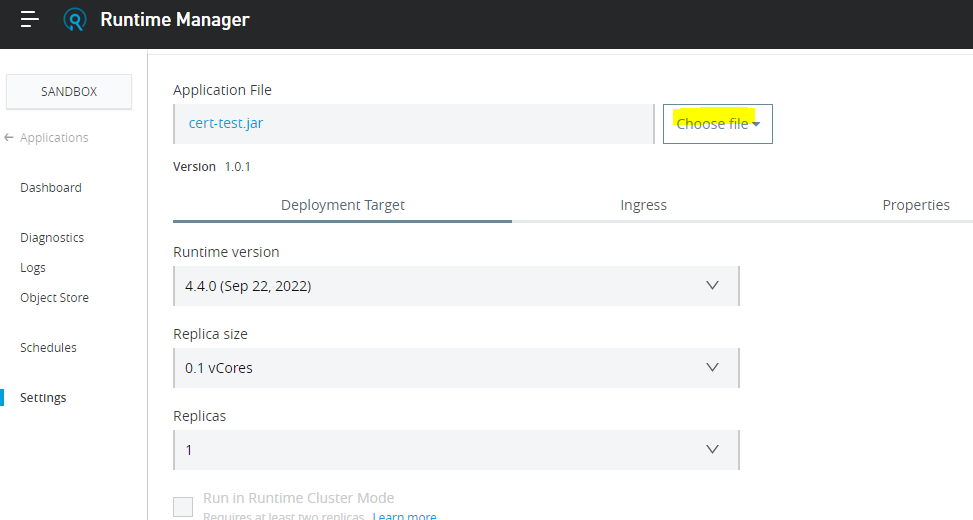
**Rolling Update Model:**

When we select rolling update while deploying or redeploying your application it will ensure a zero downtime deployment, it means till the new application deployment doesn’t end or doesn’t over it will keep the whole instance of your application and consumer can access that old instance of the application once the new application has been created the old instance will be removed and the request will come to the new instance so that is how we can do the zero downtime

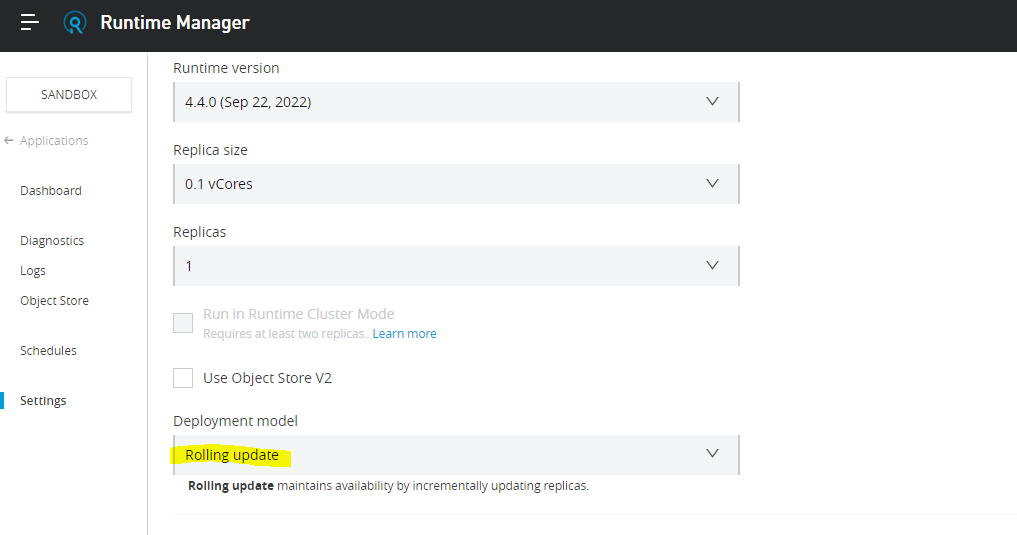
**Recreate Model:**

Incase of recreate what will happen means so when you are redeploying or deploying a application using deployment model recreate your application will be unavailable and it will be not reachable till the deployment get completed

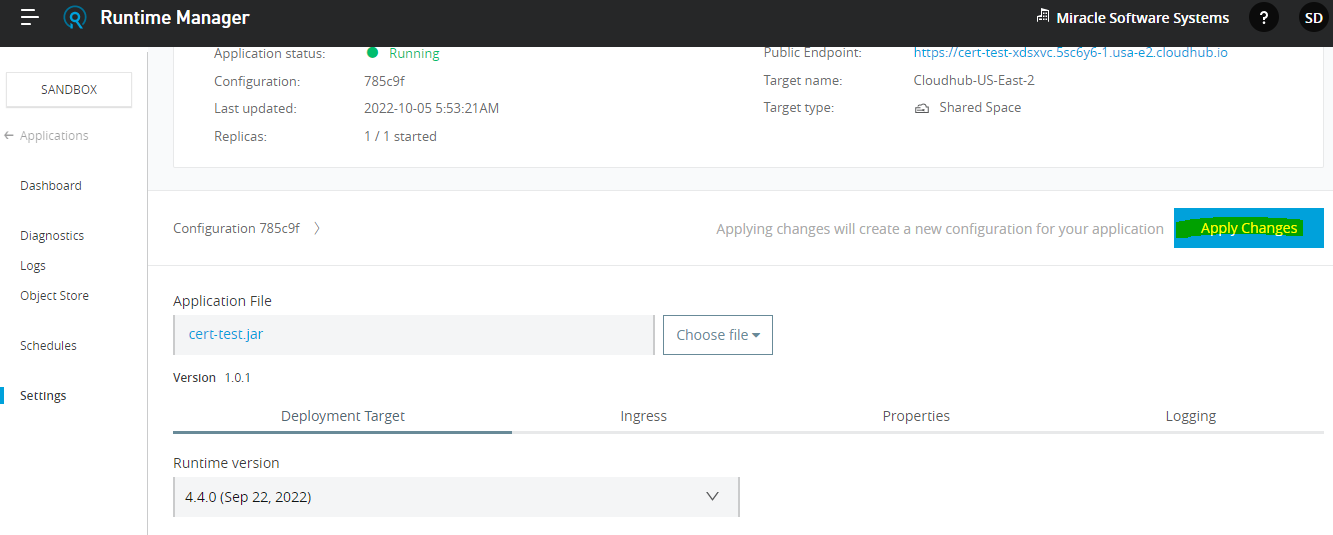
**Step1:** Choose file Application jar file which we want to deploy



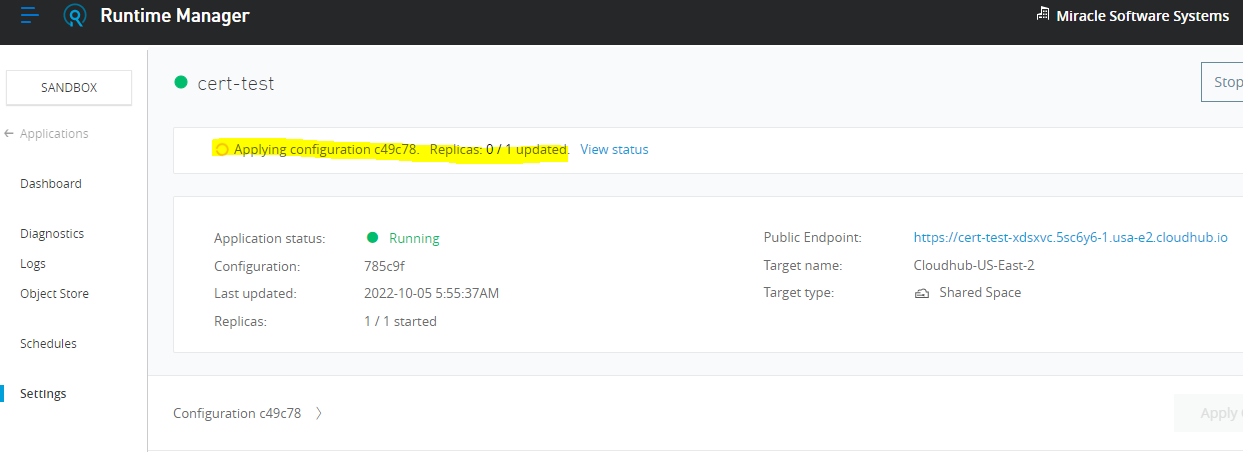
**Step2:** Select the option Rolling Update

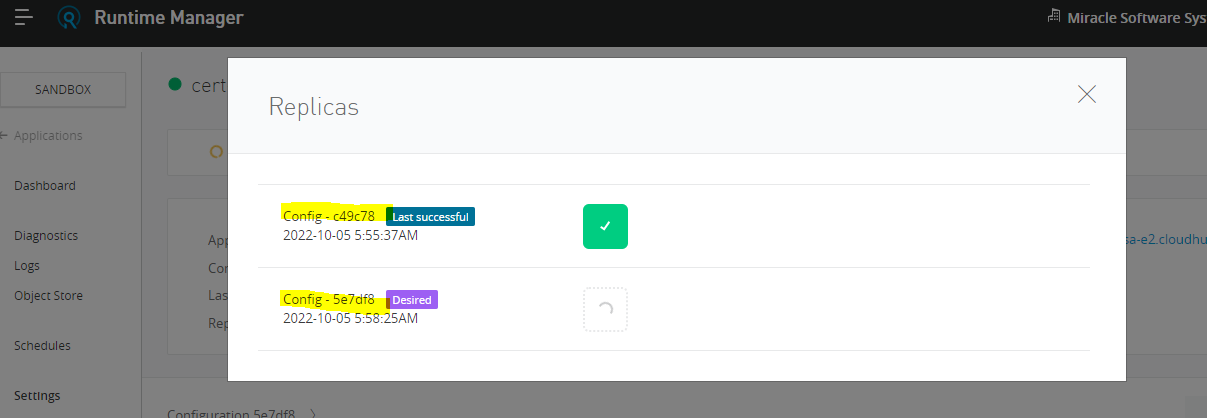


**Step3:** We need to click the apply changes

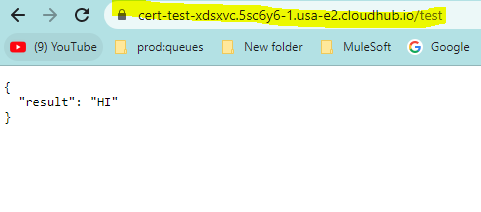


**Step4:** The application is in running mode





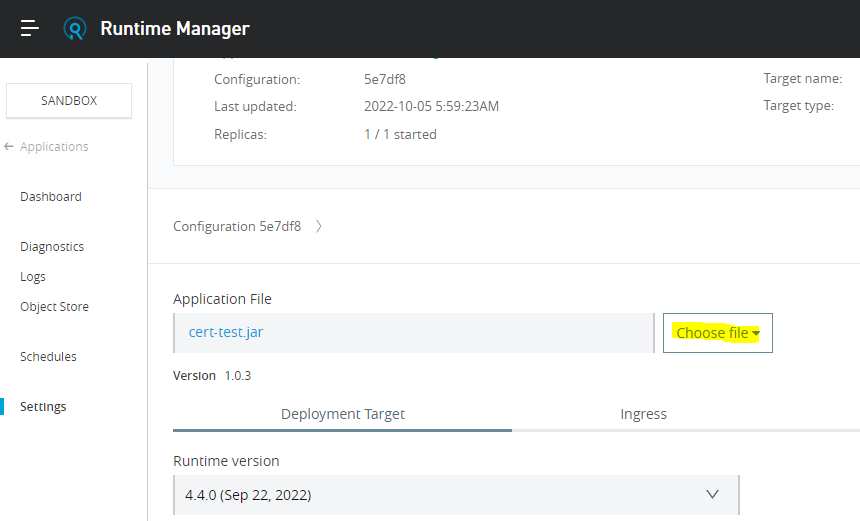
**Step5:** But when we test the application by using the endpoint we will get the response from the older instance of our application



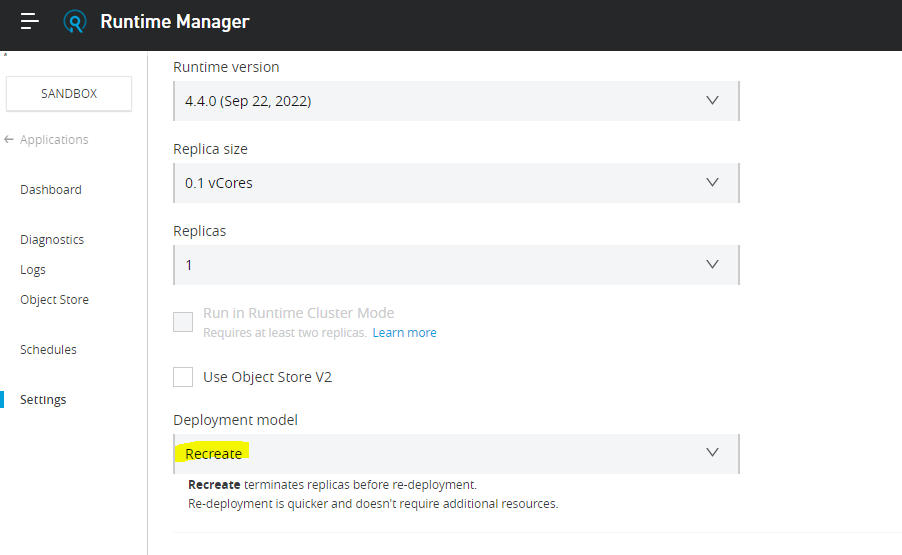
When we select deployment model as a rolling update you can do a deployment as a zero downtime updates

**Incase of Recreate Model**

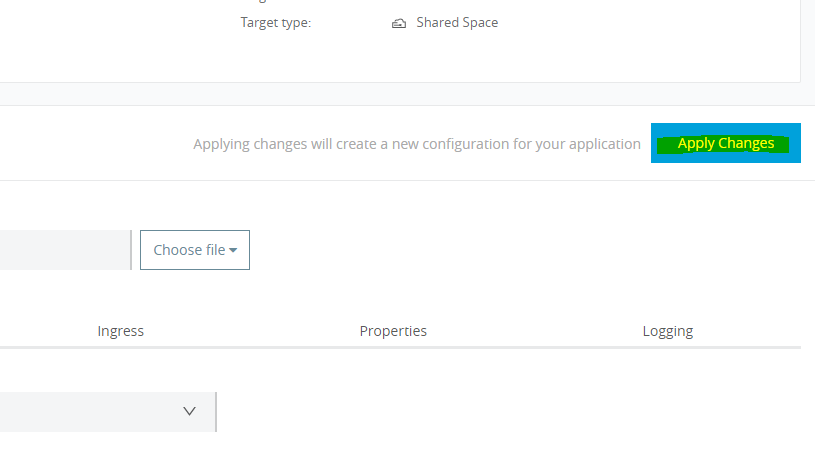
**Step1: Choose the file again**

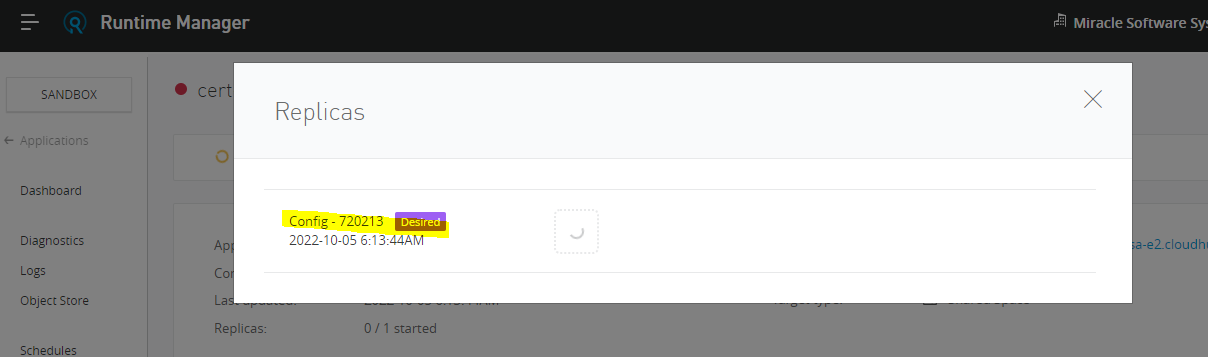
****

**Step2: Choose Recreate deployment model**

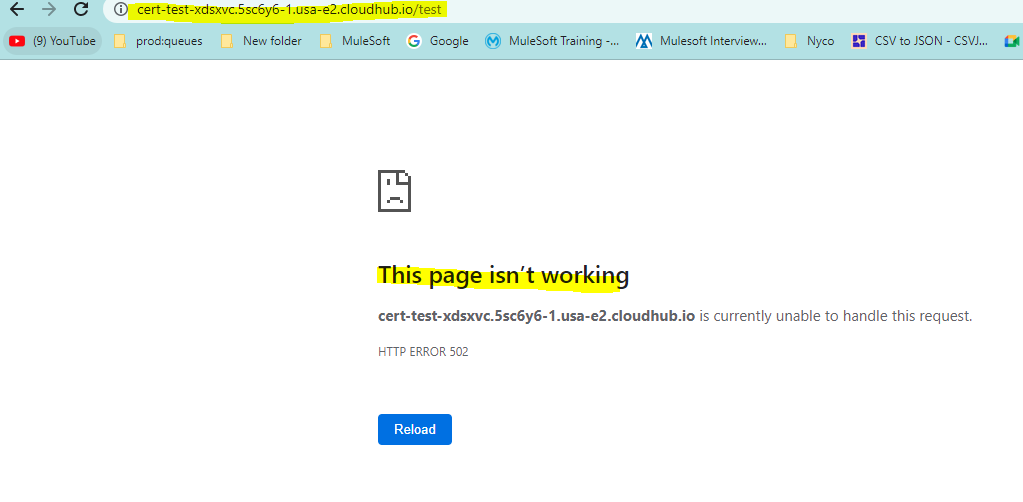
****

**Step3: Click on Apply changes**

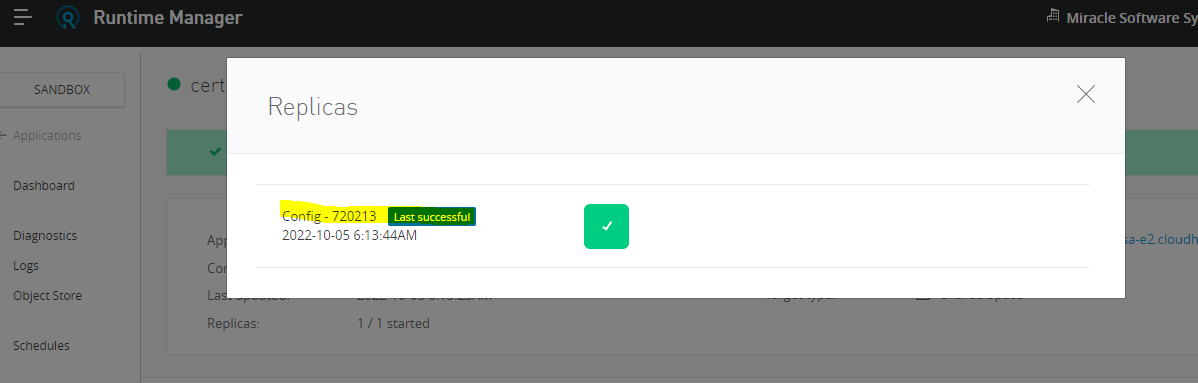
****

****

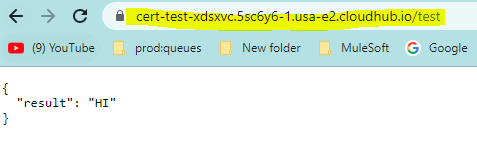
**Step4: So In case of recreate we cannot achieve zero downtime updates**

****

**Step5: After the successful completion of application deployment**

****

**Step6: We get the proper response**

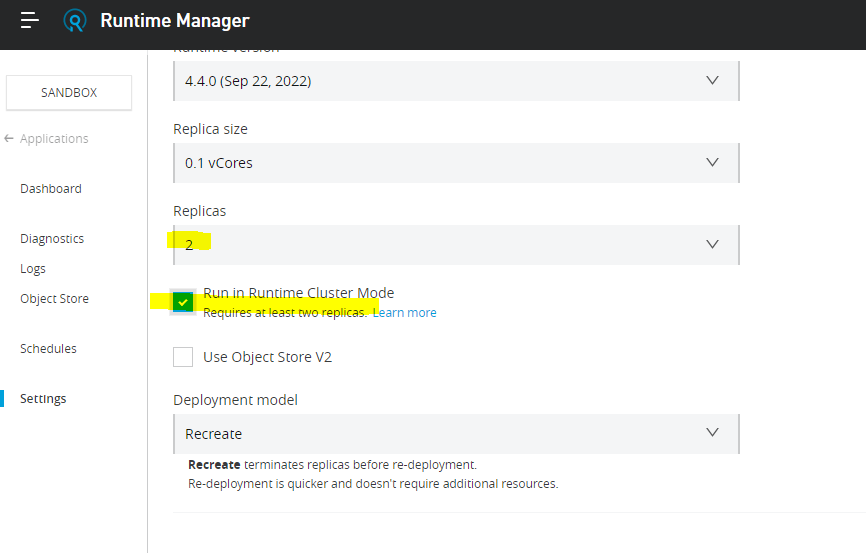
****

# Lab5: Horizontal and Vertical scaling in CloudHub 2.0

**Horizontal Scaling:**

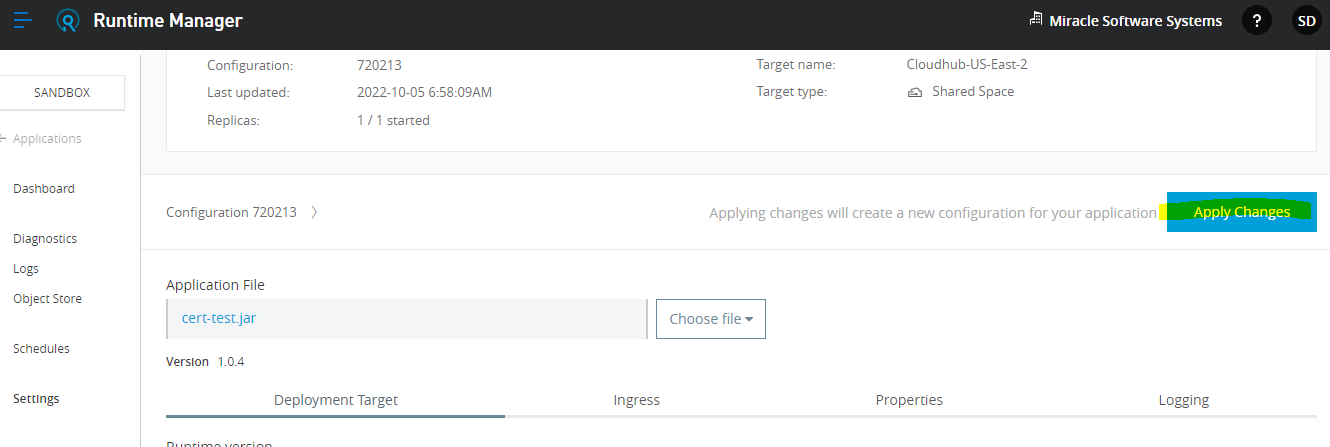
It means you have to increase the number of nodes or number of workers or number of replicas within your application so basically we are adding the more instance of the application so that is know as horizontal scaling.

**Step1:** Need to increase the number of replicas so I will do 2 and each replica will be of 0.1vCode and this application will consist of total 0.2vCode

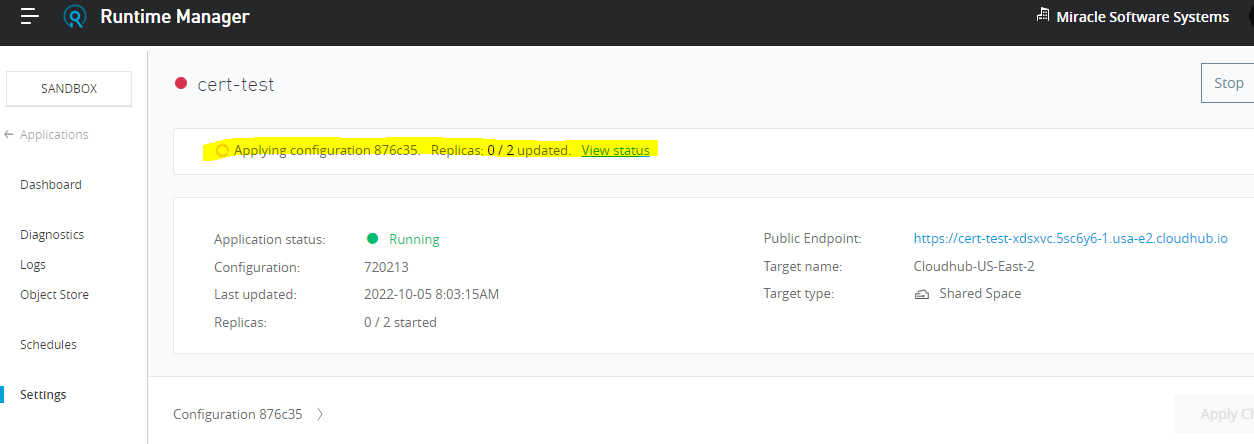


**Run in Runtime Cluster mode:** Anypoint Clustering it means if your application have a vm queue and is object stored and you want to share that object store and vm queue across the multiple replicas of the application so we can enable a cluster mode for this application

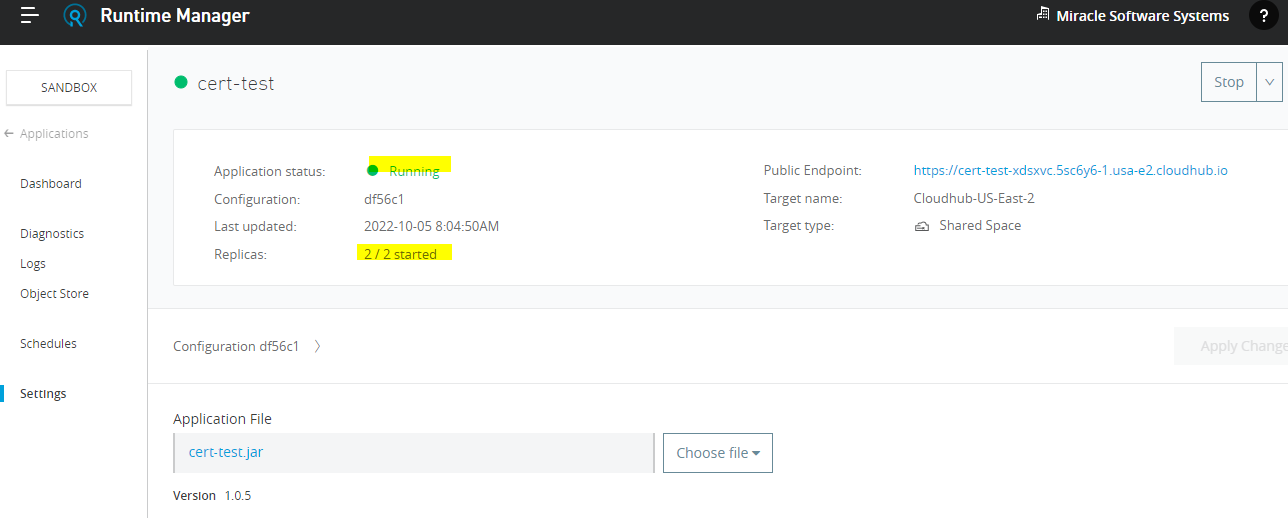
**Step2:** We need to click the apply changes



**Step3:** Here it shows 0/2 replicas



**Step4:** Now we can see both the replicas have been successfully deployed

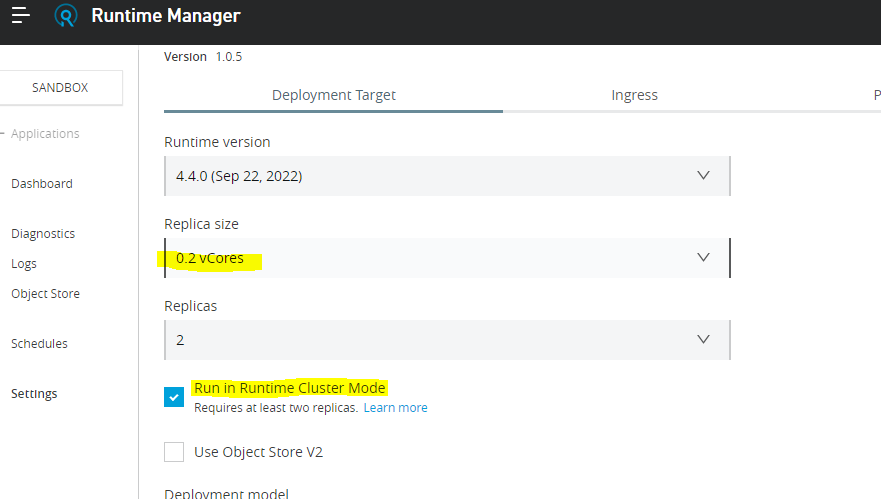


So, two out of two replicas have been started and now our application is running on the multiple replicas so that it can take nore traffic and that traffic will be distributed across the replicas you have an ingris load balancer so it will take care through distributed traffic across the multiple replicas of our application

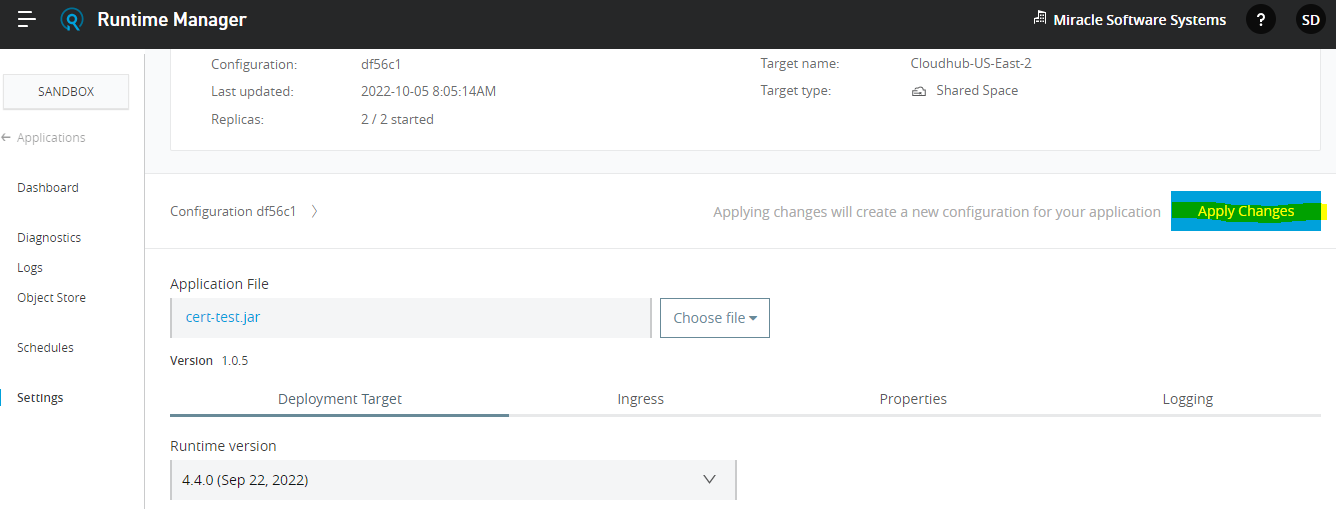
**Vertical Scaling:**

It means when we are increasing the computing resources let’s consider if you are increasing a ram or random access memory of the current computer or current server is known as vertical scaling.

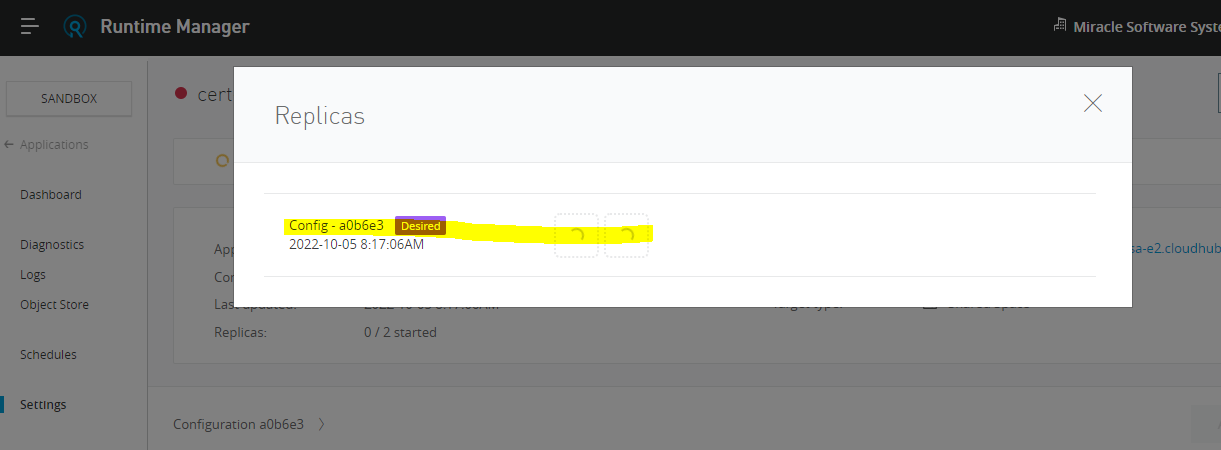
**Step1:** We need to change the replica size according to the requirement



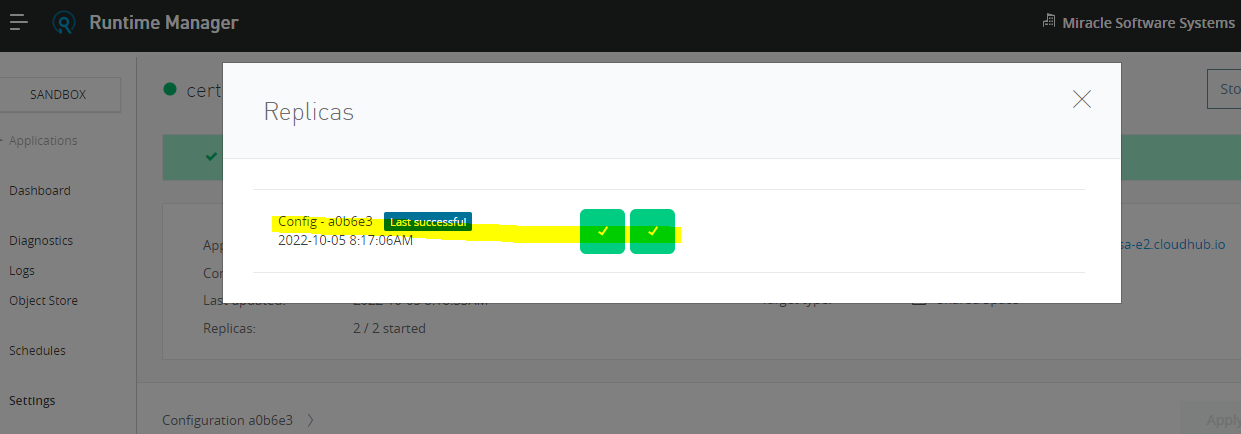
**Step2:** Click apply changes



**Step3:** Application deployment started



**Step4:** We can see that the two replicas started successfully



**Step5:** We can observe that the application is deployed successfully and running and in this way we can achieve the vertical scaling by increasing the replica size

