Argo-cd apps gitops

Wednesday, May 29, 2024

2:40 PM

Automatically deployment of kubernetes clusters.

Argo CD is am

Namespace in kubernetes :

In the gitops branch we have

Argocd-apps.yaml and argocd-gar.yaml needs to be managed by the ci/cd team.

Helm Chart Session

Thursday, May 30, 2024

12:21 PM

**Type : Application -**> we can also set this to library which will signify that this chart will provide the supporting libraries for other charts.

Helm Charts VS Manifest files :

* **Simple Deployments:** For very simple deployments with static configurations, manifest files might suffice.
* **Complex Deployments:** For deployments with multiple components, dynamic configurations, or dependency management, Helm charts offer significant advantages.

Ultimately Helm generates the manifest file to deploy to kubernetes cluster but it's generated after filling the values dynamically.

Flow of using the Helm Chart:

* Helm create-> it install the basic template of the helm chart with all the required files.
* Helm install -> once we have configured everything we need to run this command to deploy the application in kubernetes cluster.
* Helm upgrade -> if we make any changes after deployment we use this command to dynamically update the application. It will update the version of the helm chart.
* Helm template : renders the templates of a helm chart without deploying them. So that we can verify the generated manifest file. It requires some arguments.

How installation works in Helm : helm install releaseName

* When we run the helm install command, a process is started.
* Helm library -> install api is called.
* After some verification, the library renders the charts template
* Library loads the resulting resources into kubernetes.
* Library returns the release object like pod to the client. (in our case it is service.yaml file)
* The client exits.

There are 2 life cycle hooks here :

1. Pre-install - by the name we can say that it is processed before the installation starts.
2. Post-install - it is processed after the installation is done.

Hooks are given weights and based on those weight they are executed. Lower weights are executed first.

We define the hook under "annonations" in "metadata" attribute of yaml file (defined in our deployment.yaml file).

Annonations:

* Helm.sh/hook: pre-install

**FOR PM2**

apiVersion: v2 -> on each upgrade command it will increase -> version control  
name: m2madapter -> name of the chart  
description: A Helm chart for Kubernetes -> what chart does   
type: application -> signifies that chart deploys a fully fledged application   
version: 0.1.0 -> version of the chart itself, usful for tracking updates   
appVersion: "1.16.0" -> likely the application version being deployed

dependencies:  
- name: netreqmsgprc  
 repository: <file://charts/netreqmsgprc>  
 condition: netreqmsgprc.enabled -> will check the values of enabled variable in values.yml file and it should be set to true then only this dependency will be installed.

To install parent it will first install child charts only if enables is set to true

To determine the dependency of each service we need to look at the appsettings.Production.json file.

**FOR netreqmsgprc** :

apiVersion: v2  
name: netreqmsgprc  
description: M2M Network Request Message Protocol  
type: application  
version: 0.1.0  
appVersion: "cd98d8ac" -> this is the docker image tag

Templates folder -> how the chart is deployed to kubernetes, they are written in go template language. ( need to research on it)

Templates :

Configmap.yaml - it is just used for storing normal configuration data.

* apiVersion : configmap.yaml version
* Kind configmap type
* Metadata : here we define the name and labels
  + Name : dynamically generating the name of the configmap for deployment
  + Label : creating the label so that we can use it from our service or other files.
* Data : it contains all the key value pairs we have to store.

Deployment.yaml file : it generates the file and configuration for pods which is to be deployed.

Service.yaml : studied this yesterday.

* Services can be different type loadbalancer, NodePort, ClusterIp.

ServiceAccount.yaml : this file provides an identity for pods to access kubernetes resources without requiring hardcoded credentials.

Global keyword significance in values.yaml file : This keyword is often used to define global configuration vlaues that are shared across multiple componentsor service within the chart, This helps to keep the configuration DRY(Don't repeat Yourself) and makes it easier to manage & update the shared settings.

* When you define a value using global keyword in the values.yaml, they can be overridden locally if needed.
* If a value is defined globally and locally within a specific component or service the local value takes precedence.
* When local values override global values, there is conflict & Here is how it is resolved :
  + Helm first checks if a value is defined locally for a specific component or service. If it finds a local value, it uses that value, regardless of whether there's global value with the same identifier.
  + If the local value is not defined for a particular component or service, Helm falls back to using the global value, if available.

service:

[type](onenote:#Helm%20Chart%20Session&section-id={0394F28F-0206-4424-879B-42E77D94DD20}&page-id={36F832F5-7A85-440D-BB4F-D85985FCCA74}&object-id={3D005B5B-4495-4C06-9BBC-B562CCF9EB32}&61&base-path=C:\Users\keshriha\Desktop\Notes\Java%20Notes\CI-CD%20Notes.one): [Load Balancer](onenote:#Helm%20Chart%20Session&section-id={0394F28F-0206-4424-879B-42E77D94DD20}&page-id={36F832F5-7A85-440D-BB4F-D85985FCCA74}&object-id={3D005B5B-4495-4C06-9BBC-B562CCF9EB32}&61&base-path=C:\Users\keshriha\Desktop\Notes\Java%20Notes\CI-CD%20Notes.one)

[annotations](onenote:#Helm%20Chart%20Session&section-id={0394F28F-0206-4424-879B-42E77D94DD20}&page-id={36F832F5-7A85-440D-BB4F-D85985FCCA74}&object-id={3D005B5B-4495-4C06-9BBC-B562CCF9EB32}&AC&base-path=C:\Users\keshriha\Desktop\Notes\Java%20Notes\CI-CD%20Notes.one):

          networking.gke.io/load-balancer-type: "Internal"

      livenessProbe:

        httpGet:

          path: /health/live

          port: 8080

      readinessProbe:

        httpGet:

          path: /health/ready

          port: 8080

      startupProbe:

        httpGet:

          path: /health/startup

          port: 8080

        failureThreshold: 10

        periodSeconds: 10

Service Type: There are several types of services in kubernetes to expose applications running in the cluster to other applications or users outside the cluster.

ClusterIp : exposes the service on an internal IP address reachable only within the cluster. This is default type is this.

NodePort : This type exposes the serviceon a static port on each node's IP address. It makes the service accessible from outside the cluster using NodeID:NodePort.

LoadBalancer : this type creates an external load balancer in the cloud providers network, which forwards traffic to the service. It's typically used when running kubernetes in a cloud env that supports load balancers.

ExternalName : This type maps the service to DNS name. It's used to provide a custom DNS name for an external service.

Headless : used to create a service without an associated cluster IP. It's use for services that don't need load balancing or require direct access to individual pods.

Also in specific env with certain distributions, ou might find other types of services that are tailored to the platforms' features and capabilities.

For example, in Google Kubernetes Engine (GKE), there's a type called **Internal Load Balancer**, which exposes the service on a private IP address accessible only within the same VPC (Virtual Private Cloud).

* Image name should be picked from the Build name in the container list in the excel file.
* There are multiple helm charts for some images, so for those charts, values.yaml file wcnf services and kafka services values can found inside the prod folder of sbs repo it will be like appsettingsServiceName.json file.
* So we need to take the values from that json and not from the appsetting-production.json file.

Helm Chart Deployment

Friday, May 31, 2024

6:54 PM

On what cases appsettings\_configmap.yaml file condition will be true.

name: {{ include "fullname" . }}-container-appsettings

  labels:

This will call fullname method but what is this --container-appsettings for?

Why are we iterating configmap.yaml file, is this configmaps values coming from the appsettings\_configmap.yaml file and if not where are they coming from.

What is .name here -:

name: {{ include "fullname" $ }}-{{ .name }}

data:

  {{ .file }}: |

    {{- .content | nindent 4 }}

what is the meaning of this here :

 {{- if or .Values.podAnnotations (and .Values.configMaps (gt (len .Values.configMaps) 0)) (eq (include "includeContainerAppsettingsConfigMap" .) "true") }}

      annotations:

        {{- with .Values.podAnnotations }}

{{- tpl (toYaml .) $ | nindent 8 }}

What is meaning of above line. What is this -tpl (toyaml .) $

Wha tis the purpose of this :

checksum/config-{{ .name }}: {{ include "configmapChecksum" . | quote }}

ports:

    {{- range .Values.service.ports }}

    - port: {{ .port }}

      targetPort: {{ .targetPort }}

      protocol: {{ .protocol }}

      name: {{ .name }}

    {{- end }}

Why are we using .targetport or .port or .name

On what cases serviceAccount.yaml file if condition will be true.

What does this line does :

{{- with .Values.serviceAccount.annotations }}

What is virtual service and when will this be enabled.

gateways:

    {{- toYaml .Values.virtualService.gateways | nindent 2 }}

  hosts:

    {{- toYaml .Values.virtualService.hosts | nindent 2 }}

What is the meaning of above line.

What is the meaning of pullPolicy.

podAnnotations:

      vault.security.banzaicloud.io/run-as-non-root: "true"

      vault.security.banzaicloud.io/vault-role: "{{ .Values.parentChart }}"

    podLabels:

      vault-injection: enabled

podSecurityContext:

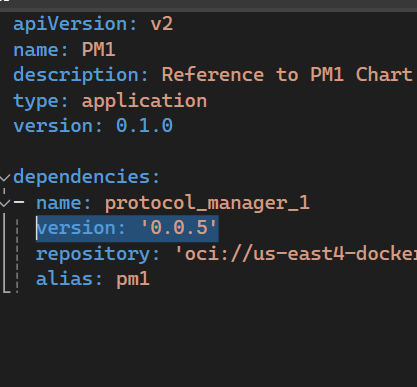
      seccompProfile:

        type: RuntimeDefault

      runAsNonRoot: true

Steps to create and deploy a helm chart :

1. Take the latest pull form the emerge helm charts branch.
2. Create/copy existing chart.
3. Assign the name of the chart same as your assigned task name. if your task name is Container lg-emerge-m2m-jasapi than your chart name should be "jasapi".
   1. Once your chart is created change the port number.
4. Once you have created the helm chart and done the required changes, push it to the git repo.
5. After pushing the chart, take the latest pull of the emerge-gitops branch. (we are doing this just to maintain the version of our chart such that argocd know that we have changed our chart and it needs to deploy the latest changes).
6. Change the version of the protocol manager



1. Push the change.
2. Once you have pushed the change, package the emerge-helm-charts.
3. Once you have

After you have pushed the changes

Create the package using helm push Protocol\_Manager docker\_repo

Kubernatives

1. Create a kubernatives Cluster:->

Using Minikube to create a cluster :-

Kubernates Cluster - Kubernates coordinates a highly available cluster of computers that are connected to work as a single unit.

The abstractions in kubernetes allow you to deploy containerized applications to a cluster without tying them specifically to individual machines.

To make use of this new model of deployment, applications needs to be packaged in such a way that decouples them from individual hosts : they need to be containerized

Containerized applications are more flexible and available than past deployment models, where applications were installed directly onto specific machines as packages deeply integrated into the host.

Kubernates automates the distribution and scheduling of application containers across a cluster in a more efficient way.

Kubernetes consists of two types of resource:-

1. Control plane- coordinates the cluster -> it is responsible for managing the cluster, coordinates all applications in your cluster such as scheduling applications, maintaining applications desired state, scaling applications and rolling out new updates.
2. Nodes- they are the workers that run applications.

A node is a VM or a physical computer that serves as a working machine in a kubernetes cluster, each node has a kubelet, which is an agent for managing the node and communicating with kubernetive control plane.

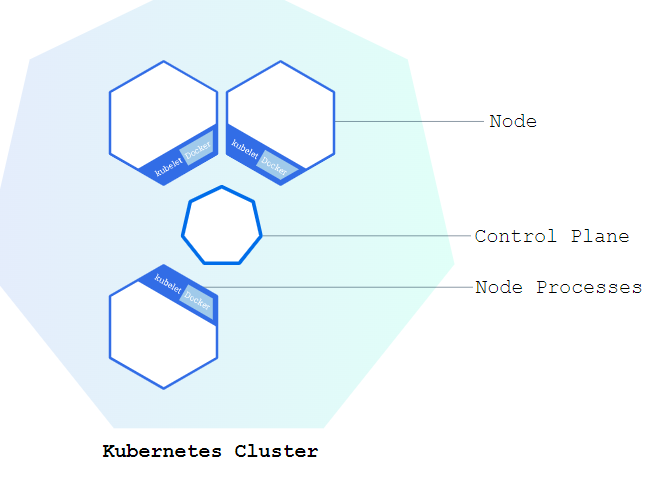
The node should also give tools for handling for container operations such as containerized or a docker.

A kubernetes cluster that handles production traffic should have minimum of three nodes because if one goes down, both an etcd member and a control plane instance are lost , and redundancy is compromised. You can mitigate the risk by adding more control plane nodes.

When we deploy application on kubernetes, you tell the control panel to start the application containers. The control plane schedules the containers to run on the cluster's nodes.

The nodes communicate the with the control plane using Kubernetes API, which the control plane exposes.

End users also use kubernets api to directly interact with the cluster.



Kubernetes cluster can be deployed on either physical machine or virtual machines.

To get started with kubernetes you can use MINIKUBE which is a lightweight kubernetes implementation that creates a local VM on your local machine and deploys a simple cluster containing only one node.

Saturday, December 31, 2022

11:36 PM

What Kubernatives do for us:->

Containerization helps package software to serve these goals, enabling applications to release and updated without no downtime.

Kubernatives helps you make sure those containerized application run where and when you want and helps then find the resource and tools they need to work.

Kubernatives is a open-source platform in container orchestration combined with best of breed ideas from the community. Developed by google.

Use Kubectl to create a deployment:->

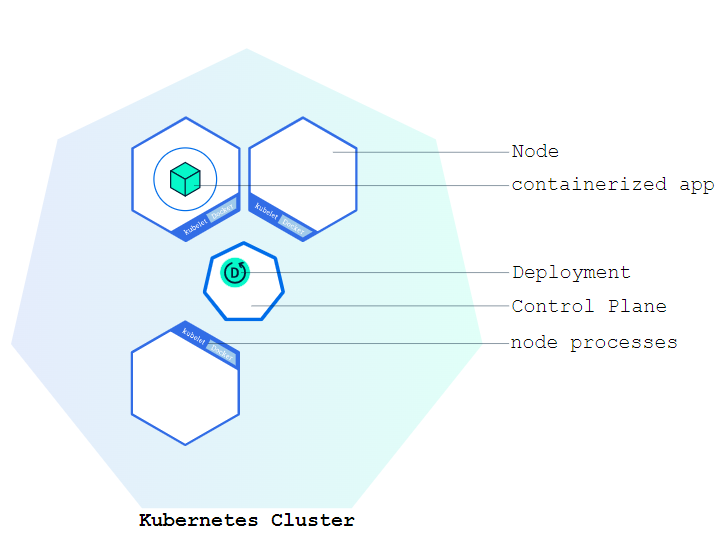
Kubernetes Deployement: once we have kubernetes cluster we can deploy our containerized applications on top of it. To do so we create a Deployment configuration. The deployment instructs how to create and update instances your application.

Kubernetes Deployment - once you have a kubernetes cluster, you can deploy your containerized applications on top of it, to do it we create a kubernetes configuration.

Deployment instructs Kubernetes how to create and update instances of your application. Once you have created a deployment , the kubernetes control plane schedules the application instances included in that deployment to run on individual nodes in the cluster.

Once the application instances are created, a kubernetes deployment container continuously monitors, if the node hosting an instance goes down or is deleted, the deployment controller replaces the instances with an instance of another node in the cluster. ***This provides the self-healing mechanism to address machine failure or maintenance.***

In pre-orchestration world, installation scripts would often be used to start applications but they did not allow recovery from machine failure By both creating your applications instances and keeping them running across Nodes, kubernetes deployments provide a fundamentally different approach to application management.



You can create and manage a deployment by using the kubernetes command line interface, kubectl. Kubectl uses the kubernets API to interact with the cluster.

When you create a deployment you will need to specify the container image for your application and number of replicas that you want to run. We can change that information by updating your deployment.

Kubernetes Pods:-

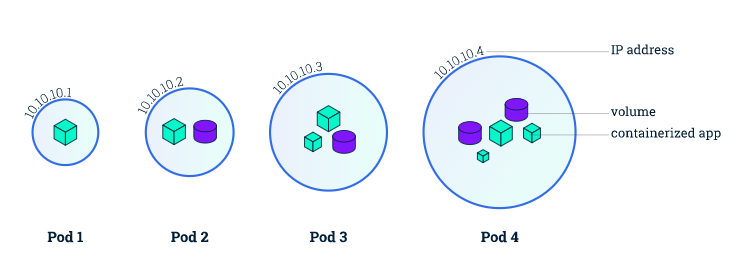
When you create a deployment , kubernaetes created a pod to host your application instance. A pod is kubernetes abstraction that represents a group of one or more application containers (such as docker) and shared resource for those containers. Those resources include:-

1. Shared storage as volumes
2. Networking, as a unique cluster IP address
3. Information about how to run each container, such as container image version or specific ports to use

A pod models an application specific "logical-host" and can contain different application containers which are relatively tightly coupled.

The containers in a pod shares an IP address and a port space , are always co-located and co-scheduled, and run in a shared context on the same node.

Pods are the smallest unit on the kubernetes program, when we create a deployment on kubernetes, that deployment creates pods with containers inside them (as opposed to containers directly). Each pod is tied to the where it is scheduled and remain there until termination or deletion. In case of node failure, identical pods are scheduled on other available nodes in cluster.



Node : A pod always runs on a node. A node is virtual machine on a kubernetes or may either a virtual machine or physical machine depending on the cluster

Each node is managed by the control plane, a node can have multiple pods and the kubernetes control plane automatically handles scheduling the pods across the nodes in the cluster.

The control plane's automatic scheduling takes into account the available resource on each node.

Every kubernetes nodes run at least :-

1. Kubelet, a process responsible for communication between the kubernetes control plane and node, it manages the pods and containers running on the machine.
2. A container runtime (like docker) responsible for pulling the container image from a registry, unpacking the container and running the application.

kubectl get-> responsible for listing all the resources.

Kubectl describe-> shows detailed information about the resource.

Kubectl logs-> prints the logs from a container in a pod.

Kubectl exec -> execute a command on a container in a pod.

**Etcd in kubernetes :-**

If you have ever interacted with the kubernetes cluster in any way, chances are it was powered by etcd under the hood. But even though etcd is at the heart of how kubernetes works, it's rare to interact with it directly on a day-to-day basis.

**How etcd fits into Kubernetes :-**

At a high level, kubernetes cluster has three categories of control -plane processes :

1. **Centralized controllers** like the scheduler, controller-manager, and third party controllers, which configure pods and other resources.
2. **Node-specific processes**, the most important of which is kubelet, which handle the nitty-gritty of setting up pods and networking and networking based on the desired configuration.
3. The **API server**, which coordinates between all of the control plane processes and the nodes.

One of the interesting design choices in kubernetes is that the API server itself does very little :

When a user or process performs an API call, the API server:

1. Determines whether the API call is authorized (using RBAC). (RBAC - it is role based access control , which a method of regulating access to computer or network resources based on the roles of individual users within your organization.
2. Possibly changes the payload of the API call through mutating webhooks.

Docker

origin/wip\_1970708\_container\_lg-emg-sbs-fwdldmgt

origin/wip\_1970717\_container\_inbmsgprc

origin/wip\_1970734\_container\_lg-emg-sbs-inbmsgrcv

origin/wip\_1970743\_container\_inbnetcomprc

origin/wip\_1970752\_container\_lg-emg-sbs-inbsrvreqrcv

origin/wip\_1970752\_dev4.0\_inbsrvreqrcv

origin/wip\_1970761\_container\_oubmsgdlv

origin/wip\_1970770\_container\_lg-emg-sbs-oubmsgprc

origin/wip\_1970780\_container\_oubmsgrcv

origin/wip\_1970789\_container\_sysset

Tuesday, June 4, 2024

9:14 PM

Login into the gcloud console ->

Gcloud auth login --no-launch-browser

Gcloud config set project cpet-emerge-dev-05-ahf-01

Gcloud auth configure-docker us-east4-docker.pkg.dev

Set the env variables :

export BASE\_IMAGE=us-east4-docker.pkg.dev/cpet-emerge-dev-05-ahf-01/cpet-emerge-dev-05-rep-a/lg-emg-base-aspnet:8.0.1

export REGISTRY=us-east4-docker.pkg.dev/cpet-emerge-dev-05-ahf-01/cpet-emerge-dev-05-rep-a

export BUILDER\_IMAGE=us-east4-docker.pkg.dev/cpet-emerge-dev-05-ahf-01/cpet-emerge-dev-05-rep-a/lg-emg-build-dotnet:8.0.1

origin/wip\_1970708\_container\_lg-emg-sbs-fwdldmgt

origin/wip\_1970717\_container\_inbmsgprc

origin/wip\_1970734\_container\_lg-emg-sbs-inbmsgrcv

origin/wip\_1970743\_container\_inbnetcomprc

origin/wip\_1970752\_container\_lg-emg-sbs-inbsrvreqrcv

origin/wip\_1970752\_dev4.0\_inbsrvreqrcv

origin/wip\_1970761\_container\_oubmsgdlv

origin/wip\_1970780\_container\_oubmsgrcv

origin/wip\_1970789\_container\_sysset

origin/wip\_1970770\_container\_lg-emg-sbs-oubmsgprc

-> docker command to build an image

docker build -f Dockerfile/Dockerfile\_lg-emg-sbs-fwdldmgt . -t $REGISTRY/lg-emg-sbs-fwdldmgt:$(git rev-parse --short HEAD) --build-arg BUILDER\_IMAGE=$BUILDER\_IMAGE --build-arg BASE\_IMAGE=$BASE\_IMAGE

-> docker command to run the image which is build : docker run imageName:ImageTag

docker build -f Dockerfiles/Dockerfile\_lg-emg-sbs-oubmsgdlv. -t $REGISTRY/lg-emg-sbs-oubmsgdlv:$(git rev-parse --short HEAD) --build-arg BUILDER\_IMAGE=$BUILDER\_IMAGE --build-arg BASE\_IMAGE=$BASE\_IMAGE

docker build -f Dockerfiles/Dockerfile\_lg-emg-sbs-sysset . -t $REGISTRY/lg-emg-sbs-sysset :$(git rev-parse --short HEAD) --build-arg BUILDER\_IMAGE=$BUILDER\_IMAGE --build-arg BASE\_IMAGE=$BASE\_IMAGE

docker build -f Dockerfiles/Dockerfile\_lg-emg-sbs-sysset . -t $REGISTRY/lg-emg-sbs-sysset:$(git rev-parse --short HEAD) --build-arg BUILDER\_IMAGE=$BUILDER\_IMAGE --build-arg BASE\_IMAGE=$BASE\_IMAGE

docker build -f Dockerfile/Dockerfile\_lg-emg-sbs-fwdldmgt . -t $REGISTRY/lg-emg-sbs-fwdldmgt:$(git rev-parse --short HEAD) --build-arg BUILDER\_IMAGE=$BUILDER\_IMAGE --build-arg BASE\_IMAGE=$BASE\_IMAGE

docker build -f Dockerfile/Dockerfile\_lg-emg-sbs-inbmsgprc . -t $REGISTRY/lg-emg-sbs-inbmsgprc :$(git rev-parse --short HEAD) --build-arg BUILDER\_IMAGE=$BUILDER\_IMAGE --build-arg BASE\_IMAGE=$BASE\_IMAGE

gcloud config set project cpet-emerge4-dev-43-s-ajw-01

gcloud beta compute ssh cpet-proxy --tunnel-through-iap --zone=us-east4-a -- -4 -L8888:127.0.0.1:443 -N -q -f - '

alias cpet-proxy='gcloud beta compute ssh cpet-proxy --tunnel-through-iap --zone=us-east4-a -- -4 -L8888:127.0.0.1:443 -N -q -f - '

gcloud container clusters get-credentials gke-dev43 --region=us-east4

alias ak='HTTPS\_PROXY=127.0.0.1:8888 kubectl'

ak get nodes

ak get secrets -n argo-cd argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d

ak exec -it istio-ingressgateway-7577bdb59f-kkhcr -n istio-ingressgateway -- curl <http://10.104.70.133> # Test AUX communication

ak exec -it istio-ingressgateway-7577bdb59f-kkhcr -n istio-ingressgateway -- nc -v 10.104.70.136 5444 # Test Postgres communication

To generate the template of the helm chart : helm template argocd-apps . -f values-applications.yaml -f values-applicationsset.yaml -f values-projects.yaml

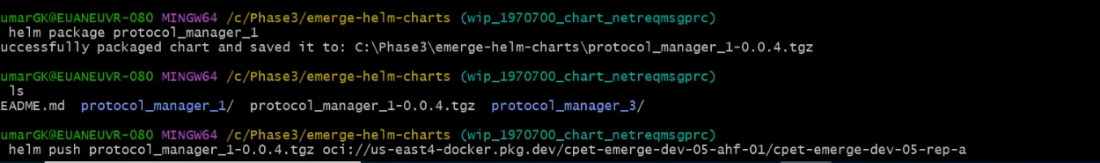
Steps to make a helm chart

Tuesday, June 4, 2024

11:35 PM

Once you have pushed your chart run the following commands :

1. Helm package chartName
2. Helm push chartName.tgz docker-registry-url



Problems faced in building docker

Thursday, June 6, 2024

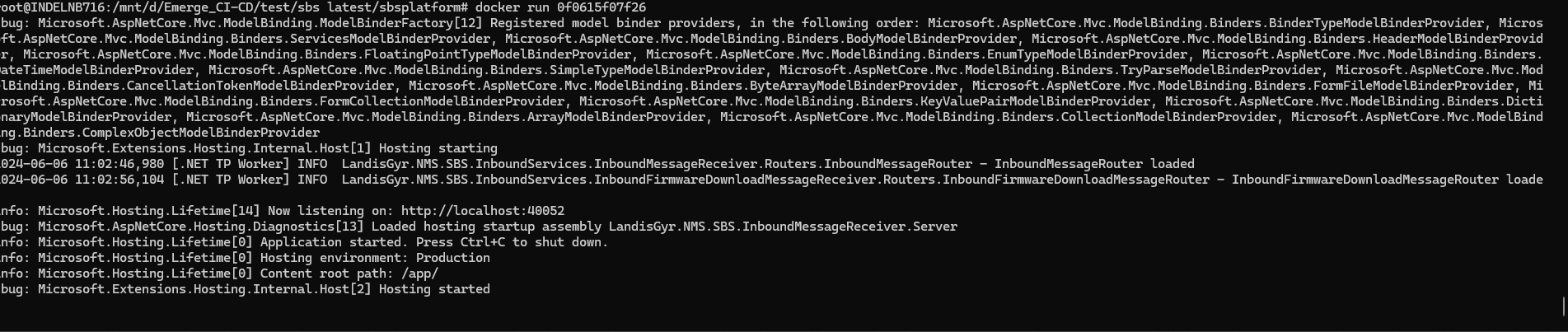
4:29 PM

All the images are build but facing problem while running them :

**Lg-sbs-emg-fwdldmgt -> facing issues while connecting to database.**



**lg-emg-sbs-inbmsgrcv : working fine (no errors)**



**lg-emg-sbs-inbsrvreqrcv : working fine (no errors)**



**lg-emg-sbs-oubmsgprc : image is started but getting error while connecting to kafka server**



Configure env to access from wsl

Friday, June 7, 2024

3:04 PM

gcloud beta compute ssh cpet-proxy --tunnel-through-iap --zone=us-east4-a -- -4 -L8888:127.0.0.1:443 -N -q -f -

Then I defined an alias:

alias ak='HTTPS\_PROXY=127.0.0.1:8888 kubectl'

Access to gke cluster from wsl

Monday, June 10, 2024

6:28 PM

To successfully access the kubernetes cluster we need to run these commands :

gcloud auth login --no-launch-browser

gcloud config set project cpet-emerge-dev-05-ahf-01

gcloud auth configure-docker us-east4-docker.pkg.dev

gcloud beta compute ssh cpet-proxy --tunnel-through-iap --zone=us-east4-a --project=cpet-emerge4-dev-43-s-ajw-01 -- -4 -L8888:127.0.0.1:443 -N -q -f -

alias ak='HTTPS\_PROXY=127.0.0.1:8888 kubectl'

Ak get pods

Ak get nodes

Check if any service is running on the kubenetes cluster :

Check : ps aux | grep "8888"

Kill : kill -9 serviceId

Either kill the service or change the port number. (preffered to kill the port number).

To view all the namespace :

Ak get ns

To view all the pods inside the namespace :

Ak get pods --namespace=name

Provide the project id with the gcloud beta command -> gcloud beta compute ssh cpet-proxy --tunnel-through-iap --zone=us-east4-a --project=cpet-emerge4-dev-43-s-ajw-01 -- -4 -L8888:127.0.0.1:443 -N -q -f -

Set the alias : alias ak='HTTPS\_PROXY=127.0.0.1:8888 kubectl'

gcloud container clusters get-credentials gke-dev43 --region us-east4 --project cpet-emerge4-dev-43-s-ajw-01

run this commnd cpet-proxy server me ssh leke

then kubectl get pods

Run these commands in wsl and cluster ke andar jane ke liye run this command -> [click me](onenote:#Access%20to%20gke&section-id={0394F28F-0206-4424-879B-42E77D94DD20}&page-id={F0AE18FE-3830-44AA-A344-569D85DD4C56}&object-id={ABD64EC6-84AC-4133-BB37-E0895D8A1991}&20&base-path=C:\Users\keshriha\Desktop\Notes\Java%20Notes\CI-CD%20Notes.one)

gcloud beta compute ssh cpet-proxy --tunnel-through-iap --zone=us-east4-a project -- -4 -L8888:127.0.0.1:443 -N -q -f -

Then I defined an alias:

alias ak='HTTPS\_PROXY=127.0.0.1:8888 kubectl'

Here we are running the service in 8888 port so if any service is running on this port we need to kill it

Check : ps aux | grep "8888"

Kill : kill -9 serviceId

Tuesday, June 11, 2024

8:06 PM

Dockerfile ->commit -> docker images-> container id will generate-> push into artifact registry-> once image will come ->cc manifest file will update ->helm chart will know file is update-> helm chart pipeline will be updated-> helm chart update->

Cc manifest id will change -> this file will be generated for the generated.

Helm chart will be updated as per this manifest file.

Argo cd will update monitor helm and

Argo cd will have alag alag deployment for kafka pm1 and namespace.

Har env ka apna namespace hoga.

Usme pods honge.

16 container ke 16 pods banenge.

Aux ki kiss service se communicate kar raha hai uske liye

Device is communicating with the

Load balancer is for a service -> 1000's of devices are communicating through the load balancer into a single service so to process each request accordingly we have to add a load balancer such that whole process is completed smoothly. It's like a queue where 1000.. Of req comes and each one is processed one by one.

Hashicorp vault

Thursday, June 13, 2024

5:19 PM

How to access the Hashicorp Vault:

Set the gcp project:

gcloud config set project projectid

Connect to the cpet-proxy over IAP and do port-forwarding:

gcloud beta compute ssh cpet-proxy --tunnel-through-iap --zone=us-east4-a -- -4 -L8888:127.0.0.1:443 -N -q -f -

Fetch cluster credentials:

gcloud container clusters get-credentials clustername --region us-east4

Configure an alias for kubect using proxy

alias ak='HTTPS\_PROXY=127.0.0.1:8888 kubectl'

Fetch vault password:

ak get secret bank-vaults -n vault -o jsonpath="{.data.vault-root}" | base64 -d

Port forward the Vault service to localhost

ak port-forward svc/vault -n  vault 8200:8200

Questions

Monday, June 17, 2024

11:29 AM

class Solution {

public List<List<Integer>> threeSum(int[] nums) {

List<List<Integer>> ans = new ArrayList<>();

Arrays.sort(nums); // Sort the array

for (int start = 0; start < nums.length - 2; start++) {

if (start > 0 && nums[start] == nums[start - 1]) continue; // Skip duplicates

int left = start + 1;

int right = nums.length - 1;

int target = -nums[start];

while (left < right) {

int sum = nums[left] + nums[right];

if (sum == target) {

ans.add(Arrays.asList(nums[start], nums[left], nums[right]));

left++;

right--;

// Skip duplicates

while (left < right && nums[left] == nums[left - 1]) left++;

while (left < right && nums[right] == nums[right + 1]) right--;

} else if (sum < target) {

left++;

} else {

right--;

}

}

}

return ans;

}

}

class Solution {

    public List<List<Integer>> threeSum(int[] nums) {

        List<List<Integer>> ans=new ArrayList<>();

        Arrays.sort(nums);

        for(int start=0;start<nums.length-1;start++){

            if(start>0 && nums[start]==nums[start+1])continue;

            int left=start+1;

            int right=nums.length-1;

            int target=-nums[start];

            while(left<right){

                int sum=nums[left]+nums[right];

                if(sum==target){

                    ans.add(Arrays.asList(nums[start],nums[left],nums[right]));

                    left++;

                    right--;

                    while(left<right && nums[left]==nums[left-1]) left++;

                    while(right>left && nums[right]==nums[right+1]) right--;

                }else if(sum+target>0){

                    right--;

                }else{

                    left++;

                }

            }

        }

        return ans;

    }

}

Not able to generate secret key.

Are there some pods which have multiple containers running inside them.

Which pods are we exposing and which are communicating through kafka.

Can you please explain how this deployment file works.

apiVersion: argoproj.io/v1alpha1

kind: Application

metadata:

  name: argocd-apps

  namespace: argo-cd

  annotations:

    argocd.argoproj.io/manifest-generate-paths: .

spec:

  project: default

  source:

    repoURL: <https://gitlab.cicd.landisgyr.com/landisgyr/rnd/hesmod/commandctr/emerge-gitops.git>

    targetRevision: emerge-dev-05

    path: argocd-apps

    helm:

      valueFiles:

        - values-applications.yaml

        - values-applicationssets.yaml

        - values-projects.yaml

  destination:

    server: <https://kubernetes.default.svc>

    namespace: argo-cd

  syncPolicy:

    automated: {}

  revisionHistoryLimit: 1

STUDY THIS FILE, THIS IS CREATING THE WHOLE PROJECT.

WHAT WILL BE THE SERVER IN VALUES-PROJECTS.YAML FILE

Validation for GKE env

Monday, June 17, 2024

12:33 PM

What we need to check in this single node pool, how is it installed.

In the excel you have the check list of vm, rather than doing it on that excel, just do a validation of all the points.

All the nominal steps we can get it to document.

Kubernetes and argo cd session

Thursday, June 20, 2024

12:31 PM

How is gke cluster built -> lets look at dev-43, it is basically blue print to cluster, we have all those input files to create vms.

There is a file input\_compute.tf where all the vms are defined, vms defined here gets created. There is a input file, there is a location under subnetworks here ,computes points to compute resources which is defined in .

Work loads are pods where your app will run.

To run a it needs a docker image, as a security we need to define some policy kaha se image karna allowed hai ya nahi hai. Every thing in kubernetes is an object , here policy is a type of resource and to define it we use anthos config management, when the cluster is setup we point this cluster to a repository. In acm config-> acm it is bunch of yaml files which are applied on the cluster, these are resources which gets created in the cluster.

Anthos is not gcp, kubectl commands are used to interact with kubernetes, now if I have create something on the cluster, so to do that I will have to put it on yaml file and run kubectl -f filename, to create any resource in kubectl you need to put definitation in a file, if I wanna create a argocd in that, like if I want to run argo cd here, in order to create argo I need yaml file of every one of that. If I going to create argo-cd in every cluster there should be a way to pull the yaml and anthos provides a way to pull it.

We have provided a reference of the anthos in the kubernetes cluster yaml file.

All the definition regarding policies are left on anthos.

Config sync functionality is used to sync the repo with the cluster.

Ipmasquerade -> kubernetes pods use this network subnet, some subnets are used by control panel/nodes.

It will specify the source ip if pods is talking to other pod internally and something else if it is talking to some other pods.

In acm restrict-repos.yaml is a whitelister for the repo.

What is masqueradeCIDRs.

What is net-> network address translation. Whenever you communicate with something

Python Converter Script

Friday, June 21, 2024

2:45 PM

python3 helm\_converter.py path\_to\_appsettings\_production\_json path\_to\_output\_file

python3 converter.py LandisGyr.NMS.SBS.OutboundMessageDelivery.Host/appsettings.Production.json values.yaml

Thursday, June 27, 2024

11:07 PM

