국민권익위원회

범정부 AI기반 통합콜센터 서비스 구축 2차

운영자지침서

1.상담APP,KMS

문서번호	2024GW-APP-5021
버 전	1.0
단 계	이행
작 성 자	강신규
최종작성일	2024.10.30.

아리시스 컨소시엄

제 개 정 이 력

날짜	버전	작성자	승인자	내용
2024.04.17	0.1	김진연		서식 표준 작성
2024.09.02	0.2	강신규		최초 작성
2024.09.05	0.6	강신규		초안 작성 완료
2024.09.09	0.8	김진연		품질검토
2024.09.10	0.9		이룡	PM검토
2024.10.30	1.0		김경민	BaseLine(고객승인)
2024.12.07	1.1	김두포		8개기관 추가.

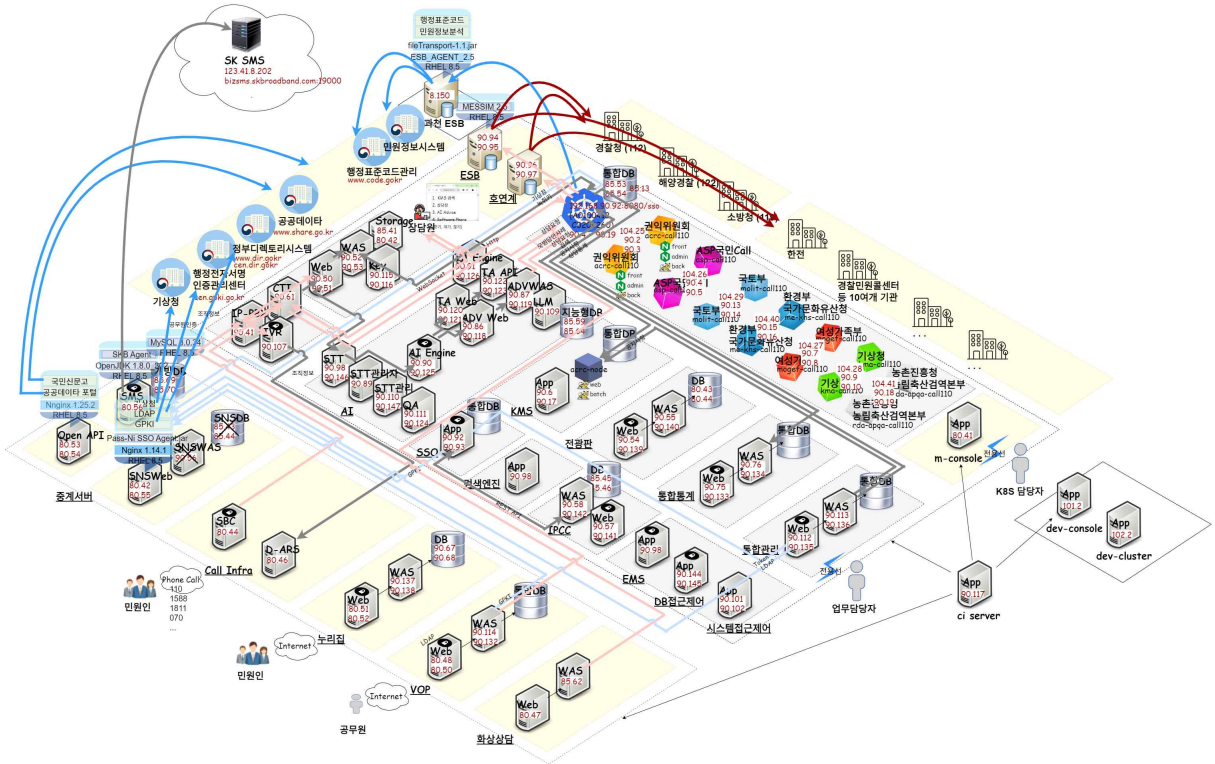
<제목 차례>

1. 개요

범정부 AI기반 통합 콜센터 서비스 구축 사업의 상담APP 및 KMS 서비스의 구성 및 환경에 대한 정보를 기술함.

2. 시스템 구성도

상담APP 와 KMS는 DEVOPS 서비스를 이용하여 Kubernetes 솔루션 환경에서 구현 됨.



3. K8S

3.1. Infrastructure Layer

- 인프라 영역으로 주로 Cluster생성 및 Namespace, MasterNode와 WorkerNode의 분리 및 VM 매핑 등 처리.

3.1.1. Cluster

/root/.kube/config

root@mconsole: ~/.kube

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0t
qQVZNUk13RVFZRApWUVFERXdwcmRXSmxjbTVsZEdwek1JSUJJakFOQmdrcWhraUc5dzE
laNTBhMXlhVTJLdExqQ3lZOGFoZ0dTOVdUc3dlaENGbFl3YQpLbFF4UDlEOFFGclVqdE
UovSWhQdlpnUnlRY3FyZ0xIT1JKS3I1aFlCWnMvTUFBSDBGY1FkNFpCQmM5NGxQcGVCd
eExmN1YwNnBRTG93RlFZRFZSMFIKQkE0d0RJSUthM1ZpWlhKdVpYUmxjekFOQmdrcWhr
pMEw0VS8vaVBIUXYxak12U1h5dEhpVUw1aVNudVZBNmRIVmxjdHZaSW5GTFEKeC9vVEs
JRUGhYCkNMWG1VSXo2Rkk3QW9IK0VLMmJjQkVWU2hDR1YyMWYvZUhya0NrMFBoc2xNar
    server: https://acrc-k8s-cluster-qrp2h.ske.gkr-west.scp-in.com:6
  name: acrc-k8s-cluster-qrp2h
contexts:
- context:
    cluster: acrc-k8s-cluster-qrp2h
    namespace: gitlab
    user: user
  name: user@acrc-k8s-cluster-qrp2h
current-context: user@acrc-k8s-cluster-qrp2h
kind: Config
preferences: {}
users:
- name: user
  user:
    client-certificate-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1
dNVGN6TmxvdWpLREVYTUJVR0ExVUVDaE1PYzNsemRHVnRPbTFoYzNSbGNuTXhEVEFMQm
lNlT0NJeitzYXNnY094RnNwN1pQajFRcGtENWxmT25lRWpBNUJCc3h5eUJHNVNKZ3JUe
QzlsWXJXQVp3WUFFd2lROVRyTmlJdGdIRVhNYngyeUx3azZwQVB6TkZOdms2Y2k5VkFh
FRkJRY0RBakFNQmdOVkhSTUIKQWY4RUFqQUFNQjhHQTFVZEl3UVlNQmFBRkFtMOVtTUF
1pRWdRcEpQOUlBME5aNXBXTW5Sc3dYS0ZoVlZTZDlsdVZNeTdMNEdxVEkKTmNnMGFXc2
TFCCllXbUJxNGR5OHZCTHhTcFY0Uy92NlNPYUVOVXNXejVpSjNiUTlXMlZ6d1RpQklFM
    client-key-data: LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUl
lZSzA4eEwKUlvsdWxuTFFhSFZQWTA5TXM5Nyt1Z2OwcitweFpMZOZlVHB3dG94Ry9isV
kl2VlFHcHVESThyVWloK2xsYmIyRXhLZVN4aHpsclIKMlNIdGdsbnZtMkY4N3RlaGZZV
SVZnM2xiekZMWFpNdG5LUOVxcDlJaE8zWTFVbEk3WnJMaTNheFBubGVNeHpFM3lycmIv
nWkMzZU1kdHMvSURVUkZHCnBJWnlFQm45eGZOWXJKYXE5a3NxaktDTHIOUGkrOGdKNTF
8zZEVsTERBTkZUZ1BmbXVSTHZaNG9yUTZ1ejhIaGgzemRvczVLZXhXCk4wZXkydzR6dl
wpOYit1VnJJSU1McOZqejUzZVcwdWFSZnNaWjZ4b2FyL2VoT3d4ckdJNmx2a2tBNmhRM
VTdzYk5KZnlIc2JDWXRxVHpJcWN5RTNUMApOcOhoU3hrRDVIVFZIT0tpOXlTajFUVDNE
ob3Y3aUlNNTZBQ2RoY1FUdG53MWZIYldyc3BuSDVIQU9rVEtWMDFIWGlpai8yM3ZmQgo
93Y3pkU1RZdDYKY3N4Z2OzL3M3eExxBOGJXaGxSSk1VNkVXUDRBcThCbWFzMktxZ2ZkOG
```

현재 K8S Client는 root OS계정으로 Master Node인 https://acrc-k8s-cluster-qrp2h.ske.gkr-west.com.scp-in.com 에 user라는 계정으로 접속해 있다.
현재 내 Context Setting은 cluster를 acrc-k8s-cluster-qrp2h를 사용하고 있고, Default로 gitlab 네임스페이스를 사용한다.

3.1.2. NODE

- 모든 물리적 VM들은 가상의 NODE들을 구성한다.

Node Name	VM	IP	credential
acrc-node (kms)	VM #1	192.168.90.6	root / !1Ke571d0878b05
	VM #2	192.168.0.17	root / !1Ke220d5bb50cd
acrc-call110	VM #1	192.168.90.2	root / !1Ke8dc4b5f31f6
	VM #2	192.168.90.3	root / !1Ke323cc1928af
asp-call110	VM #1	192.168.90.4	root / !1Ke95a3085bf5b
	VM #2	192.168.90.5	root / !1Keaa788fb84b4
mogef-call110	VM #1	192.168.90.7	root / !1Ked9282dbeaed
	VM #2	192.168.90.8	root / !1Ke017d7c971db
kma-call110	VM #1	192.168.90.9	root / !1Keea9fd92fbc6
	VM #2	192.168.90.10	root / !1Ke21a08e5de98
molit-call110	VM #1	192.168.90.13	root / !1Ke1f41387afdc
	VM #2	192.168.90.14	root / !1Ke4a86a023137
me-khs-call110	VM #1	192.168.90.15	root / !1Keee139837f59
	VM #2	192.168.90.16	root / !1Ke938def96529
rda-apqa-call110	VM #1	192.168.90.18	root / !1Kebfa2e81a562
	VM #2	192.168.90.19	root / !1Kee2fdb9b86b8

-

1) Master Node

✔ API Server

- K8S의 모든 Resosurce들을 통제하는 Commands 명령어 센터.

✔ Calico Service

- Network Service.
- K8S에 가상 NIC 제공 및 Network Stack 서비스.

✔ CoreDNS Service

- IP들을 가진 Domain-based Resource들간 통신을 위한 DNS 서비스

✔ Scheduler

- 해당 POD를 어느 Node에 배치 할것인지. 즉, Affinity, Taint등을 관리한다.

※ 몇 개 POD가 활성화 될지는 해당 Node의 Deployment > replicas를 통해 지정됨.

✔ Etcd

- K8S 모든 Resource들의 상태를 저장하는 Key/Value 저장소.

2) Worker Node

✔ Kube Proxy

- POD OS레벨의 iptable 관리. 즉, Cluster-IP와 Real-IP간 NAT 처리.

✔ Kubelet

- 주요 역할은 모든 Worker Node에서 실행되는 POD들의 상태 전체를 Master Node에 동기화. 모니터링.
- POD의 상태 변경에 따라 Master Node의 API Server의 명령에 따라 대응 처리.

✔ Containerd

- Image를 실행하는 Runtime으로 Docekr, Containered등이 있는데 이중, 경량화 최적화된 Containerd를 사용한다.

※ KMS, 상담APP POD들은 이 Containered상에서 동작한다.

3.1.3. Image Registry

KMS 상담APP Application들 Docker Image로 Build & Push.
협업 사용자들이 해당 Image를 Pull & Run해 사용할 중앙 Registry.

acrcregistry-eedmykle.scr.gkr-west.scp-in.com (acrc@systeer.com / Master#23)
acrcregistrydev-tdbvvmwx.scr.gkr-west.scp-in.com (acrc@systeer.com / Master#23)

1) Login
$ docker login acrcregistry-eedmykle.scr.gkr-west.scp-in.com
Docker login 시 접속정보가 담긴 config.json 파일이 생성됨

2) CLI
$ docker tag test:latestacrcregistry-eedmykle.scr.gkr-west.scp-in.com/<리포지토리>:<태그>
$ docker push acrcregistry-eedmykle.scr.gkr-west.scp-in.com/<리포지토리>:<태그>
$ docker pull acrcregistry-eedmykle.scr.gkr-west.scp-in.com/<리포지토리>:<태그>

3) Create Secret
$ kubectl create secret -n acrc generic private-registry
--from-file=.dockerconfigjson=/root/snap/docker/2893/.docker/config.json
--type=kubernetes.io/dockerconfigjson
$ kubectl get secret -A

실제 deploy를 통해 pod를 배포할 때 private registry에 있는 이미지를 pull 받아 하기 때문에 private registry에 접근하기 위해 secret 생성 필요(docker login 시 생성되는 config.json 파일을 이용해 생성)

3.1.4. PV, PVC

- PV : Kubernetes cluster 내에서 사용하는 storage
- PVC : POD가 PV에 하는 요청

1) PV, PVC yaml 작성

$ vi /root/Kubernetes/common/volume/volume.yaml

root@mconsole: ~/k8s/common/volume

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: acrc-pv
  namespace: acrc
  labels:
    name: acrc-pv
spec:
  storageClassName: "acrc-storage-class"
  capacity:
    storage: 400Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  nfs:
   server: 192.168.80.41
   path: /data


---

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: acrc-pvc
  namespace: acrc
spec:
  storageClassName: "acrc-storage-class"
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 400Gi
  selector:
    matchLabels:
      name: acrc-pv
~
```

2) PV, PVC 적용

$ kubectl apply -f volume.yaml

$ kubectl get pv,pvc -A

※ Namespace
K8S에서 기본적으로 제공되는 Namespace로 kube-system, kube-public, default등이 있고, 순서대로 K8S Core Resources, Sharing Resources, Application등이 각각 배치된다.

1) kube-system
K8S CNCF(Cloud Native Computing Foudation) OpenSource Project에서 기본적으로 제공하는 coredns, kube-proxy, metrics-server 이외, 선택적 calico와 같은 CNI(Container Network Interface)의 구현체인 Network Service등은 일반적으로 kube-system namespace에서 동작한다.

- calico
- coredns
- kube-proxy
- metrics-server
- nfs
- node-exporter
- vpn

2) kube-public
N/A

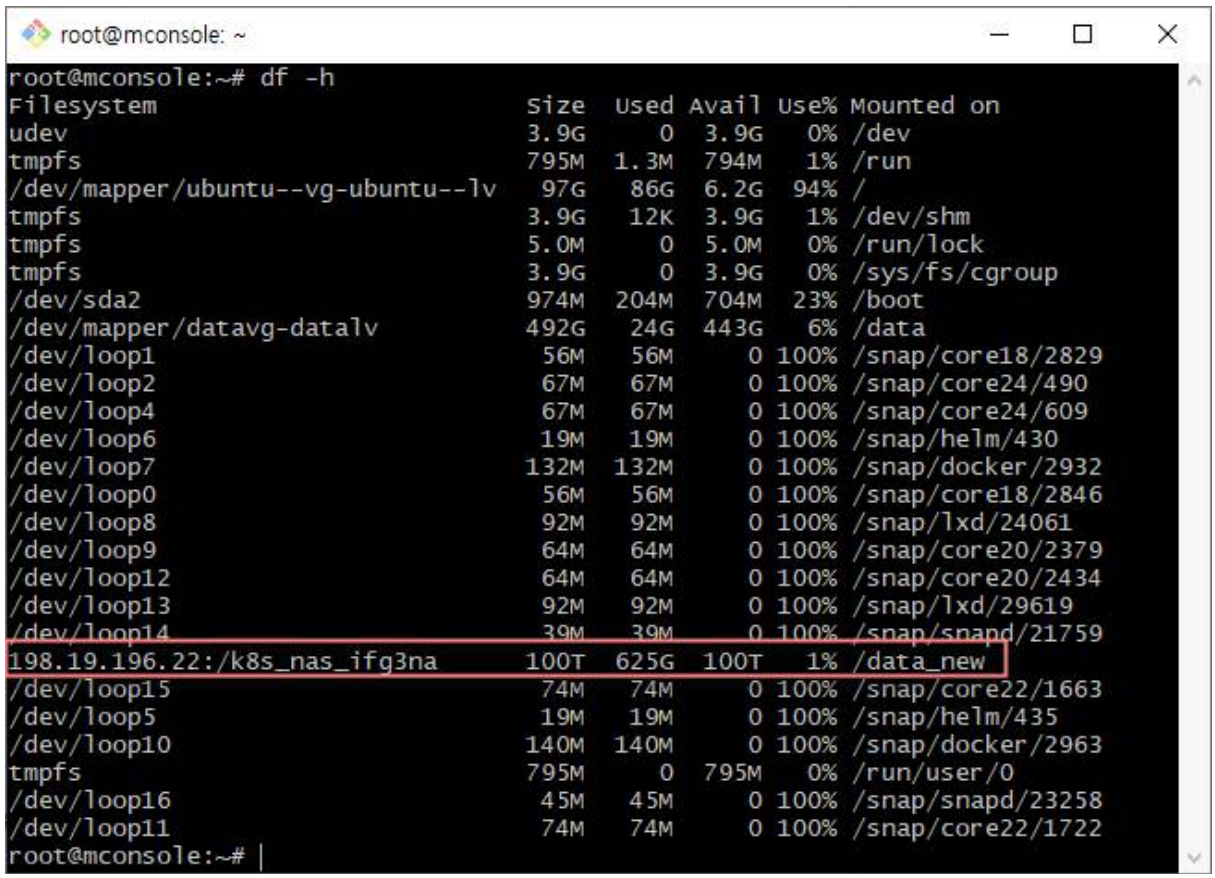
3) default
N/A

4) acrc
상담APP 및 KMS POD들은 이 해당 acrc namespace에서 동작한다.

※ NFS
198.19.196.22
/k8s_nas_ifg3na/hkcloud

1) monitoring server (nfs-server)
- 필요 모듈 설치
$ apt-get install nfs-kernel-server 확인
$ systemctl status nfs-server dir mount access 허용
$ vi /etc/exports => /data 192.168.90.2(rw)
$ vi /etc/exports => /data 192.168.90.3(rw)
$ vi /etc/exports => /data 192.168.90.4(rw)
$ showmount -e 확인
node server (nfs-client) 필요 모듈 설치
$ yum install nfs-utils 확인
$ systemctl status nfs-client dir mount
$ mount -t nfs 192.168.80.41:/data /data
$ df -h => /data mount 확인
$ vi /etc/fstab => 192.168.80.41:/data /data nfs 추가 (재부팅 시에도 mount)

pod 내 올라가 있는 어플리케이션(상담APP-palette, kms)의 log, file 등 공유하기 위해 nfs 설정



3.2. Container Orchestration Layer

Application 설계영역으로 Deployment나 StatefulSet 또는 ReplicatSet을 통해 배포하면, POD들이 생성되고, 이들이 Service객체를 통해 Network에 노출된다.

3.2.1. Critical Component

1) POD

POD는 기본적으로 Inline-way인 kubectl run pod1 —image=registry1.com:/app1:latest 와 같은 명령으로 구동 가능하지만 실무적으로는 yaml-way 즉, kubectl apply –f deployment.yml 과 같은 방식을 통해 배포한다.

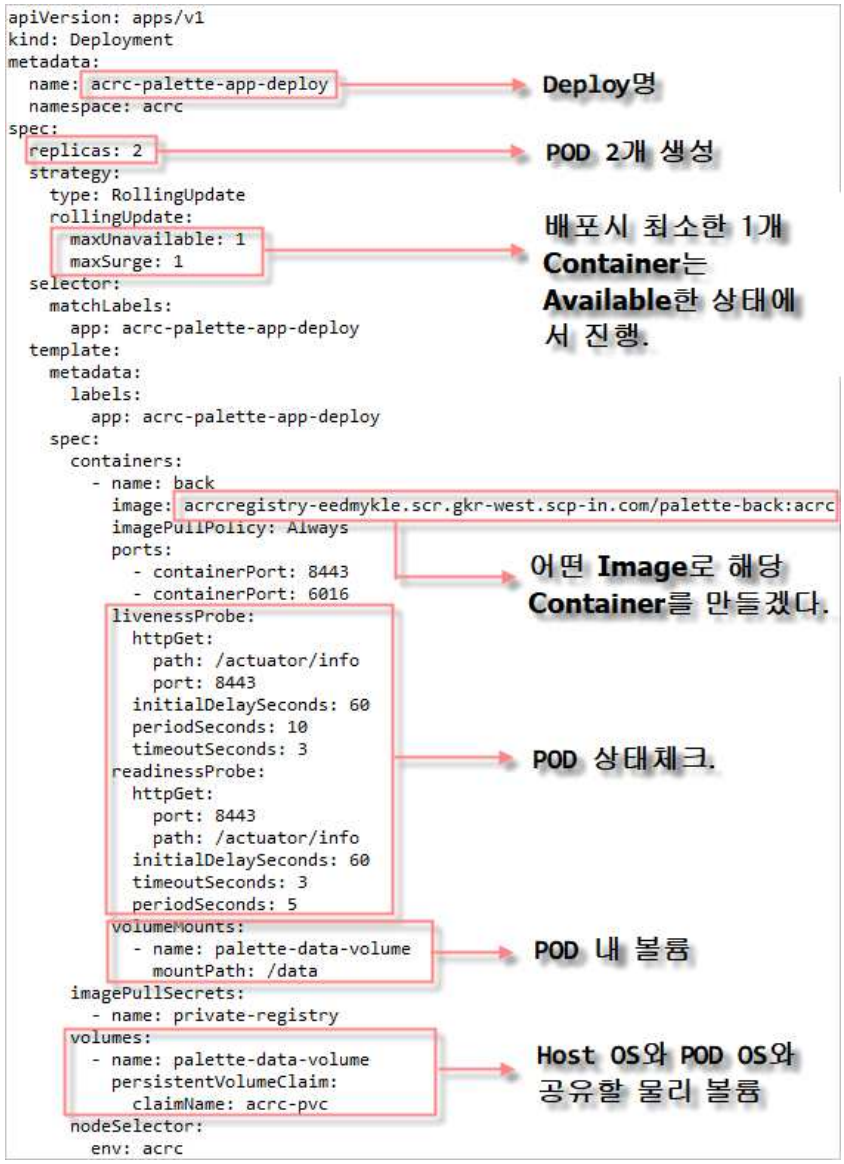
deployment.yaml에서 사용하는 총 3가지 Resource 옵션으로 StatefulSet, DaemonSet, Deployment가 있다. 이중 DaemonSet이나 StatefulSet은 일반적으로 Helm과 같은 도구를 통한 수준에서 사용하고,
실제 개발자가 직접 작성하는 영역은 Deployment이다.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: acrc-palette-deploy
  namespace: default
spec:
  replicas: 2
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 1
  selector:
    matchLabels:
      app: acrc-palette-deploy
  template:
    metadata:
      labels:
        app: acrc-palette-deploy
    spec:
      hostNetwork: true
      containers:
      - name: back
        image: acrcregistrydev-tdbvvmwx.scr.gkr-west.scp-in.com/palette-back:latest
        imagePullPolicy: Always
        ports:
        - containerPort: 9002
          hostPort: 9002
        volumeMounts:
        - name: palette-back
          mountPath: /data/palette/logs/web # POD 로그 경로
      volumeMounts:
      - name: palette-admin
        mountPath: /logs/palette/admin/acrc/logs  # POD 로그 경로
    imagePullSecrets:
    - name: scp-secret
    volumes:
    - name: palette-front
      hostPath:
        path: /data/palette/logs/front  # 호스트의 로그 저장 경로
        type: DirectoryOrCreate
    - name: palette-back
```

2) Deployment

1) 권익위워원회
acrc-palette-web-deployment.yaml
acrc-palette-app-deployment.yaml
2) 농림축산검역본부
apqa-palette-web-deployment.yaml
apqa-palette-app-deployment.yaml
3) ASP국민Call
asp-palette-web-deployment.yaml
asp-palette-app-deployment.yaml
4) 국가문화유산청
khs-palette-web-deployment.yaml
khs-palette-app-deployment.yaml
5) 기상청
kma-palette-web-deployment.yaml
kma-palette-app-deployment.yaml
6) 환경부
me-palette-web-deployment.yaml
me-palette-app-deployment.yaml
7) 여성가족부
mogef-palette-web-deployment.yaml
mogef-palette-app-deployment.yaml
8) 국토부
molit-palette-web-deployment.yaml
molit-palette-app-deployment.yaml
9) 농림진흥청
rda-palette-web-deployment.yaml
rda-palette-app-deployment.yaml

예제. acrc-palette-app-deployment.yaml



3) Service

type

+ ClusterIP : Kubernetes cluster 내에서만 해당 서비스에 접근 가능
+ NodePort : Kubernetes cluster 내에 있는 모든 node 서버의 node port로 접근 가능
+ LoadBalancer : LB 서버로 접근 가능 (클라우드 제공 업체마다 사용 방법 다름)

1) 권익위원회
acrc-palette-web-service.yaml
acrc-palette-app-service.yaml
2) 농림축산검역본부
apqa-palette-web-service.yaml
apqa-palette-app-service.yaml
3) ASP국민Call
asp-palette-web-service.yaml
asp-palette-app-service.yaml
4) 국가문화유산청
khs-palette-web-service.yaml
khs-palette-app-service.yaml
5) 기상청
kma-palette-web-service.yaml
kma-palette-app-service.yaml
6) 환경부
me-palette-web-service.yaml
me-palette-app-service.yaml
7) 여성가족부
mogef-palette-web-service.yaml
mogef-palette-app-service.yaml
8) 국토부
molit-palette-web-service.yaml
molit-palette-app-service.yaml
9)농림진흥청
rda-palette-web-service.yaml
rda-palette-app-service.yaml

예제. acrc-palette-app-service.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: acrc-palette-app-svc
  namespace: acrc
  annotations:
    cloud.samsungsds.com/load-balancer-x-forwarded-for: "INSERT"
    cloud.samsungsds.com/load-balancer-type: "internal"
    cloud.samsungsds.com/load-balancer-service-ip: 192.168.104.25
    cloud.samsungsds.com/load-balancer-client-cert-id: "CERT-RYl5usVWqshREbx-lOwU4a"
    cloud.samsungsds.com/load-balancer-monitor-interval: "600"
    cloud.samsungsds.com/load-balancer-monitor-timeout: "300"
    cloud.samsungsds.com/load-balancer-monitor-count: "1"
spec:
  type: LoadBalancer
  clusterIP:  172.20.145.85
  selector:
    app: acrc-palette-app-deploy
  ports:
    - name: back
      port: 8443
      targetPort: 8443
      protocol: TCP
    - name: socket
      port: 6016
      targetPort: 6016
      protocol: TCP
```

K8S 내부 네트워크에서만 인식되는 IP

POD 포트 노출되는 포트

4) StatefulSet
고정된 ip, 이름 등을 사용

Redis Cluster
Redis cluster는 k8s의 open source package manager인 helm을 사용하여 설치
해당 open source에서 docker.io/bitnami/redis-cluster:7.2.1-debian-11-r0 버전 사용하기 때문에 인터넷 되는 server에서 해당 버전 이미지 반입 필요

$ cd /root/k8s/common/redis/cluster
$ helm fetch bitnami/redis-cluster
$ tar -zxvf redis-cluster-9.0.5.tgz
$ cd redis-cluster
$ vi values.yaml
ㄴ namespace 검색하여 “acrc”로 수정
ㄴ podSecurityContext.fsGroup, runAsUser를 nfs-server의 redis 계정 id, gid에 맞게 수정
ㄴ containerSecurityContext.runAsUser도 위처럼 수정
ㄴ service.type을 ClusterIP로 수정
ㄴ service.clusterIP : "172.20.216.27" 설정
ㄴ storageClass를 acrc-redis-cluster-sc로 수정
ㄴ metrics.image.registry, repository, tag를 사용할 저장소 및 repository, tag에 맞게 수정
ㄴ acrcregistry-eedmykle.scr.gkr-west.scp-in.com/redis-cluster:acrc
ㄴ global.imagePullSecrets, image.pullSecrets에 registry-secret 추가

ㄴ secret, registry 등 관련 에러 나오면 $ vi templates/redis-statefulset.yaml
ㄴ spec.spec.container.image에
acrcregistry-eedmykle.scr.gkr-west.scp-in.com/redis-cluster:acrc 수정
ㄴ spec.spec.imagePullSecrets에 registry-secret 추가

3-2) redis-cluster pv 생성

$ vi /root/k8s/common/redis/cluster/volume/redis-cluster-pv.yaml

root@mconsole: ~/k8s/common/redis/cluster/volume

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: acrc-redis-pv-0
  namespace: acrc
  labels:
    name: acrc-redis-pv-0
spec:
  storageClassName: "acrc-redis-cluster-sc"
  capacity:
    storage: 2Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  nfs:
   server: 192.168.80.41
   path: /data/redis/cluster-0

---

apiVersion: v1
kind: PersistentVolume
metadata:
  name: acrc-redis-pv-1
  namespace: acrc
  labels:
    name: acrc-redis-pv-1
spec:
  storageClassName: "acrc-redis-cluster-sc"
  capacity:
    storage: 2Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  nfs:
   server: 192.168.80.41
   path: /data/redis/cluster-1

---

apiVersion: v1
kind: PersistentVolume
metadata:
  name: acrc-redis-pv-2
  namespace: acrc
  labels:
    name: acrc-redis-pv-2
spec:
  storageClassName: "acrc-redis-cluster-sc"
  capacity:
    storage: 2Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  nfs:
   server: 192.168.80.41
   path: /data/redis/cluster-2

---
```

ㄴ pv node 수에 맞게 생성 (default 6개 생성해야 함)

ㄴ nfs server를 따로 구축해 해당 서버에서 로그, 파일 등 관리하기 때문에 SC(StorageClass)를 사용하지 않고 수동으로 개수에 맞게 PV를 만들어 redis-cluster chart PVC와 맵핑

$ mkdir /data/redis/cluster-0~5

4-3) redis-cluster 생성

$ helm install -n acrc redis-cluster /root/k8s/common/redis/cluster/redis-cluster

```
root@mconsole:~/k8s/kms/dev# kubectl get po -A
NAMESPACE     NAME                                       READY   STATUS      RESTARTS   AGE
acrc          kms-dev-deploy-6668855dd8-25nmw            1/1     Running     0          5h26m
acrc          palette-back-dev-deploy-fdd8c77c9-c7zzt    1/1     Running     0          5h20m
acrc          palette-front-dev-deploy-76f7cfb5cf-5jbnk  1/1     Running     0          5h18m
acrc          redis-cluster-0                            1/1     Running     0          5h11m
acrc          redis-cluster-1                            1/1     Running     0          5h11m
acrc          redis-cluster-2                            1/1     Running     0          5h11m
acrc          redis-cluster-3                            1/1     Running     0          5h11m
acrc          redis-cluster-4                            1/1     Running     0          5h11m
acrc          redis-cluster-5                            1/1     Running     0          5h11m
acrc          redis-deployment-5cd5bdcb6-5r9g1           1/1     Running     0          8d
controller    ingress-controller-9c68658b-8gvj7          1/1     Running     0          14d
gitlab        acrc-gitlab-97c6865dc-glpgj                1/1     Running     0          18d
jenkins       acrc-jenkins-2f8lh                         0/1     Completed   0          8d
jenkins       acrc-jenkins-f688bf86f-gtj29               1/1     Running     0          8d
kube-system   calico-kube-controllers-67cd6c67d4-qsb2m   1/1     Running     0          20d
kube-system   calico-node-wdzwl                          1/1     Running     0          20d
kube-system   coredns-75cfdfd5cc-bbbb7                   1/1     Running     0          20d
kube-system   coredns-75cfdfd5cc-qjzcv                   1/1     Running     0          20d
kube-system   kube-proxy-2zngx                           1/1     Running     0          20d
kube-system   metrics-server-847459cb4-6d2dm             1/1     Running     0          20d
kube-system   nfs-subdir-external-provisioner-5bff5d8d99-pp8f4   1/1     Running     0          20d
kube-system   node-exporter-ph6zh                        1/1     Running     0          20d
kube-system   vpn-target-677ddf9859-v7bnm                1/1     Running     0          20d
```

3.2.2. Miscelleanous

1) Liveness & Readiness

Process가 살아는 있는지, 살아는 있는데 가용한 상태인지를 체크.
initialDelaySeconds는 초기화 되는데까지 응답이 없더라도 오류라 간주 안하고 대기.
timeoutSeconds와 periodSeconds는 얼마의 주기로 응답시간을 체크해 답이 없을 경우 실패로 간주하겠다.

```
livenessProbe:
  httpGet:
    path: /actuator/info
    port: 8443
  initialDelaySeconds: 60
  periodSeconds: 10
  timeoutSeconds: 3
readinessProbe:
  httpGet:
    port: 8443
    path: /actuator/info
  initialDelaySeconds: 60
  timeoutSeconds: 3
  periodSeconds: 5
```

2) K8S Client

K8S Console로 KMS 및 상담APP 관련한 이 Client는 192.168.80.41에서 Control한다.

Essential Commands

1. POD 상태
- kubectl get pods

2. POD 재시작
kubectl scale deployment khs-palette-web-deploy --replicas=0 –n acrc
kubectl scale deployment me-palette-app-deploy --replicas=2 –n acrc

3. Re-deploy
(cd /root/k8s/palette/yaml/deployment; kubectl delete deployments molit-palette-web-deploy -n acrc; kubectl apply –f molit-palette-web-deployment.yaml;)

4. Log
kubectl logs <podname>

5. POD OS에 Interactive Mode로 진입.
kubectl exec –it <podname> -c <containername> -- bash

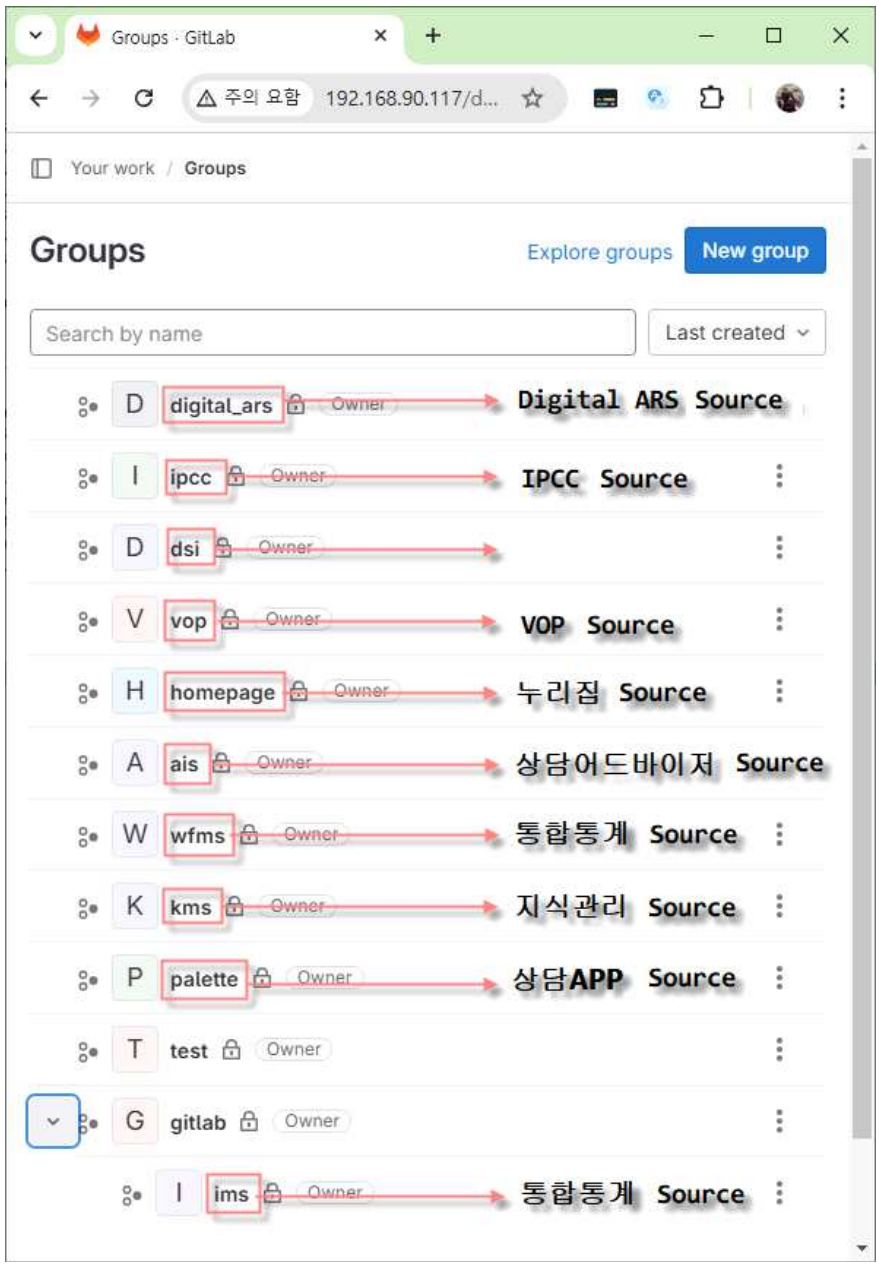
4. CI/CD 구성

Jenkins는 GitLab으로부터 소스를 취합하여, Java build 및 npm build 등 Compile후, .war나 .jar, Docker Image로 생성, K8S 배포.

4.1. GitLab

기능 : 콜센터 각종 어플리케이션 소스 Repository.
URL : http://192.168.90.117

종료 : /usr/bin/gitlab-ctl stop
구동 : /usr/bin/gitlab-ctl start



4.2. Jenkins

기능 : 콜센터 각종 어플리케이션 Build & Deploy, Server Restart.
URL : http://192.168.90.117:8081

종료 : ps -ef | grep jenkins | awk '{print $2}' | xargs kill -9
구동 : export JENKINS_HOME=/ims/jenkins_home
nohup /workspace/tools/jdk/jdk-11.0.23/bin/java
-Dcom.sun.management.jmxremote.local.only=false
-Dcom.sun.management.jmxremote=true
-Dcom.sun.management.jmxremote.rmi.port=19102
-Dcom.sun.management.jmxremote.port=19103
-Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote.authenticate=false
-Djava.rmi.server.hostname=192.168.90.117 -Xms2048m -Xmx3072m -jar /ims/jenkins.war
--httpPort=8081 &