**<< Helm chart >>**

Project Links

* Helm Code: <https://github.com/Kubernetes/helm>
* Charts Code: <https://github.com/Kubernetes/Charts>
* Documentation: [https://docs.helm.sh](https://docs.helm.sh/)

**▲ Helm 패키지 매니저**

ㅇ Helm은 쿠버네티스 패키지 매니저임

ㅇ helm 사용하면 클러스터에 Tiller라는 팟이 설치되고, 이 팟을 통해 Helm 패키지(이하 차트) 내부에 정의한 쿠버네티스 리소스들을 클러스터에 올릴 수 있음.

ㅇ Helm에서는 템플릿 기능을 사용하여 도메인에 따라 쉽게 설정 변경 가능

ㅇ CS 구조

- client

. client는 end user를 위한 command line client

. 주로 local chart 개발이나 repository managing

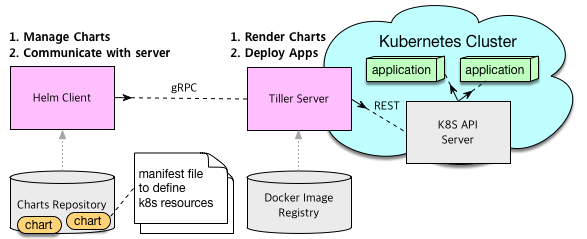
. server(tiler)에 chart 설치 요청 등 주로 chart를 관리

- server(tiller)

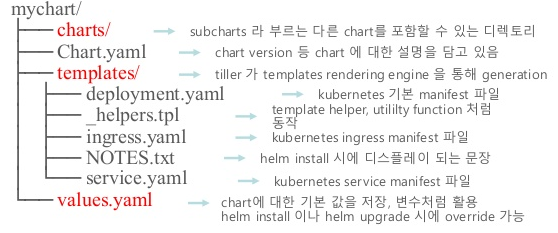
. in-cluster 서버

. chart의 배포/ 릴리즈를 관리

ㅇ 흐름도



ㅇ helm chart directory tree



**▲ Helm 설치**

ㅇ 설치 script 다운 및 실행

snap, homebrew 등 다양한 설치 방법이 존재하지만 script 방식이 가장 간단한 듯.

|  |
| --- |
| $ curl https://raw.githubusercontent.com/helm/helm/master/scripts/get > get\_helm.sh  $ chmod 700 get\_helm.sh  $ ./get\_helm.sh |

**▲ tiller 설치**

tiller 설치의 의미는 tiller-deploy 라는 deployment 와 service 를 kube-system 에서 실행시키는 행위 이다. 설치작업이 끝난 후 kubectl -n kube-system get deployment 명령으로 확인 할 수 있다. uninstall 의 경우는 해당 deployment 를 삭제하면 된다.

ㅇ 아래 3가지 형태로 설치할 수 있다.

1) 기본설치

minikube인 경우 또는 별도 보안설정이 필요없을때 사용한다.

serviceaccount 가 기본(default)로 설치된다.

|  |
| --- |
| $ helm init  # dry-run 으로 simulation 해 볼 수 있다.  $ helm init --dry-run --debug |

2) RBAC 이용 보안설치 : cluster-admin role 이용

kube-system 네임스페이스에 tiller sa 생성 및 clusterrolebinding

|  |
| --- |
| $ kubectl -n kube-system create sa tiller  $ kubectl create clusterrolebinding tiller \  --clusterrole cluster-admin \  --serviceaccount=kube-system:tiller  # tiller-deploy deployment 실행  $ helm init --service-account tiller |

3) RBAC 이용 보안설치 : 특정 네임스페이스에만 수행되도록.

ClusterRole 또는 ClusterRoleBinding을 지정하는 대신 역할 및 역할 바인딩을 지정하여 Tiller의 범위를 특정 네임 스페이스로 제한 할 수 있다.

|  |
| --- |
| $ kubectl create namespace tiller-world  namespace "tiller-world" created  $ kubectl create serviceaccount tiller --namespace tiller-world  serviceaccount "tiller" created  $ cat > role-tiller.yaml  ---  kind: Role  apiVersion: rbac.authorization.k8s.io/v1  metadata:  name: tiller-manager  namespace: tiller-world  rules:  - apiGroups: ["", "batch", "extensions", "apps"]  resources: ["\*"]  verbs: ["\*"]  ---  kind: RoleBinding  apiVersion: rbac.authorization.k8s.io/v1  metadata:  name: tiller-binding  namespace: tiller-world  subjects:  - kind: ServiceAccount  name: tiller  namespace: tiller-world  roleRef:  kind: Role  name: tiller-manager  apiGroup: rbac.authorization.k8s.io  ---  $ kubectl create -f rolebinding-tiller.yaml  rolebinding "tiller-binding" created  # tiller 설치  $ helm init --service-account tiller --tiller-namespace tiller-world  # 샘플 install  $ helm install nginx --tiller-namespace tiller-world --namespace tiller-world |

ㅇ 다른 네임스페이스에서 tiller 제한

myorg-system 네임 스페이스에 Tiller를 설치하고 Tiller가 myorg-users 네임 스페이스에 리소스를 배포하도록 허용하는 예제

|  |
| --- |
| $ kubectl create namespace myorg-system  $ kubectl create serviceaccount tiller --namespace myorg-system  $ rolebinding-tiller.yaml  ---  kind: Role  apiVersion: rbac.authorization.k8s.io/v1  metadata:  name: tiller-manager  namespace: myorg-users  rules:  - apiGroups: ["", "batch", "extensions", "apps"]  resources: ["\*"]  verbs: ["\*"]  ---  kind: RoleBinding  apiVersion: rbac.authorization.k8s.io/v1  metadata:  name: tiller-binding  namespace: myorg-users  subjects:  - kind: ServiceAccount  name: tiller  namespace: myorg-system  roleRef:  kind: Role  name: tiller-manager  apiGroup: rbac.authorization.k8s.io  ---  $ kubectl create -f rolebinding-tiller.yaml |

ㅇ troble shooting

|  |
| --- |
| # tiller forbidden 오류 발생시 아래와 같이 sa 를 tiller deployment 에 추가해야 함.  $ kubectl patch deploy --namespace kube-system tiller-deploy -p '{"spec":{"template":{"spec":{"serviceAccount":"tiller"}}}}' |

ㅇ helm init --output yaml

helm init 명령어가 어떤 형식의 manifest 파일을 만들어서 tiller를 kubernetes cluster 위에 베포하는지 yaml 파일을 생성

|  |
| --- |
| $ helm init --output yaml  ---  apiVersion: extensions/v1beta1  kind: Deployment  metadata:  creationTimestamp: null  labels:  app: helm  name: tiller  name: tiller-deploy  namespace: kube-system  spec:  replicas: 1  strategy: {}  template:  metadata:  creationTimestamp: null  labels:  app: helm  name: tiller  spec:  automountServiceAccountToken: true  containers:  - env:  - name: TILLER\_NAMESPACE  value: kube-system  - name: TILLER\_HISTORY\_MAX  value: "0"  image: gcr.io/kubernetes-helm/tiller:v2.12.3  imagePullPolicy: IfNotPresent  livenessProbe:  httpGet:  path: /liveness  port: 44135  initialDelaySeconds: 1  timeoutSeconds: 1  name: tiller  ports:  - containerPort: 44134  name: tiller  - containerPort: 44135  name: http  readinessProbe:  httpGet:  path: /readiness  port: 44135  initialDelaySeconds: 1  timeoutSeconds: 1  resources: {}  status: {}  ---  apiVersion: v1  kind: Service  metadata:  creationTimestamp: null  labels:  app: helm  name: tiller  name: tiller-deploy  namespace: kube-system  spec:  ports:  - name: tiller  port: 44134  targetPort: tiller  selector:  app: helm  name: tiller  type: ClusterIP  status:  loadBalancer: {} |

ㅇ openshift 의 경우는 아래 참조

<https://blog.openshift.com/getting-started-helm-openshift/>

|  |
| --- |
| Step 1: Create an OpenShift project for Tiller  $ oc new-project tiller  $ oc project tiller  $ export TILLER\_NAMESPACE=tiller  Step 2: Install the Helm client locally  $ curl -s https://storage.googleapis.com/kubernetes-helm/helm-v2.9.0-linux-amd64.tar.gz | tar xz  $ cd linux-amd64  $ ./helm init --client-only  Step 3: Install the Tiller server  $ oc process -f https://github.com/openshift/origin/raw/master/examples/helm/tiller-template.yaml -p TILLER\_NAMESPACE="${TILLER\_NAMESPACE}" -p HELM\_VERSION=v2.9.0 | oc create -f -  $ oc rollout status deployment tiller  $ ./helm version  Step 5: Grant the Tiller server edit access to the current project.  $ oc policy add-role-to-user edit "system:serviceaccount:${TILLER\_NAMESPACE}:tiller" |

**▲ Chart 설치 (helm install ~~)**

ㅇ 설치 가능한 chart 목록 search

|  |
| --- |
| $ helm search  $ helm search mysql  $ helm inspect stable/mariadb |

ㅇ helm install

|  |
| --- |
| $ helm install stable/mariadb ← defualt namespace에 설치된다.  # 특정 release 이름 지정시 --name 옵션 추가  $ helm install stable/mariadb --name my-maria  $ helm install nginx --tiller-namespace tiller-world --namespace tiller-world  $ helm install . --name elasticsearch --namespace song-lab  # install 하기전 설치될 yaml 파일을 미리 확인  $ helm install . --name elasticsearch --namespace song-lab --dry-run --debug |

* 이름 지정

같은 chart를 여러번 install 할수 있는데 그때마다 새로운 release object 가 생성되어 override 된다.

|  |
| --- |
| <첫번째 install >  $ helm install stable/mariadb  NAME: jaunty-ostrich  <두번째 install >  $ helm install stable/mariadb  NAME: sullen-ladybird  특정이름으로 명시할 수 있다.  helm install . --name [특정이름] |

ㅇ helm status

|  |
| --- |
| $ helm status my-maria |

ㅇ helm list : 클러스터에 설치된 릴리스 목록

|  |
| --- |
| $ helm list  NAME REVISION UPDATED STATUS CHART NAMESPACE  wishful-mastiff 1 Thu Jul 12 15:53:56 2018 DEPLOYED wordpress-2.1.3 default  NAME REVISION UPDATED STATUS CHART APP VERSION NAMESPACE  calico-clam 1 Wed Mar 13 11:36:44 2019 DEPLOYED redis-6.3.0 4.0.13 neocode24 |

ㅇ helm delete

|  |
| --- |
| $ helm del --purge elasticsearch |

ㅇ helm upgrade

chart의 새버젼이 나오거나 이미 release한 chart의 설정값을 변경하고 싶을때는 helm upgrade 명령어로 수행 가능

|  |
| --- |
| $ helm ls logstash  NAME REVISION UPDATED STATUS CHART APP VERSION NAMESPACE  logstash 1 Wed Mar 20 00:23:38 2019 DEPLOYED logstash-1.5.2 6.6.1 song-lab  # value 내용 수정  예를들면 replicaCount 수를 1 → 2로 조정  $ helm upgrade logstash .  $ ks get pod  NAME READY STATUS RESTARTS AGE  logstash-0 1/1 Running 0 13m  logstash-1 1/1 Running 0 1m  $ helm ls logstash  NAME REVISION UPDATED STATUS CHART APP VERSION NAMESPACE  logstash 2 Wed Mar 20 00:23:38 2019 DEPLOYED logstash-1.5.2 6.6.1 song-lab  $ helm history logstash  REVISION UPDATED STATUS CHART DESCRIPTION  1 Wed Mar 20 00:23:38 2019 SUPERSEDED logstash-1.5.2 Install complete  2 Wed Mar 20 00:36:30 2019 DEPLOYED logstash-1.5.2 Upgrade complete |

ㅇ helm rollback

|  |
| --- |
| # revision 1로 rollback 하기  $ helm rollback logstash 1  $ helm history logstash  REVISION UPDATED STATUS CHART DESCRIPTION  1 Wed Mar 20 00:23:38 2019 SUPERSEDED logstash-1.5.2 Install complete  2 Wed Mar 20 00:36:30 2019 SUPERSEDED logstash-1.5.2 Upgrade complete  3 Wed Mar 20 00:38:12 2019 DEPLOYED logstash-1.5.2 Rollback to 1 |

**▲ helm 기타 명령어**

ㅇ chart fetch

repository에 등록된 chart를 tar ball형태로 down

tarball의 압축을 풀면 chart의 구성을 확인 가능

|  |
| --- |
| $ helm fetch stable/mariadb  helm fetch stable/elasticsearch  helm fetch stable/elasticsearch-curator  helm fetch stable/elasticsearch-exporter  helm fetch stable/fluentd-elasticsearch  helm fetch stable/fluentd  helm fetch stable/kibana  helm fetch stable/logstash |

mariadb 조회해서 다운받아 install 하기

|  |
| --- |
| # mariadb 가 local 저장소에 존재하는지 확인  $ helm search stable  $ helm search stable/mariadb  # local directory 로 받아오기  $ helm fetch mariadb  $ tar -zxvf mariadb-5.2.4.tgz  $ tree mariadb  - helm install  $ helm install ./mariadb --name mariadb-default --tls  - helm release 확인  $ helm list mariadb-default --tls |

ㅇ **helm lint**

문법 체크 : chart 내에 존재하는 yaml들의 indent 및 문법이 잘못되어 있거나 kubernetes 객체를 정의하는 template 언어의 문법이 잘못되어 있는 것을 미리 체크해 볼 수 있음

|  |
| --- |
| $ helm lint  ==> Linting .  [ERROR] templates/: parse error in "drupal/templates/deployment.yaml": template: drupal/templates/deployment.yaml:26: unexpected  Error: 1 chart(s) linted, 1 chart(s) failed |

ㅇ helm history

|  |
| --- |
| $ helm history my-release |

**▲ provider helm repogitory 관리**

- Helm repogitory는 Helm 차트를 저장할 수 있는 HTTP 서버임

- 클러스터를 제공하는 provider 에서 repogitory 지원이 가능한지 문의 필요.

- provider : Azure Container Registry, Google, IBM 등도 별도 존재 함.

ㅇ stable 리포지토리

일반적인 응용 프로그램 차트를 포함하는 공용 리포지트리 임

ㅇ 기타 명령

|  |
| --- |
| $ az acr helm show wordpress  # chart push  $ az acr helm push wordpress-2.1.10.tgz  $ az acr helm repo add |

ㅇ helm repo

|  |
| --- |
| $ helm repo update  Hang tight while we grab the latest from your chart repositories...  ...Skip local chart repository  ...Successfully got an update from the "stable" chart repository  Update Complete. ⎈ Happy Helming!⎈  # 기타  $ helm repo update : 최신 chart 리스트 업데이트  $ helm repo list : 조회  NAME URL  stable https://kubernetes-charts.storage.googleapis.com  local http://127.0.0.1:8879/charts  $ helm repo add localhost http://localhost:8879/charts : localhost라는 이름의 chart repository 등록  $ helm repo remove localhost : localhost라는 이름의 chart repository 삭제 |

**▲ trouble shooting**

ㅇ 오류1: helm install 과정에서 오류 발생시

|  |
| --- |
| ㅇ 오류내용 : forbidden: User "system:serviceaccount:kube-system:default" cannot get namespaces in the namespace    ㅇ 원인 : tiller-deploy pod 를 수행하는 권한 문제  ㅇ 조치 :  kubectl patch deploy --namespace kube-system tiller-deploy -p '{"spec":{"template":{"spec":{"serviceAccount":"tiller"}}}}' |

**▲ [예제1] helm 으로 mariadb 배포하기**

ㅇ helm chart 받아오기

|  |
| --- |
| # mariadb 가 local 저장소에 존재하는지 확인  $ helm search stable  $ helm search stable/mariadb  # 없으면 repository 를 생성한다.  # 서버의 tiller 새성하지 않으려면 client-only 옵션을 준다.  $ helm init --client-only  # local directory 로 받아오기  $ helm fetch mariadb  $ tar -zxvf mariadb-5.2.4.tgz  $ tree mariadb  mariadb  ├── Chart.yaml  ├── files  │ └── docker-entrypoint-initdb.d  │ └── README.md  ├── OWNERS  ├── README.md  ├── templates  │ ├── \_helpers.tpl  │ ├── initialization-configmap.yaml  │ ├── master-configmap.yaml  │ ├── master-statefulset.yaml  │ ├── master-svc.yaml  │ ├── NOTES.txt  │ ├── secrets.yaml  │ ├── slave-configmap.yaml  │ ├── slave-statefulset.yaml  │ ├── slave-svc.yaml  │ ├── test-runner.yaml  │ └── tests.yaml  ├── values-production.yaml  └── values.yaml |

ㅇ helm install

helm install 시 현재 namespace 로 설치되기때문에

반드시 namespace 확인하여 원하는 namespace 가 아니면 변경 필요

|  |
| --- |
| - helm install  $ helm install ./mariadb --name mariadb-default --tls  - helm release 확인  $ helm list mariadb-default --tls |

▲

▲