developer.skatelescope.org Documentation

Release 0.1.0-beta

Marco Bartolini

Sep 20, 2023

HOME

:bulb: Deploy a Minikube development environment using `make`.

CHAPTER

ONE

SUMMARY

This repo provides a basic recipe for deploying Minikube as a development environment that will support the standard features required for the SKA Skampi MVP (https://gitlab.com/ska-telescope/ska-skampi). Please note the OS specific variations at the end of this README. Also note that this repository gets updated on a semi-regular basis in order to track the versions of tools, amongst other things, from:

- https://github.com/kubernetes/minikube/releases
- https://github.com/helm/helm/releases
- https://registry.hub.docker.com/_/haproxy
- https://kubernetes.io/releases/

By default, `podman` is used as the driver for Linux as it is the recommended driver for the SKAO, however, it is possible to revert to using the prior Docker driver by passing in `DRIVER=docker` to all `make` invocations.

Also, by default, a set of addons are automatically enabled, these are:

- logviewer - simple logview available on port 32000 - browse with `sensible-browser http://$(minikube ip):32000`
- metrics-server - simple metrics server
- ingress - NGINX based Ingress Controller for exposing HTTP/HTTPS services
- metallb - enable creation of `LoadBlancer` type `Service` resources to expose application ports out of Kubernetes. This is deployed in conjunction with a DNS responder (`extdns`) that can be integrated with the local users DNS settings to have automatic name resolution for these services.

:zap: Note: if you need any help with getting this going, then please provide the full output of any commands you run along with the details of your OS and version in the #team-system-support Slack channel. It also helps to check that you have pulled the latest version of this repository as it does change every 2-3 months.

CHAPTER

TWO

HOW TO USE

:information_source: It is very important that you check the *Caveats* section below to ensure that no major issues - that might affect the stability of the rest of the OS - are encountered.

:information_source: Before you continue, see the *OS Variations* section below to ensure your dependencies are satisfied.

Checkout ska-cicd-deploy-minikube (this repo) with:

```
git clone git@gitlab.com:ska-telescope/sdi/ska-cicd-deploy-minikube.git
cd ska-cicd-deploy-minikube
```

Now explore, using:

```
$ make minikube-vars
Minikube Installed: Yes!
Helm Installed:     Yes!
DRIVER:             podman
RUNTIME:            docker
ADDONS:             --addons=logviewer --addons=metrics-server --addons=ingress --
↪addons=metallb
CPUS:               6
MEM:                8192
OS_NAME:            linux
OS_ARCH:            x86_64
OS_BIN:             amd64
EXE_DIR:            /usr/local/bin
IPADDR:             10.115.201.231
MINIKUBE_IP:        * Profile minikube not found. Run minikube profile list to view all␣
↪profiles.   To start a cluster, run: minikube start
HOSTNAME:           wattle
FQDN:               wattle.local.net
MOUNT_FROM:         /srv
MOUNT_TO:           /srv
PROXY_VERSION:      2.4
PROXY_CONFIG:       /home/piers/.minikube/minikube-nginx-haproxy.cfg
MINIKUBE_VERSION:   v1.25.2
KUBERNETES_VERSION: v1.23.5
HELM_VERSION:       v3.8.2
YQ_VERSION:         4.14.1
INGRESS:            http://* Profile minikube not found. Run minikube profile list to␣
↪view all profiles. To start a cluster, run: minikube start
USE_CACHE:          yes
```

(continues on next page)

(continued from previous page)

```
CACHE_DATA:         /home/piers/.minikube/registry_cache
Minikube status:
* Profile "minikube" not found. Run "minikube profile list" to view all profiles.
  To start a cluster, run: "minikube start"
```

Inspect the help and defaults with:

```
$ make
preparing the help
make[1]: Entering directory '/home/piers/git/public/ska-telescope/sdi/ska-cicd-deploy-
↪minikube'

------
SECTION: Makefile
MAKE TARGETS:
all                             do setup and install
bats-reinstall                  reinstall bats-core dependencies
bats-test                       Run unit tests using BATS
clean                           alias for minikube-clean
minikube-cache                  docker image caching proxy
minikube-clean-cache             clean the cache - WARNING you must make minikube-clean␣
↪Minikube first!!!!
minikube-clean                   Delete Minikube cluster etc. - **caution** there is no␣
↪going back!
minikube-extdns                 Deploy CoreDNS k8s_external responder
minikube-haproxy                Setup haproxy to present the Ingress controller to the␣
↪outside world
minikube-install                 install Minikube, Ingress Controller, Metal LB, and␣
↪HAProxy (on Linux)
minikube-install-podman          Install podman tools, if this Makefile supports your OS.
minikube-kubeconfig             print KUBECONFIG for Minikube and swapping to external IP␣
↪(try: make minikube-kubeconfig > minikube_config; export KUBECONFIG=$(pwd)/minikube_
↪config)
minikube-load-images            load images from the charts listed in K8S_CHARTS
minikube-node-ports             list potential node ports from the running Minikube␣
↪cluster
minikube-registry               Setup a local Docker Registry integrated with Minikube
minikube-setup                   install command line tools for Minikube, kubectl, helm,␣
↪k9s and yq dependencies
minikube-test                    run an interactive  test deployment
minikube-tools                  install HAProxy
minikube-vars                   list the public vars and Minikube status

MAKE VARS (+defaults):
ADDONS                          --addons=logviewer --addons=metrics-server --
↪addons=ingress --addons=metallb ## Minikube addons
CACHE_VERSION                   0.6.0## local registry cache based on NGINX
CI_JOB_ID                       local
CPUS                            4## Number of (v)CPUs to allocate to Minikube
DRIVER                          hyperkit
DRIVER                          podman## Virtualisation layer used to host minikube -␣
↪default podman for Linux, and hyperkit for MacOS
```

(continues on next page)

(continued from previous page)

```
FLAGS                          ## Additional flags to pass to minikube
FQDN                           $(HOST).local.net
HELM_VERSION                   v3.8.2## Helm tool version to install
HOSTNAME                       $(HOST)
IPADDR                         $(IP)
K8S_CHART_PARAMS               ## Parameters to pass to helm template for image discovery
K8S_CHARTS                     ska-tango-base ## Charts to parse for images to preload
K9S_VERSION                    v0.25.18## k9s tool version to install
KUBERNETES_VERSION             v1.23.5## Kubernetes version to bootup
MEM                            8192## Amount of RAM to allocate to Minikube (MB)
MINIKUBE_HOME                  $(HOME)
MINIKUBE_NETWORK               minikube## Minikube network name (Docker Networking)
MINIKUBE_VERSION               v1.25.2## Minikube tool version to install
MOUNT_FROM                     $(MOUNT_FROM_DIR)## Source directory to mount into␣
↪Minikube VM(or CnC)
MOUNT_TO                       /srv##Target directory to mount into Minikube VM(or CnC)
NODES                          1## Number of Nodes for Minikube to create - should␣
↪always be 1
PROXY_VERSION                  2.4## haproxy version to install for frontend proxy
RUNTIME                        cri-o## How containers are run within minikube - docker,␣
↪cri-o or containerd
USE_CACHE                      ## Deploy a local NGiNX based image cache
YQ_VERSION                     4.14.1## yq tool version to install
```

CHAPTER

THREE

INSTALL THE TOOLS

The complete install of tools (dependencies: minikube, kubectl, helm, and k9s), and the deployment of a Minikube cluster can be done with:

```
$ make all
make[1]: Entering directory '/home/piers/git/public/ska-telescope/sdi/ska-cicd-deploy-
↪minikube'
Minikube status:
  Profile "minikube" not found. Run "minikube profile list" to view all profiles.
  To start a cluster, run: "minikube start"
Minikube not running, continuing...
Using driver: podman
Extra ARGS set:
Local mount: /srv:/srv
  minikube v1.23.2 on Ubuntu 21.04
     MINIKUBE_WANTREPORTERRORPROMPT=false
     KUBECONFIG=/home/piers/.kube/config
     MINIKUBE_HOME=/home/piers
     MINIKUBE_WANTUPDATENOTIFICATION=false
  Using the podman driver based on user configuration
  Your cgroup does not allow setting memory.
     More information: https://docs.docker.com/engine/install/linux-postinstall/#your-
↪kernel-does-not-support-cgroup-swap-limit-capabilities
  Starting control plane node minikube in cluster minikube
  Pulling base image ...
E1029 02:49:43.787056 3607127 cache.go:201] Error downloading kic artifacts:  not yet␣
↪implemented, see issue #8426
  Creating podman container (CPUs=2, Memory=8192MB) ...
  Preparing Kubernetes v1.22.3 on Docker 20.10.8 ...
     Generating certificates and keys ...
     Booting up control plane ...
     Configuring RBAC rules ...
  Verifying Kubernetes components...
     Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.0
     Using image ivans3/minikube-log-viewer:latest
     Using image k8s.gcr.io/ingress-nginx/controller:v1.0.0-beta.3
     Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.0
     Using image k8s.gcr.io/metrics-server/metrics-server:v0.4.2
     Using image metallb/controller:v0.9.6
     Using image metallb/speaker:v0.9.6
     Using image gcr.io/k8s-minikube/storage-provisioner:v5
```

(continues on next page)

(continued from previous page)

```
  Verifying ingress addon...
  Enabled addons: logviewer, metallb, metrics-server, storage-provisioner, default-
↪storageclass, ingress
  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by␣
↪default
Apply the standard storage classes
kubectl apply -f ./scripts/sc.yaml
storageclass.storage.k8s.io/nfs created
storageclass.storage.k8s.io/nfss1 created
storageclass.storage.k8s.io/block created
storageclass.storage.k8s.io/bds1 created
make[1]: Leaving directory '/home/piers/git/public/ska-telescope/sdi/ska-cicd-deploy-
↪minikube'
# must run the following again in make to get vars
make[1]: Entering directory '/home/piers/git/public/ska-telescope/sdi/ska-cicd-deploy-
↪minikube'
Apply the metallb config map - prefix: 192.168.49
configmap/config configured
make[1]: Leaving directory '/home/piers/git/public/ska-telescope/sdi/ska-cicd-deploy-
↪minikube'
make[1]: Entering directory '/home/piers/git/public/ska-telescope/sdi/ska-cicd-deploy-
↪minikube'
Patch Ingress Controller for permissions issues:
kubectl -n ingress-nginx patch deployment/ingress-nginx-controller \
--type json   -p='[{"op": "replace", "path": "/spec/template/spec/containers/0/
↪securityContext", "value": {"runAsUser": 0, "allowPrivilegeEscalation": true,
↪"capabilities": {"add": ["NET_BIND_SERVICE"]}}}]'
deployment.apps/ingress-nginx-controller patched
make[1]: Leaving directory '/home/piers/git/public/ska-telescope/sdi/ska-cicd-deploy-
↪minikube'
make[1]: Entering directory '/home/piers/git/public/ska-telescope/sdi/ska-cicd-deploy-
↪minikube'
make[2]: Entering directory '/home/piers/git/public/ska-telescope/sdi/ska-cicd-deploy-
↪minikube'
# Now setup the Proxy to the NGINX Ingress and APIServer, and any NodePort services
# need to know the device and IP as this must go in the proxy config
Installing HAProxy frontend to make Minikube externally addressable
echo "MINIKUBE_IP: 192.168.49.2" && \
echo "${HAPROXY_CONFIG}" | envsubst > /home/piers/.minikube/minikube-nginx-haproxy.cfg; \
export NODE_PORTS="80:80 443:443  "; \
for i in ${NODE_PORTS}; do \
        export PORT=$(echo "$i" | sed 's/.*://'); echo "Adding proxy for NodePort ${PORT}
↪"; echo "${ADD_HAPROXY_CONFIG}" | sed "s/XPORTX/${PORT}/g" >> /home/piers/.minikube/
↪minikube-nginx-haproxy.cfg ; \
    export PORTS="${PORTS} -p ${PORT}:${PORT} "; \
done; \
if [[ "podman" == "docker" ]]; then \
sudo --preserve-env=http_proxy --preserve-env=https_proxy podman run --name minikube-
↪nginx-haproxy --net=minikube \
        -p 6443:6443 ${PORTS} \
        -v /home/piers/.minikube/minikube-nginx-haproxy.cfg:/usr/local/etc/haproxy/
↪haproxy.cfg \
```

(continues on next page)

(continued from previous page)

```
        -d haproxy:2.4 -f /usr/local/etc/haproxy/haproxy.cfg; \
else \
sudo --preserve-env=http_proxy --preserve-env=https_proxy podman run --name minikube-
↪nginx-haproxy --sysctl net.ipv4.ip_unprivileged_port_start=0  \
        -p 6443:6443 ${PORTS} \
        -v /home/piers/.minikube/minikube-nginx-haproxy.cfg:/usr/local/etc/haproxy/
↪haproxy.cfg \
        -d haproxy:2.4 -f /usr/local/etc/haproxy/haproxy.cfg; \
fi
MINIKUBE_IP: 192.168.49.2
Adding proxy for NodePort 80
Adding proxy for NodePort 443
1ccdc3ac5327d3cdc402f27d028def53f573b601da2f0eb77ccb4ba9b03c8f46
make[2]: Leaving directory '/home/piers/git/public/ska-telescope/sdi/ska-cicd-deploy-
↪minikube'
make[1]: Leaving directory '/home/piers/git/public/ska-telescope/sdi/ska-cicd-deploy-
↪minikube'
make[1]: Entering directory '/home/piers/git/public/ska-telescope/sdi/ska-cicd-deploy-
↪minikube'
Minikube Installed: Yes!
Helm Installed:     Yes!
DRIVER:             podman
CPUS:               2
MEM:                8192
IPADDR:             192.168.178.22
OS_NAME:            linux
MINIKUBE_IP:        192.168.49.2
HOSTNAME:           wattle
FQDN:               wattle.local.net
MOUNT_FROM:         /srv
MOUNT_TO:           /srv
PROXY_VERSION:      2.4
PROXY_CONFIG:       /home/piers/.minikube/minikube-nginx-haproxy.cfg
MINIKUBE_VERSION:   v1.23.2
KUBERNETES_VERSION: v1.22.3
HELM_VERSION:       v3.7.1
INGRESS:            http://192.168.49.2
Minikube status:
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

This will install minikube, and optionally the haproxy if you are on Ubuntu.

CHAPTER

FOUR

TEST THE DEPLOYMENT

A basic functional test can be performed with the following `make minikube-test`. It will create a PersistentVolume, Deployments, an Ingress, and a LoadBalancer Service, and test connectivity using `curl`:

```
$ make minikube-test
export CLASS=nginx; \
bash ./scripts/test-ingress.sh
Check the Kubernetes cluster:
Connecting using KUBECONFIG=/home/piers/.kube/config

Version Details:
Client Version: version.Info{Major:"1", Minor:"22", GitVersion:"v1.22.3", GitCommit:
↪"c92036820499fedefec0f847e2054d824aea6cd1", GitTreeState:"clean", BuildDate:"2021-10-
↪27T18:41:28Z", GoVersion:"go1.16.9", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"22", GitVersion:"v1.22.3", GitCommit:
↪"c92036820499fedefec0f847e2054d824aea6cd1", GitTreeState:"clean", BuildDate:"2021-10-
↪27T18:35:25Z", GoVersion:"go1.16.9", Compiler:"gc", Platform:"linux/amd64"}

List nodes:
NAME       STATUS   ROLES                  AGE     VERSION   INTERNAL-IP    EXTERNAL-IP␣
↪ OS-IMAGE             KERNEL-VERSION      CONTAINER-RUNTIME
minikube   Ready    control-plane,master   8m27s   v1.22.3   192.168.49.2   <none>      ␣
↪ Ubuntu 20.04.2 LTS   5.11.0-37-generic   docker://20.10.8

Check the Ingress connection details:
Ingress Controller LoadBalancer externalIP is: 192.168.49.2:80


Show StorageClasses:
NAME                 PROVISIONER                RECLAIMPOLICY   VOLUMEBINDINGMODE  ␣
↪ALLOWVOLUMEEXPANSION   AGE
bds1                 k8s.io/minikube-hostpath   Delete          Immediate          ␣
↪false                   5m59s
block                k8s.io/minikube-hostpath   Delete          Immediate          ␣
↪false                   5m59s
nfs                  k8s.io/minikube-hostpath   Delete          Immediate          ␣
↪false                   5m59s
nfss1                k8s.io/minikube-hostpath   Delete          Immediate          ␣
↪false                   5m59s
standard (default)   k8s.io/minikube-hostpath   Delete          Immediate          ␣
↪false                   8m19s
```

(continues on next page)

(continued from previous page)

```
Next: show StorageClass details.

Deploy the Integration test:persistentvolume/pvtest created
persistentvolumeclaim/pvc-test created
configmap/test created
service/nginx1 created
deployment.apps/nginx-deployment1 created
service/nginx2 created
deployment.apps/nginx-deployment2 created
ingress.networking.k8s.io/test created
NAME                                 READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS␣
↪IMAGES   SELECTOR
deployment.apps/nginx-deployment1   0/3     3            0           0s    nginx      ␣
↪nginx    app=nginx1
deployment.apps/nginx-deployment2   0/3     3            0           0s    nginx      ␣
↪nginx    app=nginx2

NAME                                       READY   STATUS    RESTARTS   AGE   IP      ␣
↪NODE     NOMINATED NODE   READINESS GATES
pod/nginx-deployment1-66cf976cc7-8kfj7   0/1     Pending   0          0s    <none>
↪<none>   <none>           <none>
pod/nginx-deployment1-66cf976cc7-m68wq   0/1     Pending   0          0s    <none>
↪<none>   <none>           <none>
pod/nginx-deployment1-66cf976cc7-xxtjc   0/1     Pending   0          0s    <none>
↪<none>   <none>           <none>
pod/nginx-deployment2-6c7cf4ffb7-l2p69   0/1     Pending   0          0s    <none>
↪<none>   <none>           <none>
pod/nginx-deployment2-6c7cf4ffb7-mfcvx   0/1     Pending   0          0s    <none>
↪<none>   <none>           <none>
pod/nginx-deployment2-6c7cf4ffb7-ww56p   0/1     Pending   0          0s    <none>
↪<none>   <none>           <none>

NAME             TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE   SELECTOR
service/nginx1   ClusterIP   10.103.22.149   <none>        80/TCP    0s    app=nginx1
service/nginx2   ClusterIP   10.111.187.25   <none>        80/TCP    0s    app=nginx2

NAME                              CLASS    HOSTS           ADDRESS   PORTS   AGE
ingress.networking.k8s.io/test   <none>   nginx1,nginx2             80      0s

Next: Check deployment.
Waiting for resources to deploy...
deployment.apps/nginx-deployment1 condition met
deployment.apps/nginx-deployment2 condition met
NAME                                 READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS␣
↪IMAGES   SELECTOR
deployment.apps/nginx-deployment1   3/3     3            3           39s   nginx      ␣
↪nginx    app=nginx1
deployment.apps/nginx-deployment2   3/3     3            3           39s   nginx      ␣
↪nginx    app=nginx2

NAME                                       READY   STATUS    RESTARTS   AGE   IP      ␣
```

(continues on next page)

(continued from previous page)

```
↪ NODE       NOMINATED NODE   READINESS GATES
pod/nginx-deployment1-66cf976cc7-8kfj7   1/1     Running   0          39s   172.17.0.10 ␣
↪ minikube   <none>           <none>
pod/nginx-deployment1-66cf976cc7-m68wq   1/1     Running   0          39s   172.17.0.9  ␣
↪ minikube   <none>           <none>
pod/nginx-deployment1-66cf976cc7-xxtjc   1/1     Running   0          39s   172.17.0.11 ␣
↪ minikube   <none>           <none>
pod/nginx-deployment2-6c7cf4ffb7-l2p69   1/1     Running   0          39s   172.17.0.8  ␣
↪ minikube   <none>           <none>
pod/nginx-deployment2-6c7cf4ffb7-mfcvx   1/1     Running   0          39s   172.17.0.3  ␣
↪ minikube   <none>           <none>
pod/nginx-deployment2-6c7cf4ffb7-ww56p   1/1     Running   0          39s   172.17.0.12 ␣
↪ minikube   <none>           <none>

NAME             TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE   SELECTOR
service/nginx1   ClusterIP   10.103.22.149   <none>        80/TCP    39s   app=nginx1
service/nginx2   ClusterIP   10.111.187.25   <none>        80/TCP    39s   app=nginx2

NAME                              CLASS    HOSTS           ADDRESS        PORTS   AGE
ingress.networking.k8s.io/test    <none>   nginx1,nginx2   192.168.49.2   80      39s

Next: perform write/read test.
Perform write and then read test to/from shared storage -expected date stamp: Fri Oct 29␣
↪02:59:37 AM NZDT 2021

echo "echo 'Fri Oct 29 02:59:37 AM NZDT 2021' > /usr/share/nginx/html/index.html" |␣
↪kubectl -n ${NAMESPACE} exec -i $(kubectl get pods -l app=nginx1 -o name | head -1) --␣
↪bash

service/nginx-deployment1 exposed

Test Ingress -> Deployment: nginx1
-----------------------------------------nginx1-----------------------------------------
no_proxy=192.168.49.2,localhost curl -s -H Host: nginx1 http://192.168.49.2:80/
Received: Fri Oct 29 02:59:37 AM NZDT 2021 == Fri Oct 29 02:59:37 AM NZDT 2021 - OK

Test Ingress -> Deployment: nginx2
-----------------------------------------nginx2-----------------------------------------
no_proxy=192.168.49.2,localhost curl -s -H Host: nginx2 http://192.168.49.2:80/
Received: Fri Oct 29 02:59:37 AM NZDT 2021 == Fri Oct 29 02:59:37 AM NZDT 2021 - OK


Test metallb LoadBalancer
---------------------------------nginx-deployment1---------------------------------
no_proxy=192.168.49.2,localhost curl -s -H Host: nginx1 http://192.168.49.95/
Received: Fri Oct 29 02:59:37 AM NZDT 2021 == Fri Oct 29 02:59:37 AM NZDT 2021 - OK

Cleanup resources
ingress.networking.k8s.io "test" deleted
service "nginx-deployment1" deleted
service "nginx1" deleted
deployment.apps "nginx-deployment1" deleted
```

(continues on next page)

(continued from previous page)

```
service "nginx2" deleted
deployment.apps "nginx-deployment2" deleted
persistentvolumeclaim "pvc-test" deleted
warning: deleting cluster-scoped resources, not scoped to the provided namespace
persistentvolume "pvtest" deleted
configmap "test" deleted
```

This will deploy some NGINX webservers, and then test accessing them through the Ingress Controller.

CHAPTER

FIVE

PRELOAD IMAGES FROM A HELM CHART

In order to speed up your deployments in Minikube, it is possible to preload the images. The following shows two ways of loading images from Helm Charts.

5.1 From a Central Artefact Repository Chart

Run the following `make` target to introspect a Helm Chart from the SKAO Central Artefact Repository, and load the images therein:

```
$ make minikube-load-images K8S_CHARTS=ska-tango-base
export K8S_CHART_PARAMS=; \
bash ./scripts/load-images.sh "ska-tango-base"
Searching in charts: ska-tango-base
Checking chart: ska-tango-base
wrote /tmp/chart-ska-tango-base.oMYhEAaCxJ/ska-tango-base/templates/deviceservers.yaml
wrote /tmp/chart-ska-tango-base.oMYhEAaCxJ/ska-tango-base/templates/deviceservers.yaml
wrote /tmp/chart-ska-tango-base.oMYhEAaCxJ/ska-tango-base/templates/deviceservers.yaml
wrote /tmp/chart-ska-tango-base.oMYhEAaCxJ/ska-tango-base/templates/deviceservers.yaml
wrote /tmp/chart-ska-tango-base.oMYhEAaCxJ/ska-tango-base/templates/databaseds.yaml
wrote /tmp/chart-ska-tango-base.oMYhEAaCxJ/ska-tango-base/templates/deviceservers.yaml
wrote /tmp/chart-ska-tango-base.oMYhEAaCxJ/ska-tango-base/templates/tangodb.yaml
wrote /tmp/chart-ska-tango-base.oMYhEAaCxJ/ska-tango-base/templates/databaseds.yaml
wrote /tmp/chart-ska-tango-base.oMYhEAaCxJ/ska-tango-base/templates/deviceservers.yaml
wrote /tmp/chart-ska-tango-base.oMYhEAaCxJ/ska-tango-base/templates/tangodb.yaml
wrote /tmp/chart-ska-tango-base.oMYhEAaCxJ/ska-tango-base/templates/deviceservers.yaml
wrote /tmp/chart-ska-tango-base.oMYhEAaCxJ/ska-tango-base/templates/ingress.yaml
wrote /tmp/chart-ska-tango-base.oMYhEAaCxJ/ska-tango-base/templates/deviceservers.yaml
wrote /tmp/chart-ska-tango-base.oMYhEAaCxJ/ska-tango-base/templates/deviceservers.yaml

looking for images in: /tmp/chart-ska-tango-base.oMYhEAaCxJ/ska-tango-base/templates/
↪databaseds.yaml
looking for images in: /tmp/chart-ska-tango-base.oMYhEAaCxJ/ska-tango-base/templates/
↪deviceservers.yaml
looking for images in: /tmp/chart-ska-tango-base.oMYhEAaCxJ/ska-tango-base/templates/
↪ingress.yaml
looking for images in: /tmp/chart-ska-tango-base.oMYhEAaCxJ/ska-tango-base/templates/
↪tangodb.yaml
Combined list of charts to load:
artefact.skao.int/ska-tango-images-tango-cpp:9.3.9
artefact.skao.int/ska-tango-images-tango-db:10.4.14
```

(continues on next page)

(continued from previous page)

```
artefact.skao.int/ska-tango-images-tango-dsconfig:1.5.3
artefact.skao.int/ska-tango-images-tango-java:9.3.6

Finished searching charts, now loading images
loading image: artefact.skao.int/ska-tango-images-tango-cpp:9.3.9
loading image: artefact.skao.int/ska-tango-images-tango-db:10.4.14
loading image: artefact.skao.int/ska-tango-images-tango-dsconfig:1.5.3
loading image: artefact.skao.int/ska-tango-images-tango-java:9.3.6

List of loaded images:
k8s.gcr.io/pause:3.5
k8s.gcr.io/metrics-server/metrics-server:<none>
k8s.gcr.io/kube-scheduler:v1.22.3
k8s.gcr.io/kube-proxy:v1.22.3
k8s.gcr.io/kube-controller-manager:v1.22.3
k8s.gcr.io/kube-apiserver:v1.22.3
k8s.gcr.io/ingress-nginx/kube-webhook-certgen:<none>
k8s.gcr.io/ingress-nginx/controller:<none>
k8s.gcr.io/etcd:3.5.0-0
k8s.gcr.io/coredns/coredns:v1.8.4
gcr.io/k8s-minikube/storage-provisioner:v5
docker.io/metallb/speaker:<none>
docker.io/metallb/controller:<none>
docker.io/library/busybox:1.28.3
docker.io/kubernetesui/metrics-scraper:v1.0.7
docker.io/kubernetesui/dashboard:v2.3.1
docker.io/ivans3/minikube-log-viewer:<none>
artefact.skao.int/ska-tango-images-tango-rest:1.14.6
artefact.skao.int/ska-tango-images-tango-java:9.3.6
artefact.skao.int/ska-tango-images-tango-java:9.3.5
artefact.skao.int/ska-tango-images-tango-dsconfig:1.5.3
artefact.skao.int/ska-tango-images-tango-dsconfig:1.5.2
artefact.skao.int/ska-tango-images-tango-dsconfig:1.5.1
artefact.skao.int/ska-tango-images-tango-db:10.4.14
artefact.skao.int/ska-tango-images-tango-db:10.4.13
artefact.skao.int/ska-tango-images-tango-cpp:9.3.9
artefact.skao.int/ska-tango-images-tango-cpp:9.3.7
artefact.skao.int/ska-tango-examples:0.4.15
artefact.skao.int/ska-ser-skallop:2.7.4

To force reload an image use: minikube image load <image name>:<tag>
```

5.2 From a local directory

The same `make` target can be supplied with a local Helm Chart directory as the soruce and will attempt the same introspection to discover images to load:

```
$ make minikube-load-images K8S_CHARTS=../../ska-skampi/charts/ska-mid/
```

Because this process uses the `helm template` command to generate the templates to parse, it maybe necessary to pass in `values` in order to get the charts to render correctly. Additional command line arguments can be passed in using the `K8S_CHART_PARAMS` make variable.

CHAPTER

SIX

TESTING WITH SKAMPI

An example of doing a complete deploy, test, destroy cycle for Skampi is:

```
# optionally pre-load images
$ make minikube-load-images K8S_CHARTS=../../ska-skampi/charts/ska-mid/
# switch to ska-skampi project
$ cd ../../ska-skampi
$ make k8s-install-chart K8S_CHART=ska-mid KUBE_NAMESPACE=default
$ make k8s-wait KUBE_NAMESPACE=default K8S_TIMEOUT=600s
$ make k8s-test KUBE_NAMESPACE=default K8S_TIMEOUT=600s
$ make k8s-uninstall-chart KUBE_NAMESPACE=default
```

CHAPTER

SEVEN

CLEAN UP AGAIN

Tear everything down again with:

```
$ make minikube-clean
  Stopping node "minikube"  ...
  Powering off "minikube" via SSH ...
  Stopping node "minikube"  ...
  Powering off "minikube" via SSH ...
  Stopping node "minikube"  ...
  Powering off "minikube" via SSH ...
  1 nodes stopped.
  Deleting "minikube" in podman ...
  Deleting container "minikube" ...
  Removing /home/piers/.minikube/machines/minikube ...
  Removed all traces of the "minikube" cluster.
```

CHAPTER

EIGHT

NOTES

- `make minikube-haproxy` adds in a proxy for all NodePort address in the cluster - if you add a new service that requires exposing, then just rerun `make minikube-haproxy`. This only works on Ubuntu as it requires `podman`.
- `http_proxy`, `https_proxy`, and `no_proxy` will be passed through to `minikube start` if set, on `make minikube-install`.
- Configuration of additional `--apiserver-names` and `--apiserver-ips` are passed through to `minikube start`, on `make minikube-install` so that Kubernetes APIServer can be addressed remotely through the HAProxy.
- `MOUNT_FROM` and `MOUNT_TO` are passed to `minikube start`, on `make minikube-install` for `--mount-string` so that a user supplied directory is mounted into the Kubernetes cluster and available for mounting in Pods.

CHAPTER

NINE

USING A PROXY

When you must use a proxy to reach the outside world then the configuration must be set in two places:

- Configure the `dockerd` to use the proxy - https://docs.docker.com/config/daemon/systemd/#httphttps-proxy
- Set the environment variables in your shell - typically set the following values in `~/.bashrc` :

```
export HTTP_PROXY=http://my.proxy.server:8888
export HTTPS_PROXY=http://my.proxy.server:8888
export NO_PROXY="202.9.15.208,localhost,127.0.0.1,10.96.0.0/12,192.168.99.0/24,192.168.
↪39.0/24,172.17.0.1/16"

export http_proxy=http://my.proxy.server:8888
export https_proxy=http://my.proxy.server:8888
export no_proxy="202.9.15.208,localhost,127.0.0.1,10.96.0.0/12,192.168.99.0/24,192.168.
↪39.0/24,172.17.0.1/16"
```

`podman` will honour the above environment variables.

Adjust the above settings according to your environment. Preserve the Kubernetes and Docker specific values for `10.96.0.0/12` and `172.17.0.1/16`. Check here for updates https://minikube.sigs.k8s.io/docs/handbook/vpn_and_proxy/ .

CHAPTER

TEN

USING A DOCKER IMAGE PROXY/CACHE

:zap: only on Ubuntu.

If you are not already using a (possibly corporate) proxy, then it is possible to setup your own personal pull-through Docker Image Proxy for Minikube using https://github.com/rpardini/docker-registry-proxy . This will create a local cache of images that are pulled so that the second time you make a deployment in Minikube, the cache will respond without going to the upstream image registry. This is currently configured to cache:

- docker.io
- gcr.io
- k8s.gcr.io
- quay.io
- registry.gitlab.com
- docker.elastic.co

This will help work around pull throttling introduced by Docker Hub (https://docs.docker.com/docker-hub/download-rate-limit/), but will also speed up your deployments, as the cache can be maintained between re/installs of Minikube.

Deploy the cache using:

```
$ make all USE_CACHE=yes
```

Running `make minikube-clean` (when you teardown Minikube - alias `clean`) will not clean out the cache. If you want to remove the cached images altogether then you must uninstall Minikube and the clean the cache with:

```
$ make minikube-clean
$ make minikube-clean-cache
```

CHAPTER

ELEVEN

USING NGINX

NGiNX is now deployed by default following the removal of Traefik as an ingress controller from the SKA cluster.

CHAPTER

TWELVE

CAVEATS

12.1 Encryption

If your base OS is using an Encrypted Filesystem - by either checking the full disk encryption option on Ubuntu Installation, or by using a Home Folder encryption on one of its flavours - this repository will *not* work for you.

The filesystem `overlayfs` can't work on top of `encryptfs`, will fail to start the `minikube` VM *and potentially can lead to data loss on your encrypted filesystem*.

CHAPTER

THIRTEEN

OS VARIATIONS

13.1 Ubuntu

:zap: Make sure that you have passwordless `sudo` setup - see https://serverfault.com/questions/160581/how-to-setup-passwordless-sudo-on-linux.

`podman` is now used instead of Docker, to deploy the haproxy, cache, and registry. Please install `podman` with:

```
$ make minikube-install-podman
```

For podman to work correctly with minikube, you will need to enable passwordless sudo. An example of this is to add the following config:

```
export ACCOUNT=<your account name here>
echo "${ACCOUNT} ALL=(ALL) NOPASSWD: ALL" > /etc/sudoers.d/${ACCOUNT}
chmod 440 /etc/sudoers.d/${ACCOUNT}
```

If you are using `18.04`, then you may see an error with `podman` similar to:

```
(base) button@collar-k8s:~/ska-cicd-deploy-minikube$ make all
make[1]: Entering directory '/home/button/ska-cicd-deploy-minikube'
Minikube status:
  Profile "minikube" not found. Run "minikube profile list" to view all profiles.
  To start a cluster, run: "minikube start"
Minikube not running, continuing...
Using driver: podman
Extra ARGS set:
Local mount: /srv:/srv
  minikube v1.23.2 on Ubuntu 18.04 (amd64)
  Using the podman driver based on user configuration

  Exiting due to PROVIDER_PODMAN_NOT_RUNNING: "sudo -k -n podman version --format " exit␣
↪status 125: Error: failed to mount overlay for metacopy check with "nodev,metacopy=on"␣
↪options: invalid argument
  Documentation: https://minikube.sigs.k8s.io/docs/drivers/podman/

Makefile:338: recipe for target 'minikube' failed
make[1]: *** [minikube] Error 63
make[1]: Leaving directory '/home/button/ska-cicd-deploy-minikube'
Makefile:321: recipe for target 'install' failed
make: *** [install] Error 2
```

This is likely to do with a support issue, and can be resolved by removing the reference to `metacopy=...` in the `/etc/containers/storage.conf` configuration file for `podman`.

Additionally, when you try and create a Minikube cluster with `make all`, you may see a message like:

```
Adding proxy for NodePort 80
Adding proxy for NodePort 443
? Please select an image:
   docker.io/library/haproxy:2.4
    quay.io/haproxy:2.4
```

In this case, you must choose the `docker.io/library/haproxy:2.4` option, as this is an error in the podman image search path.

If you live in a super controlled environment where you are unable to allocate `subuids` and `subgids`, you may see errors like:

```
Error: writing blob: adding layer with blob
↪"sha256:7692efc5f81cadc73ca1afde08b1a5ea126749fd7520537ceea1a9871329efde": Error␣
↪processing tar file(exit status 1): potentially insufficient UIDs or GIDs available in␣
↪user namespace (requested 0:12 for /var/spool/mail): Check /etc/subuid and /etc/
↪subgid: lchown /var/spool/mail: invalid argument
```

In this case, you will need to use `sudo` on every `podman` invocation. This can be achieved by doing something like renaming `podman` and replacing it with a script that first `sudo`s, eg:

```
echo -e '#!/bin/sh\nsudo /usr/bin/podman $*' | sudo tee /usr/local/bin/podman
sudo chmod a+x /usr/local/bin/podman
export PATH=/usr/local/bin:${PATH}
```

Add the `export` line to your `~/.bashrc`, so that it is permanent for you.

In later versions of Ubuntu, `resolvconf` is installed via a separate package. It is important to ensure that this is installed so that the name resolution (DNS) for metallb addresses is configured correctly. Use:

```
$ sudo apt install resolvconf
```

If this is not installed then, depending on how your DNS is configured, name resolution may not function correctly for names originating inside Minikube Kubernetes.

13.2 macOS

:zap: If you are using macOS M1 arm64, then the last known working combination is Docker Desktop 4.0.1 with Kubernetes 1.21.4 on Big Sur 11.6.1 eg: `make all DRIVER=docker KUBERNETES_VERSION=v1.21.4`.

- The `hyperkit` driver is used by default. This is auto-selected by querying `OS_NAME` in the `Makefile`.
- macOS does not have `envsubst` by default. This can be installed by:

```
brew install gettext
```

- macOS may have an old version of Gnu Make installed. At least version 3.82 is required. If you get a `*** missing separator` error, a newer version can be installed as below. **Note:** it will be installed as `gmake`, so add a symlink, and ensure that `/usr/local/bin` is ahead of `/usr/bin` in your PATH:

```
brew install make
ln -s /usr/local/bin/gmake /usr/local/bin/make
make --version
```

13.3 WSL2

Before installing minikube using this repository you will need to

- install wsl2 with ubuntu 20.04
- install docker-ce in wls2 (see instructions here
- run systemctl faker script or add into the local file `.profile` the command: `wsl.exe -u root -e sh -c "service docker status || service docker start`
- windows terminal preview is also highly recommended!

Steps:

```
# install windows docker-ce
# check docker is available under ubuntu
docker version
# get systemd enabler
git clone https://github.com/alfink/ubuntu-wsl2-systemd-script
cd ubuntu-wsl2-systemd-script
bash ubuntu-wsl2-systemd-script.sh
# or change the .profile file
echo "wsl.exe -u root -e sh -c "service docker status || service docker start" >> .
↪profile
# start new wsl2 terminal - you should see a message about systemctl
# clone this repo and install minikube
git clone https://gitlab.com/ska-telescope/sdi/cka-cicd-deploy-minikube.git
# Follow the above steps as a normal Ubuntu installation, configure the variables as you␣
↪like
# ### IMPORTANT use docker driver not non as states in "official" blog post
```

In order to access the ingress on minikube, please note that the docker internal ip address is not reacheble from windows host. Instead of using the minikube ip, please use the WSL2 ip address which can be found using the following command:

```
hostname -I | awk '{print $1}'
```

13.4 For ska-cicd-deploy-minikube Developers

Currently, it is not possible to run the full test suite for `ska-cicd-deploy-minikube` in the pipeline because it requires deployment within a VM (not container in container). This means that developers should run the `make bats-test` test suite locally before pushing commits to ensure the wider functionality is tested.

A pre-push git hook that will remind you to do the testing can be activated with:

```
$ git config --local core.hooksPath  resources/git-hooks/
```

Todo:

- Insert todo's here

CHAPTER

FOURTEEN

PACKAGE-NAME DOCUMENTATION

This section describes requirements and guidelines.

14.1 Subtitle

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

14.1.1 Public API Documentation

Functions

Classes