project-name Documentation

Release 0.1.0

author

Feb 22, 2023

CONTENTS:

This project defines a kubernetes controller for TANGO device servers.

CHAPTER

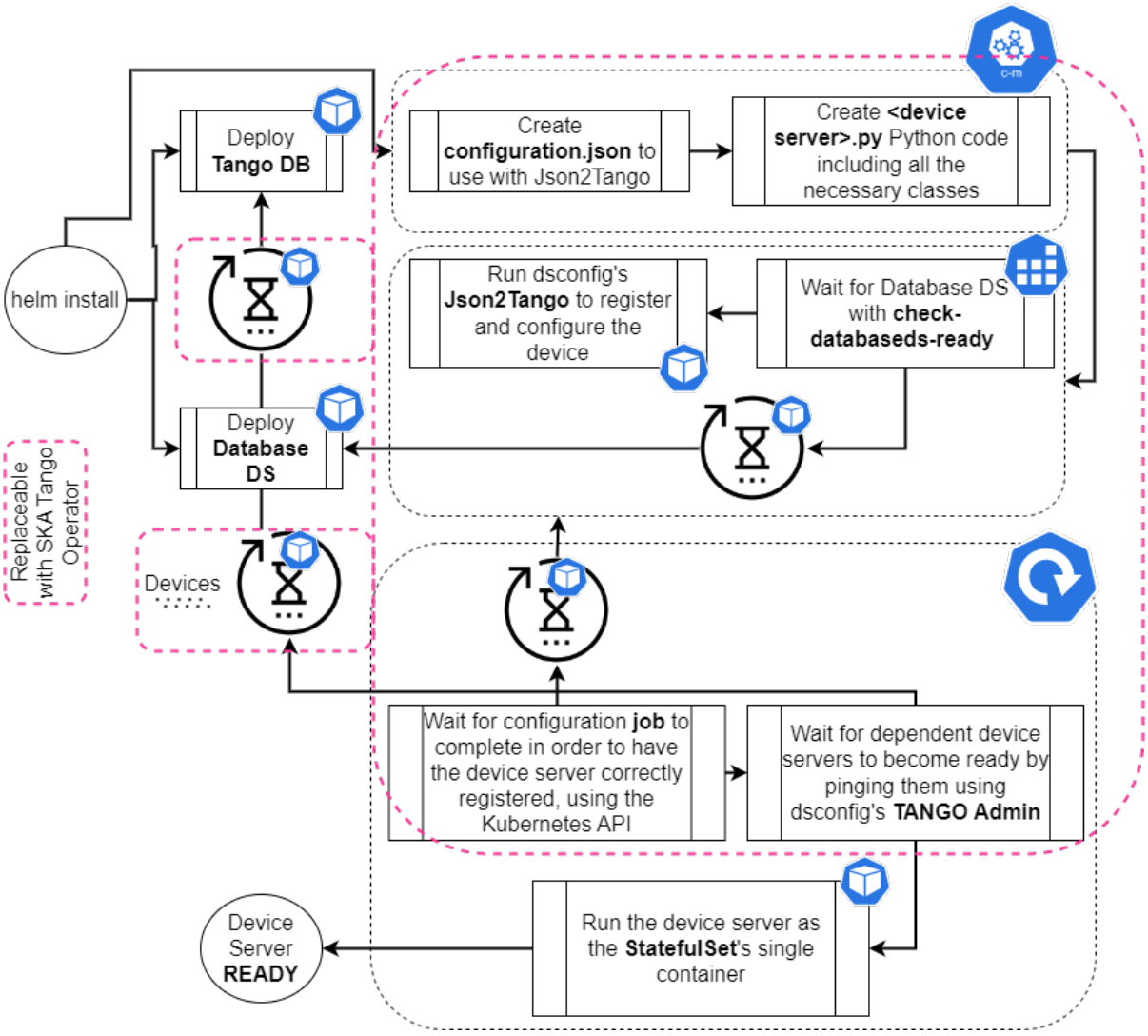
ONE

DEVICE SERVER DEPLOYMENT

The SKA telescope software is a conteinerized application that run with kubernetes (k8s). A TANGO device server can be seen as a set of k8s resources, as a service, pods, etc. deployed with the help of Helm. By using the ska-tango-util chart, a device server is composed by:

- a job for the initialization of the entry in the tangodb,
- a service,
- a statefulset with one init container per dependency,
- a role, rolebinding and a service account for waiting for the job to be finish in an init container.

The following image shows the deployment flow with the use of the ska-tango-util (in any version < 0.4.0):



Clearly this approach has some disadvantages in case of problems like software exception, bugs or wrong configuration. In all those cases, extra resources are required from the Kubernetes cluster - as it requires multiple PODs to be created as init-containers and jobs. It also leaves behind spent resources (i.e. job pods that have completed). It can take a lot longer for a Device Server to startup - because of the Crash Loop Backoff behaviour that exists in the Kubernetes cluster, the greater the POD completions without success, the longer it takes to restart - an effect that can be compounded with multiple device dependencies.

CHAPTER

TWO

EXTENDING KUBERNETES

There are many possibilities for extending kubernetes. In specific the following list shows the current extension points:

- Kubectl plugins, official client libraries - Keystone
- API Server extension - ACL, edit requests - Keystone
- Custom Resources Definitions - partner with Custom Controllers
- Custom schedulers - rare
- Custom Controllers - API aggregation, pick up custom resources - KubeDB
- Network extensions - Calico, Kuryr
- Storage plugins - Cinder storage class, and operator

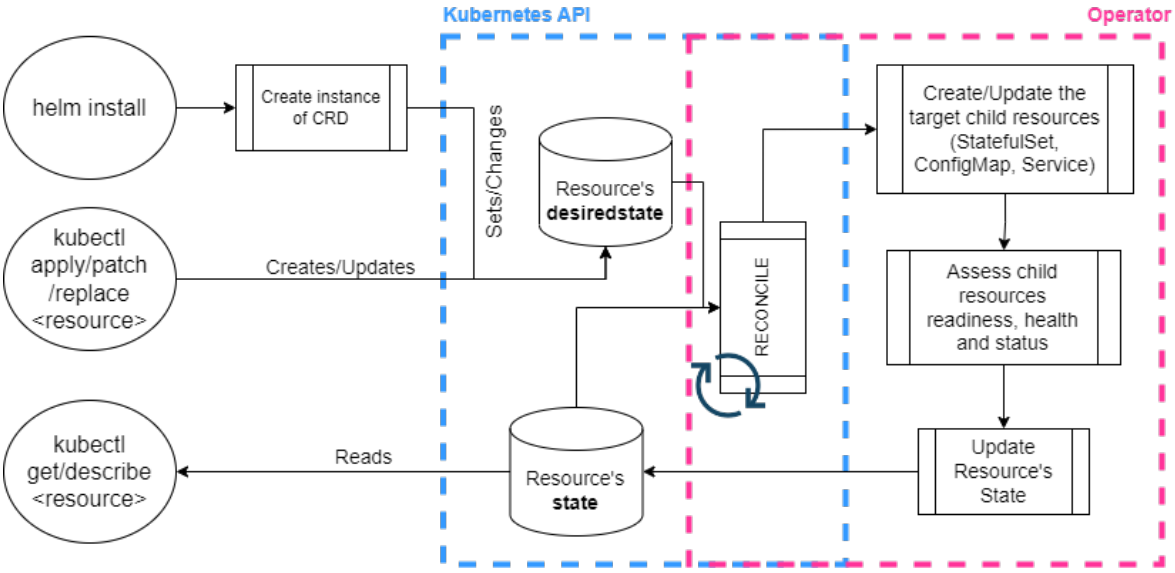
2.1 The Operator pattern

The operator pattern aims to capture the key aim of a human operator who is managing a service or set of services. Human operators who look after specific applications and services have deep knowledge of how the system ought to behave, how to deploy it, and how to react if there are problems (from k8s docs - Operator pattern). In specific:

- Extends the Control Plane to give Custom Behaviours
- Use Custom Resource Definitions (basically extend the API)
- Use the control loop pattern (in automation, a control loop is a non-terminating loop that regulates the state of a system)

The ska-tango-operator is a kubernetes operator capable of managing TANGO resources (DeviceServer and DatabaseDS) that is to control their lifecycle within the Kubernetes' native control/event loop. The goal is to have a cleaner deployment (no init-containers and jobs to perform configuration and dependency-checking operations), as well as an optimised startup time for Device Servers, as the operator can directly tap into the TANGO environment and retrieve information on dependent devices and the TANGO Host itself.

Developers know Device Servers, not StatefulSet resources, as those are components with specific behaviors relevant to the platform in use. Essentially the ska-tango-operator is an extension of the Kubernetes API with the perception of TANGO to Kubernetes mapping, automating much of the tasks a human would do to operate a TANGO resource, running on Kubernetes.

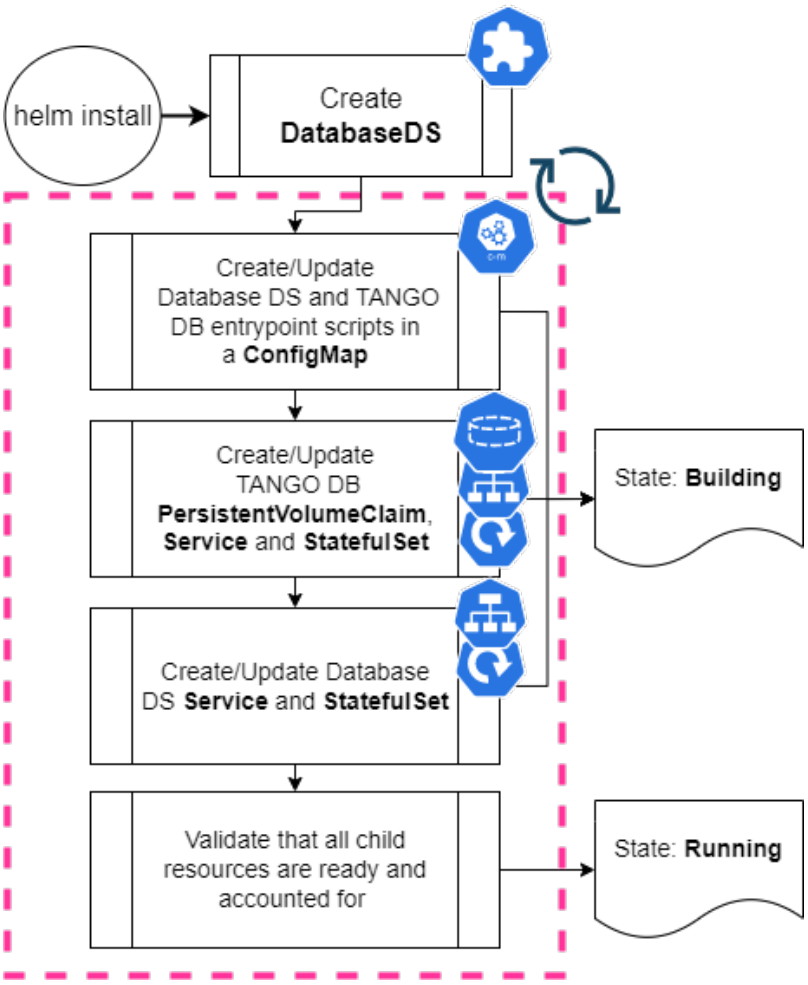


2.2 Custom Resource Definition (CRD): databaseds.tango.tango-controls.org

The command `kubectl describe crd databaseds.tango.tango-controls.org` shows the list of options for this resource definition. In specific by creating this resource the following resources will be created:

- TANGO DB StatefulSet, Service and PersistentVolumeClaim
- Database DS StatefulSet and Service
- Database DS/TANGO DB ConfigMap
- Script 'start-databaseds-tangodb.sh' used as entrypoint for TANGO Database
- Script 'start-databaseds.sh' used as docker entrypoint for Database DS
- File 'config.json' Database DS json2tango configuration

The databaseds has 2 states: *Building* and *running*.

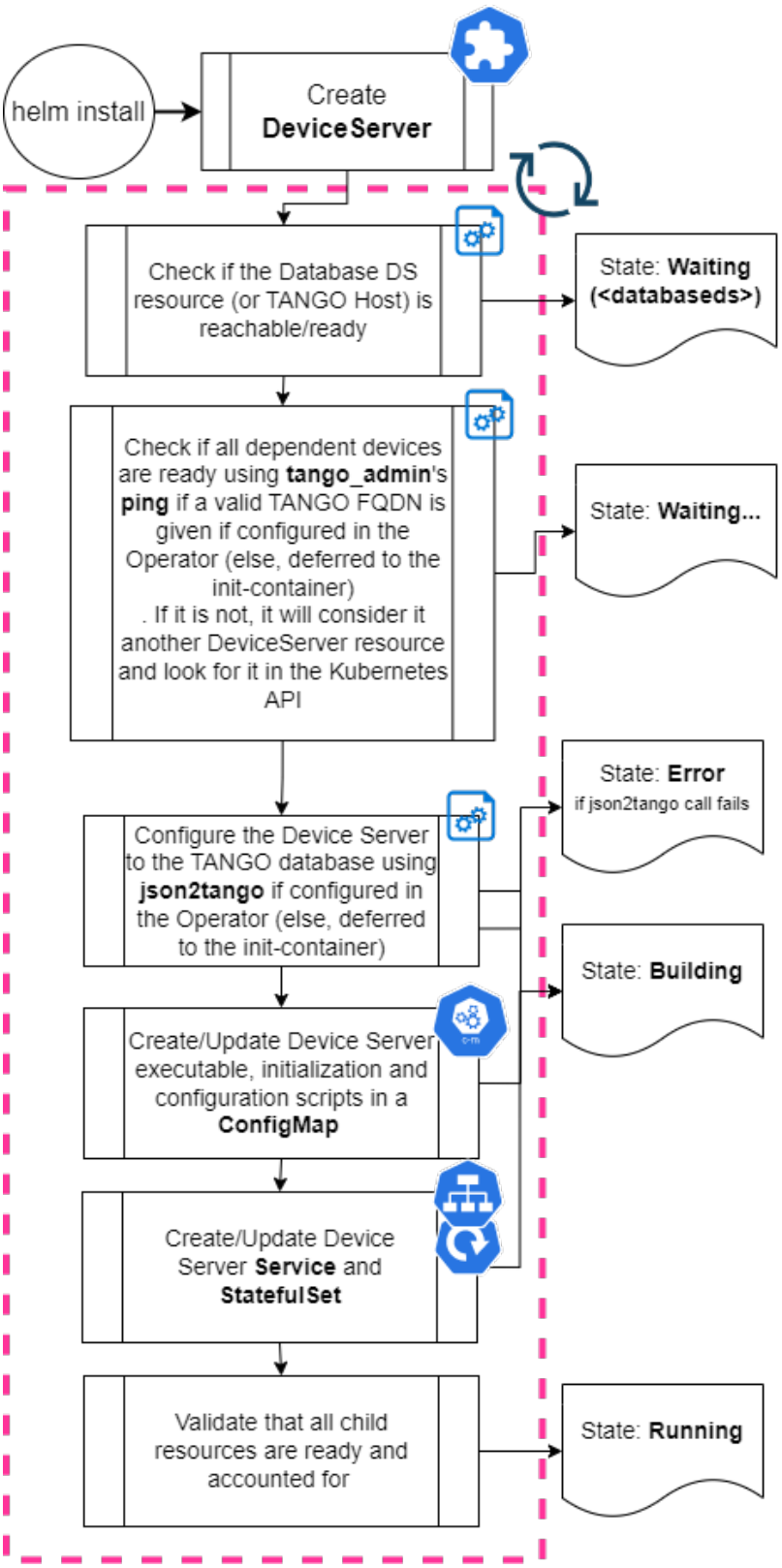


2.3 Custom Resource Definition (CRD): deviceservers.tango.tango-controls.org

The command `kubectl describe crd deviceservers.tango.tango-controls.org` shows the list of options for this resource definition. In specific by creating this resource the following resources will be created:

- Device Server StatefulSet and Service
- Device Server ConfigMap
- Device Server script used to run the device (command called within start-deviceserver.sh)

The possible states for a device server are: *Building*, *Waiting*, *Error*, *Pending*, *Running*.


helm install
Create
DeviceServer
Check if the Database DS resource (or TANGO Host) is reachable/ready
State: Waiting (<databaseds>)
Check if all dependent devices are ready using tango_admin's ping if a valid TANGO FQDN is given if configured in the Operator (else, deferred to the init-container)
. If it is not, it will consider it another DeviceServer resource and look for it in the Kubernetes API
State: Waiting...
State: Error
if json2tango call fails
Configure the Device Server to the TANGO database using json2tango if configured in the Operator (else, deferred to the init-container)
State: Building
Create/Update Device Server executable, initialization and configuration scripts in a ConfigMap
Create/Update Device Server Service and StatefulSet
Validate that all child resources are ready and accounted for
State: Running

2.4 TANGO Operator flow

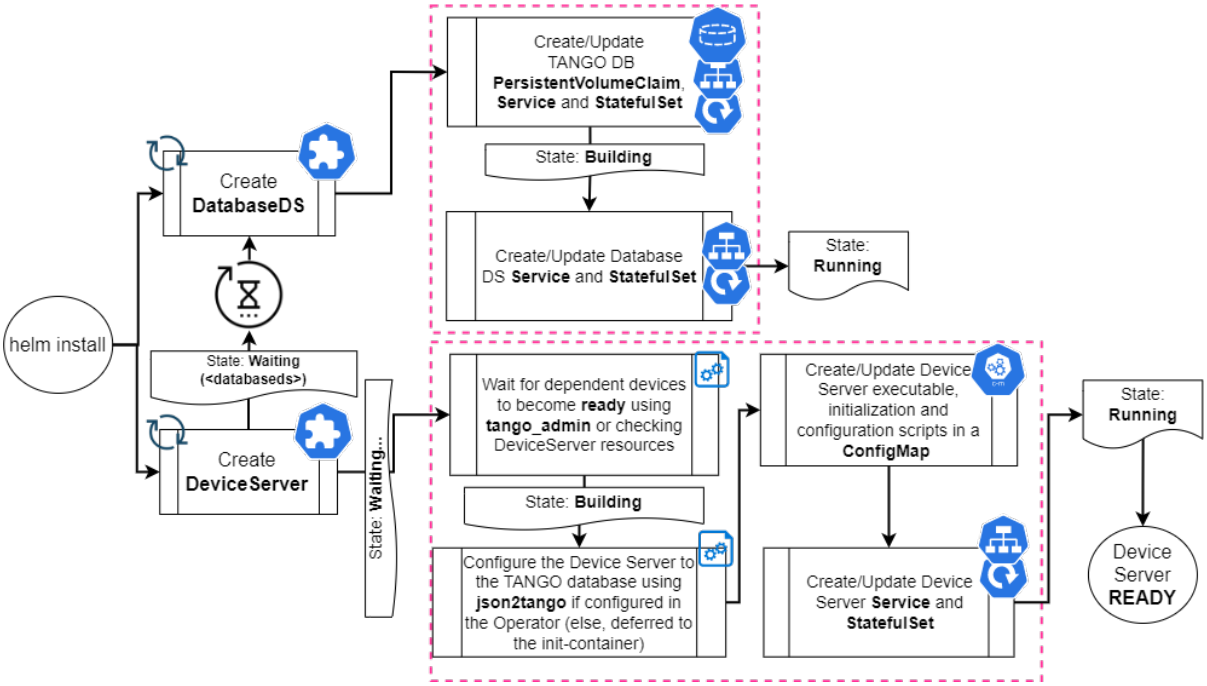
The ska-tango-base and ska-tango-util charts have been refactored in order to generate deviceserver and databaseds CRD instead of usual k8s resources depending on the parameter `global.operator` (true for deviceserver and databaseds generation). The charts are completely retro-compatible.

The following code how the system behaves in the above examples using the ska-tango-operator controller:

```
make k8s-uninstall-chart

helm repo list | grep artefact.skao.int || helm repo add k8s-helm-repository https://
↪artefact.skao.int/repository/helm-internal

helm install to k8s-helm-repository/ska-tango-operator --create-namespace --namespace␣
↪ska-tango-operator-system

make k8s-install-chart SKA_TANGO_OPERATOR=true K8S_EXTRA_PARAMS="--values my_values.yaml"
make k8s-watch SKA_TANGO_OPERATOR=true
```

The following code shows how to get some information from the deployment using the operator.

```
kubectl describe crd databaseds.tango.tango-controls.org
kubectl describe crd deviceservers.tango.tango-controls.org
kubectl get databaseds --all-namespaces
kubectl describe databaseds.tango.tango-controls.org -n ska-tango-examples
kubectl get deviceservers.tango.tango-controls.org -n ska-tango-examples
kubectl describe deviceservers.tango.tango-controls.org -n ska-tango-examples

make k8s-template-chart # will produce the file manifests.yaml
```



2.5 Metrics and grafana dashboard

When the ska-tango-operator is installed and an application is deployed in the k8s cluster, a set of metrics are available from the controller. The cluster has an ingress for those metrics available at `/<namespace where the operator is installed>/metrics`.

Every day there is a pipeline execution for the ska-tango-examples repository. So a live example of the dashboard can be found here (please select the namespace that start with `ci-ska-tango-examples-*`).

2.6 Confluence pages

There is a confluence page that describes the ska-tango-operator in great details here. A workshop has been done with this topic and the recording is available here.

CHAPTER

THREE

SKA-TANGO-OPERATOR

This project is an extension of k8s with a Custom Resource Definition (CRD i.e. Device Server and Databaseds) and a Controller to give custom behaviours in order to have a better usage of TANGO-controls in kubernetes.

3.1 Installation

This project is structured to use k8s for development and testing so that the build environment, test environment and test results are all completely reproducible and are independent of host environment. It uses `make` to provide a consistent UI (run `make help` for targets documentation).

3.1.1 Install docker

Follow the instructions available at here.

3.1.2 Install minikube

You will need to install `minikube` or equivalent k8s installation in order to set up your test environment. You can follow the instruction at here:

```
git clone git@gitlab.com:ska-telescope/sdi/deploy-minikube.git
cd deploy-minikube
make all
eval $(minikube docker-env)
```

Please note that the command `eval $(minikube docker-env)` *will point your local docker client at the docker-in-docker for minikube. Use this only for building the docker image and another shell for other work.*

3.1.3 Install GO

Follow the instructions available at here.

3.2 How to Use

Install the operator with the following command:

```
$ git clone git@gitlab.com:ska-telescope/ska-tango-operator.git
$ cd ska-tango-operator
$ make k8s-install-chart
```

Install the example with the following command:

```
$ kubectl apply -f config/samples/all-in-one.yml
```

Check what's happening with the following commands:

```
$ kubectl get deviceserver
$ kubectl get databaseds
```

Uninstall the examples and the operator with the following commands:

```
$ kubectl delete -f config/samples/all-in-one.yml
$ make k8s-uninstall-chart
```

3.2.1 Generate the charts

The charts available in this repository are generated with the following commands:

```
$ make helm-operator
$ make helm-crd
```

3.3 External references and other information

- This project has been made with the help of an operator SDK called kubebuilder written in GO.
- The databaseds is a Device server as any other and technically can be defined as device server.