neueda
futureproof your workforce

TEMPLATES AND OPERATORS

NEUEDA.COM

Objectives

- OpenShift Templates
- Templates in the Web Console and the CLI
- Template File Structure
- Kubernetes Operators
- Operators Structure and Operation

neueda
futureproof your workforce

Templates and Operators

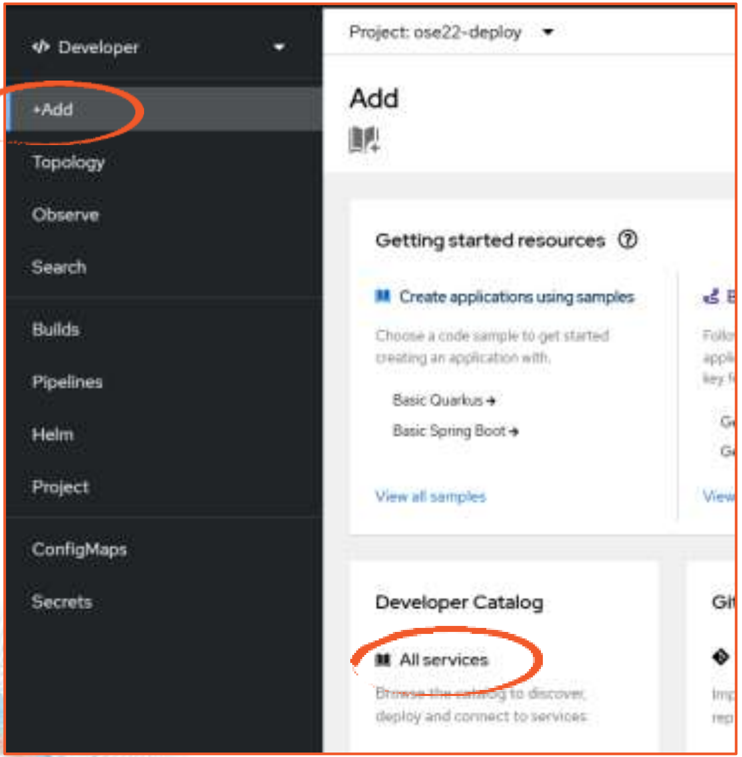
- **Templates** and **Operators** automate repetitive tasks in OpenShift
  - Triggering builds and deployments when source code changes
  - Restarting failed Pods
  - An Operator upgrading your database server
- A Template automates the <u>creation</u> of a set of resources.
- An Operator <u>also</u> deploys an application and its resources then continues to watch and govern those resources.

TEMPLATES

Templates

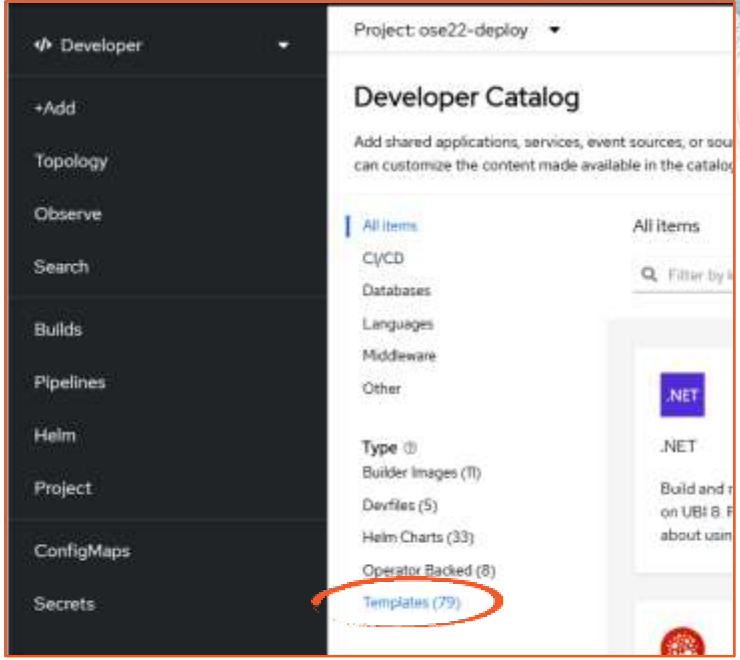
- A template is a list of objects and the names parameters of their configuration
- To view the available templates in the Web Console:
  - Click on the “+Add” item in the Developer Perspective
  - Click on “All services” in the Developer Catalog
  - The click on “Template” in the **Type** submenu in the left hand menu.

Templates in the Web Console



Choose a code sample to get started creating an application with.

Browse the catalog to discover, deploy and connect to services

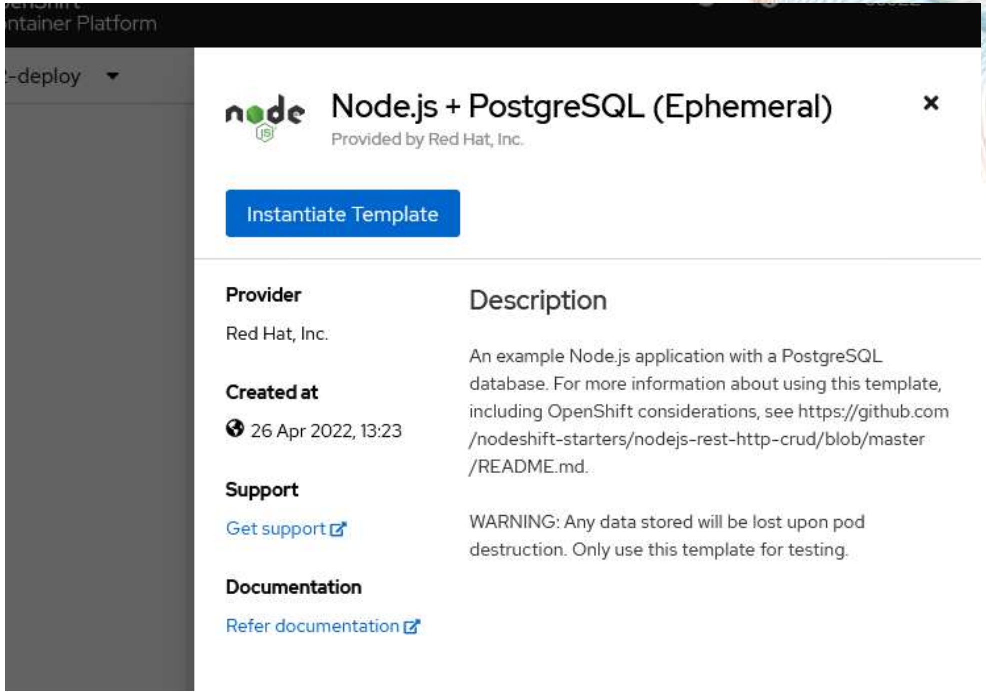


Add shared applications, services, event sources, or sou
can customize the content made available in the catalog

Build and
on UBI 8.
about usin

Instantiating a Template in the Web Console

- When a template is selected, a description is shown and you are invited to instantiate it



Node.js + PostgreSQL (Ephemeral)
Provided by Red Hat, Inc.

Provider
Red Hat, Inc.

Created at
26 Apr 2022, 13:23

Support
Get support

Documentation
Refer documentation

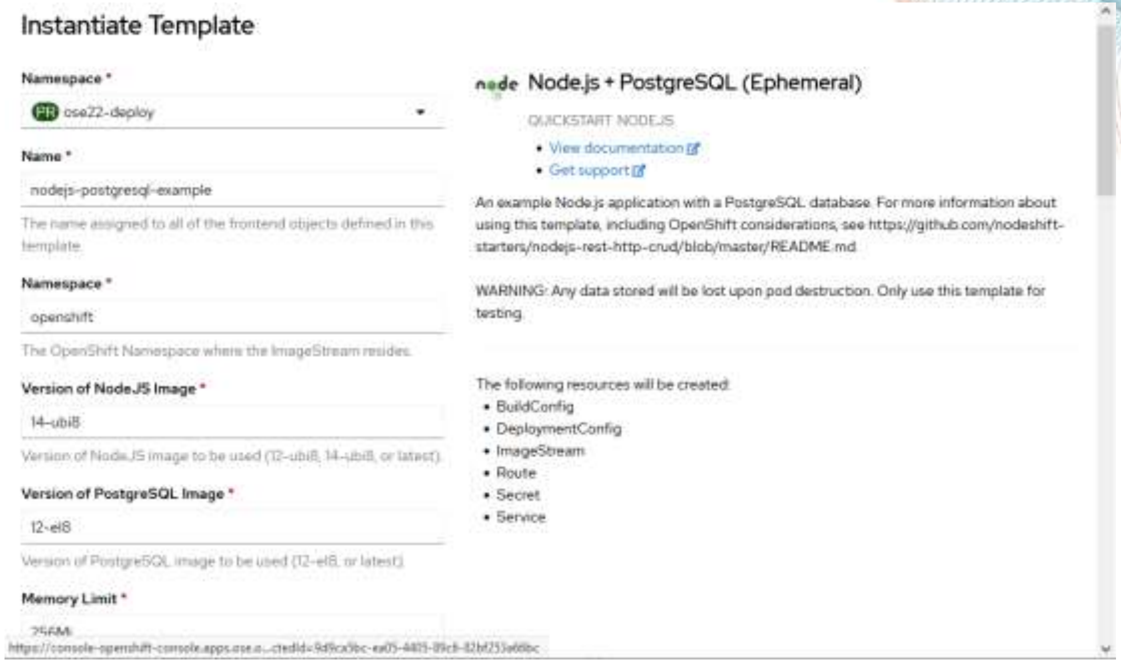
Description

An example Node.js application with a PostgreSQL database. For more information about using this template, including OpenShift considerations, see https://github.com/nodeshift-starters/nodejs-rest-http-crud/blob/master/README.md.

WARNING: Any data stored will be lost upon pod destruction. Only use this template for testing.

Instantiating a Template in the Web Console

- When you click on “Instantiate Template” you are given a list of configurable parameters:



Instantiate Template

Namespace *
PR ose22-deploy

Name *
nodejs-postgresql-example
The name assigned to all of the frontend objects defined in this template.

Namespace *
openshift
The OpenShift Namespace where the ImageStream resides.

Version of NodeJS Image *
14-ubi8
Version of NodeJS image to be used (12-ubi8, 14-ubi8, or latest).

Version of PostgreSQL Image *
12-el8
Version of PostgreSQL image to be used (12-el8, or latest).

Memory Limit *
256Mi

node Node.js + PostgreSQL (Ephemeral)

QUICKSTART NODEJS

- View documentation
- Get support

An example Node.js application with a PostgreSQL database. For more information about using this template, including OpenShift considerations, see https://github.com/nodeshift-starters/nodejs-rest-http-crud/blob/master/README.md.

WARNING: Any data stored will be lost upon pod destruction. Only use this template for testing.

The following resources will be created:
- BuildConfig
- DeploymentConfig
- ImageStream
- Route
- Secret
- Service

https://console-openshift-console.apps.ose.o...cteId=9d9ca9bc-ea05-4405-89c8-82bf253a66bc

Templates in the oc CLI

- You can also access, edit and deploy templates from the command line.
- OpenShift Templates included with a particular cluster install are in the `openshift` namespace. To see the list:

```
$ oc get templates -n openshift

NAME                     DESCRIPTION                                                                        PARAMETERS        OBJECTS
3scale-gateway           3scale's APIcast is an NGINX based API gateway used to integrate your interna...   17 (8 blank)      3
amq63-basic              Application template for JBoss A-MQ brokers. These can be deployed as standal...   11 (4 blank)      6
amq63-persistent         An example JBoss A-MQ application. For more information about using this temp...   13 (4 blank)      8
amq63-persistent-ssl     An example JBoss A-MQ application. For more information about using this temp...   18 (6 blank)      12
amq63-ssl                An example JBoss A-MQ application. For more information about using this temp...   16 (6 blank)      10
apicurito                Design beautiful, functional APIs with zero coding, using a visual designer f...   7 (1 blank)       7
cache-service            Red Hat Data Grid is an in-memory, distributed key/value store.                    8 (1 blank)       4
cakephp-mysql-example    An example CakePHP application with a MySQL database. For more information ab...   21 (4 blank)      8
>--SNIP--<
```

oc process

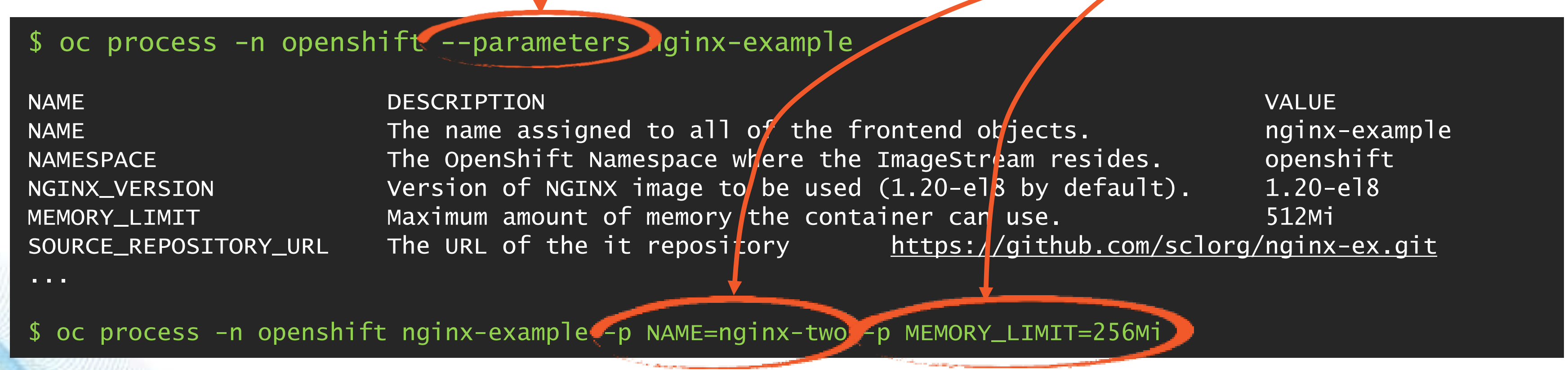
- The `oc process` subcommand processes a template
  - This produces a valid YAML manifest for the template objects with the specified parameter values filled in.
  - The output is sent to `stdout`, but could equally be piped to `oc create` to actually create the template objects.

```
$ oc process -n openshift nginx-example
```

neueda
futureproof your workforce

Viewing the parameters

- The `--parameters` option will display the configurable parameters from the template, in case you need to adjust the defaults
- Parameters can be set in `oc process` using successive `--param` or `-p` arguments



```
$ oc process -n openshift --parameters nginx-example

NAME                     DESCRIPTION                                           VALUE
NAME                     The name assigned to all of the frontend objects.     nginx-example
NAMESPACE                The OpenShift Namespace where the ImageStream resides.  openshift
NGINX_VERSION            Version of NGINX image to be used (1.20-el8 by default).  1.20-el8
MEMORY_LIMIT             Maximum amount of memory the container can use.       512Mi
SOURCE_REPOSITORY_URL    The URL of the it repository     https://github.com/sclorg/nginx-ex.git
...

$ oc process -n openshift nginx-example -p NAME=nginx-two -p MEMORY_LIMIT=256Mi
```

Instantiating a Template in the CLI

- To instantiate in the CLI
    - Process a template
    - Feed the resulting YAML into the `oc create -f` subcommand

```
$ oc process -n openshift nginx-example -p NAME=nginx-two | oc create -f -
service/nginx-two created
route.route.openshift.io/nginx-two created
imagestream.image.openshift.io/nginx-two created
buildconfig.build.openshift.io/nginx-two created
deploymentconfig.apps.openshift.io/nginx-two created
$ oc get dc nginx-two
NAME        REVISION    DESIRED    CURRENT    TRIGGERED BY
nginx-two   0           1          0          config,image(nginx-two:latest)
```

This means read from stdin

A Template file is JSON/YAML with these fields:

Metadata

- This section explains what the template does, and adds search tags. It also can select an icon from a list and gives urls for support etc.

Labels

- Labels that are added to each object created when the template is instantiated

objects

parameters

Writing your own OpenShift Template

```
apiVersion: template.openshift.io/v1
kind: Template

metadata:          # This section explains what the template does, and adds search
                   # tags. It also can select an icon from a list and gives links
                   # for support etc.

message:           # A message returned to the user on instantiation

labels:            # Labels to be added to each object created on instantiation

objects:           # This is the heart of the template - the list of objects that
                   # will be created when the template is instantiated.

parameters:        # All the configurable parameters of the template - can include
                   # auto-generated passwords etc.
```

neueda
futureproof your workforce

The Metadata Section

```
apiVersion: template.openshift.io/v1
kind: Template

metadata:
  name: redis-template                  # The unique template name
  annotations:                          # This subsection lists detail of for the
                                        # template. There are lots of optional fields
                                        # that can be used for template registries.
    description: "Description“          # A description with enough detail for users
                                        # to understand what is being deployed, links
                                        # to additional info etc. Can have newlines
    iconClass: "icon-redis“             # An icon for the Web Console (from a list)
    tags: "database,nosql"              # Tags for searching and grouping

message:
[...]
```

The Message Section

```
apiVersion: template.openshift.io/v1
kind: Template
metadata:

message: "Your password is ${REDIS_PASSWORD}“  # Parameters will be filled in at
                                               # instantiation.


labels:
objects:
parameters:
```

The Labels Section

```
apiVersion: template.openshift.io/v1
kind: Template
metadata:
message:

labels:                           # These labels are applied to each object created when
                                  # the template is instantiated. The labels can be
                                  # parameterized.
  template: redis-template
  redis: master
  app: ${APP}

objects:
parameters:
```

The Parameters Section

```
apiVersion: template.openshift.io/v1
kind: Template
metadata:
message:
labels:
objects:

parameters:                           # a list of configurable parameters
- name: APP                           # a simple parameter with a fixed value
  description: The example app
  value: example
- name: REDIS_PASSWORD                # an auto-generated 8 character password
  description: Password used for Redis authentication
  from: '[A-Z0-9]{8}'
  generate: expression
```

The Objects Section

```
objects:                                  # A list of objects to be instantiated
- apiVersion: v1                          # In this case only instantiate a single Pod
  kind: Pod
  metadata:
    name: redis-master
  spec:
    containers:
    - env:
      - name: REDIS_PASSWORD
        value: ${REDIS_PASSWORD}          # The environment variable is taken from the
                                          # parameters list.
      image: dockerfile/redis
      name: master
      ports:
      - containerPort: 6379
        protocol: TCP
```

neueda futureproof your workforce

Template and Code Reuse

- The example above is a very simple templates, more sophisticated templates can produce any deployment you can think of.
- Templates can become a useful code reuse tool - they can be checked into Git and versioned.
- For more information see the OpenShift documentation

Lab 4: Creating and Populating a DB from a Template

OPERATORS

Operators

- An Operator is an application in OpenShift
    - It is an application that manages another application (the **operand** - usually a backend service).
- Operators extend OpenShift by teaching it how to manage your application.
- An Operator makes a service self-managing.
    - Install, upgrade, keep the service running, track metrics etc.

Why Use Operators?

- Lower the barrier to using your service:
    - Skill, time or interest for manual manage are not necessary.
- Services in the main OpenShift service catalogues (OperatorHub.io, RedHat marketplace) *must* have Operators.
- Services in other service catalogues will also benefit from having an Operator.

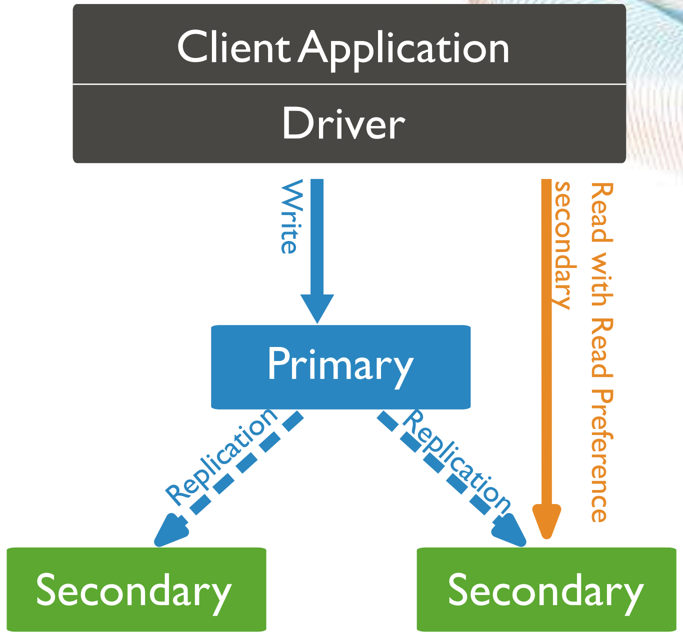
neueda
futureproof your workforce

Stateful apps are a problem in Kubernetes

- K8s manages stateless apps
  - Pod replicas are the same, controllers work for all apps.
- With stateful apps, some pods are different and use unique resources (e.g. Persistent Volumes, PVCs)
- Site-reliability engineers manage stateful apps.

Stateful App Example: MongoDB

- All Pods are *not* the same.
  - One Pod is the Primary, and all others are Secondary Pods federated into the Primary.
  - If the Primary fails, one of the Secondary is promoted and all remaining Secondaries are re-federated into the new Primary.
- Normally a Site Reliability Engineer would be responsible for this.



https://www.mongodb.com/docs/manual/replication/

Operators Help with Stateful Apps

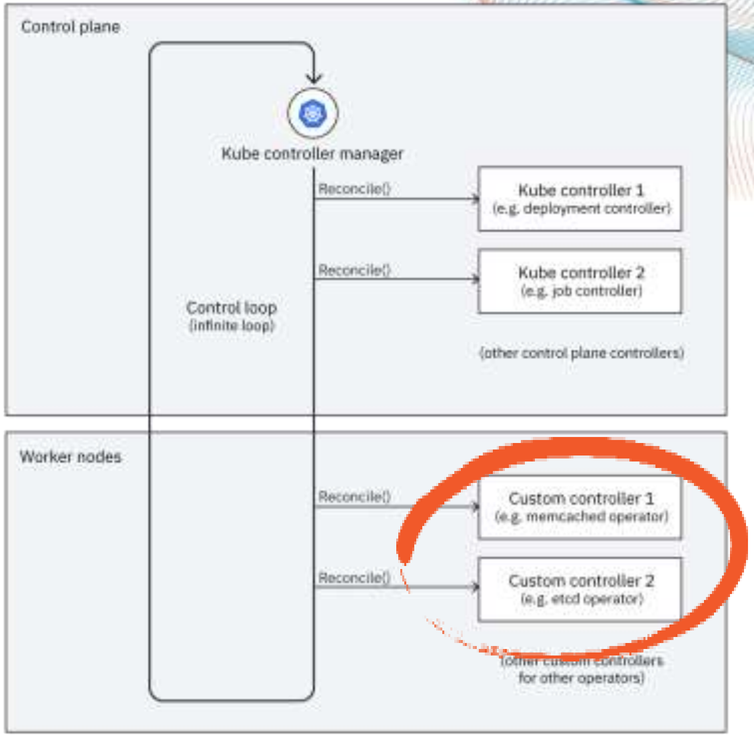
- The Operator knows how the stateful app works and make them more self-managing.
- One Operator can manage multiple operands of the same type.
- Each Operand is configured through variables.
- Operators automate a site reliability engineer’s grunt work.

Operator Structure

- An Operator runs as an Image in a Container like other applications
- The Operator heart is a Controller, and the main component of the Controller is the `Reconcile()` method.
- The Operator defines its own K8s `kind` through a Custom Resource Definition (CRD).
- Custom Resources are instances of the CRD

Operators Extend Kubernetes

- In normal running, the Kube controller manager runs an infinite loop calling the `Reconcile()` method of the control plane controllers
  - Checks jobs, deployments etc are matching their declarations.
- Operators just hook onto this loop!

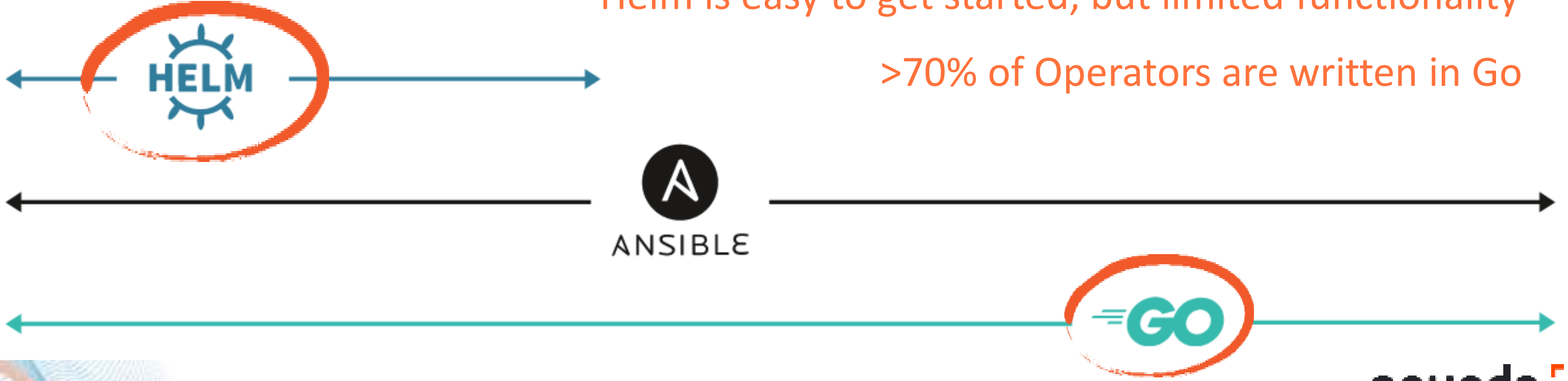


Supported Operator Languages and Control Levels

Level I	Level II	Level III	Level IV	Level V
Basic Install	**Seamless Upgrades**	**Full Lifecycle**	**Deep Insights**	**Auto Pilot**
Automated application provisioning and configuration management	Patch and minor version upgrades supported	App lifecycle, storage lifecycle (backup, failure recovery)	Metrics, alerts, log processing and workload analysis	Horizontal/vertical scaling, auto config tuning, abnormal detection, scheduling tuning

Helm is easy to get started, but limited functionality

>70% of Operators are written in Go



neueda
futureproof your workforce

The Operator Lifecycle Manager

- In OpenShift an Operator Subscription is added for each (desired) Operator from the OperatorHub.
- Operators have their own Operator in OpenShift - the Operator Lifecycle Manager (OLM) - installed by default from OpenShift 4.10.
  - This keeps track of managing the installed Operators.

Operators Conclusion

- Operators shepherd foundation services with custom logic.
- It means adding a database, or a message queue or other common backend services is similar to using a managed cloud service.
- If you are thinking to supply such a service, it is highly recommended you accompany it with an Operator!

Summary

- OpenShift Templates
- Templates in the Web Console and the CLI
- Template File Structure
- Kubernetes Operators
- Operators Structure and operation

Questions and Comments?

neueda
futureproof your workforce