kubragen2

Rangel Reale

Nov 27, 2020

CONTENTS:

KubraGen2 is a Kubernetes YAML generator library that makes it possible to generate configurations using the full power of the Python programming language.

Combined with Helmion, it is possible to use Helm as a Kubernetes Yaml source, customize them, and generate a script for applying it directly without using the Helm release process.

See Kubragen2 Samples for real-life examples.

CHAPTER

ONE

OVERVIEW

See source code for examples

- Website: https://github.com/RangelReale/kubragen2
- Repository: https://github.com/RangelReale/kubragen2.git
- Documentation: https://kubragen2.readthedocs.org/
- PyPI: https://pypi.python.org/pypi/kubragen2

CHAPTER

TWO

EXAMPLE

```
import argparse
import datetime
import os

from helmion.chart import ProcessorChain, Chart
from helmion.helmchart import HelmRequest
from helmion.processor import DefaultProcessor, FilterRemoveHelmData, ListSplitter
from helmion.resource import is_any_resource
from kubragen2.build import BuildData
from kubragen2.data import ValueData
from kubragen2.kdata import KData_PersistentVolume_HostPath, KData_
↪PersistentVolumeClaim
from kubragen2.output import OutputProject, OutputFile_ShellScript, OutputFile_
↪Kubernetes, OD_FileTemplate, \
    OutputDriver_Directory
from kubragen2.provider.aws import KData_PersistentVolume_CSI_AWSEBS
from kubragen2.provider.digitalocean import KData_PersistentVolume_CSI_DOBS
from kubragen2.provider.gcloud import KData_PersistentVolume_GCEPersistentDisk
from kubragen2.provider.local import KData_PersistentVolumeClaim_NoSelector


def main():
    parser = argparse.ArgumentParser(description='Kube Creator')
    parser.add_argument('-p', '--provider', help='provider', required=True, choices=[
        'google-gke',
        'amazon-eks',
        'digitalocean-kubernetes',
        'k3d',
    ])
    parser.add_argument('--no-resource-limit', help='don''t limit resources', action=
↪'store_true')
    parser.add_argument('-o', '--output-path', help='output path', default='output')
    args = parser.parse_args()

    pv_prometheus_files = None
    pvc_prometheus_files = None

    pvconfig = {
        'metadata': {
            'labels': {
                'pv.role': 'prometheus',
            },
        },
        'spec': {
```

(continues on next page)

(continued from previous page)

```
            'persistentVolumeReclaimPolicy': 'Retain',
            'capacity': {
                'storage': '50Gi'
            },
            'accessModes': ['ReadWriteOnce'],
        },
    }

    pvcconfig = {
        'spec': {
            'selector': {
                'matchLabels': {
                    'pv.role': 'prometheus',
                }
            },
            'accessModes': ['ReadWriteOnce'],
            'resources': {
                'requests': {
                    'storage': '50Gi',
                }
            },
        }
    }

    if args.provider == 'k3d':
        pv_prometheus_files = KData_PersistentVolume_HostPath(
            name='prometheus-storage', hostpath={'path': '/var/storage/prometheus'},␣
↪merge_config=pvconfig)
        pvc_prometheus_files = KData_PersistentVolumeClaim_NoSelector(
            name='prometheus-claim', volumeName='prometheus-storage',
            namespace='monitoring', merge_config=pvcconfig)
    elif args.provider == 'google-gke':
        pv_prometheus_files = KData_PersistentVolume_GCEPersistentDisk(
            name='prometheus-storage', fsType='ext4',
            merge_config=pvconfig)
    elif args.provider == 'digitalocean-kubernetes':
        pv_prometheus_files = KData_PersistentVolume_CSI_DOBS(name='prometheus-storage
↪', csi={
                'fsType': 'ext4',
            }, merge_config=pvconfig)
    elif args.provider == 'amazon-eks':
        pv_prometheus_files = KData_PersistentVolume_CSI_AWSEBS(name='prometheus-
↪storage', csi={
                'fsType': 'ext4',
            }, merge_config=pvconfig)
    else:
        raise Exception('Unknown target')

    if pvc_prometheus_files is None:
        pvc_prometheus_files = KData_PersistentVolumeClaim(
            name='prometheus-claim', namespace='monitoring',
            storageclass='', merge_config=pvcconfig)

    # Add namespace to items, and filter Helm data from labels and annotations
    helm_default_processor = ProcessorChain(DefaultProcessor(add_namespace=True),␣
↪FilterRemoveHelmData())
```

(continues on next page)

(continued from previous page)

```
    def helm_splitter_crd(cat, chart, data):
            return is_any_resource(
                data, {'apiVersionNS': 'apiextensions.k8s.io', 'kind':
↪'CustomResourceDefinition'})

    def helm_splitter_config(cat, chart, data):
        return is_any_resource(
            data, {'apiVersionNS': 'rbac.authorization.k8s.io'},
            {'apiVersionNS': 'policy'},
            {'apiVersionNS': '', 'kind': 'ServiceAccount'},
            {'apiVersionNS': '', 'kind': 'Secret'},
            {'apiVersionNS': '', 'kind': 'ConfigMap'},
            {'apiVersionNS': 'monitoring.coreos.com'},
            {'apiVersionNS': 'admissionregistration.k8s.io'},
        )

    def helm_splitter_job(cat, chart, data):
        return is_any_resource(
            data, {'apiVersionNS': 'batch'})

    def helm_splitter_service(cat, chart, data):
        return is_any_resource(
            data, {'apiVersionNS': '', 'kind': 'Service'},
            {'apiVersionNS': '', 'kind': 'Pod'},
            {'apiVersionNS': '', 'kind': 'List'},
            {'apiVersionNS': 'apps', 'kind': 'Deployment'},
            {'apiVersionNS': 'apps', 'kind': 'DaemonSet'},
            {'apiVersionNS': 'apps', 'kind': 'StatefulSet'})

    # Start output
    out = OutputProject()

    shell_script = OutputFile_ShellScript('create_{}.sh'.format(args.provider))
    out.append(shell_script)

    shell_script.append('set -e')

    #
    # Provider setup
    #
    if args.provider == 'k3d':
        storage_directory = os.path.join(os.getcwd(), 'output', 'storage')
        if not os.path.exists(storage_directory):
            os.makedirs(storage_directory)
        if not os.path.exists(os.path.join(storage_directory, 'prometheus')):
            os.makedirs(os.path.join(storage_directory, 'prometheus'))
        shell_script.append(f'# k3d cluster create kg2sample-prometheus-stack --port␣
↪5051:80@loadbalancer --port 5052:443@loadbalancer -v {storage_directory}:/var/
↪storage')

    #
    # OUTPUTFILE: namespace.yaml
    #
    file = OutputFile_Kubernetes('namespace.yaml')
    file.append([{
        'apiVersion': 'v1',
        'kind': 'Namespace',
```

(continues on next page)

(continued from previous page)

```
        'metadata': {
            'name': 'monitoring',
        },
    }])
    out.append(file)
    shell_script.append(OD_FileTemplate(f'kubectl apply -f ${{FILE_{file.fileid}}}'))

    #
    # OUTPUTFILE: storage.yaml
    #
    file = OutputFile_Kubernetes('storage.yaml')

    file.append(pv_prometheus_files.get_value())
    # file.append(pvc_prometheus_files.get_value()) # this will be used only as spec in Helm

    out.append(file)
    shell_script.append(OD_FileTemplate(f'kubectl apply -f ${{FILE_{file.fileid}}}'))

    #
    # HELM: traefik2
    #
    helmreq = HelmRequest(repository='https://helm.traefik.io/traefik', chart='traefik', version='9.11.0',
                          releasename='traefik-router', # rename to avoid conflict with k3d
                          namespace='monitoring', values=BuildData({
            'ingressRoute': {
                'dashboard': {
                    'enabled': False,
                }
            },
            'providers': {
                'kubernetesCRD': {
                    'enabled': True,
                    'namespaces': [
                        'default',
                        'monitoring',
                    ]
                },
                'kubernetesIngress': {
                    'enabled': False,
                }
            },
            'logs': {
                'access': {
                    'enabled': True,
                }
            },
            'globalArguments': [
                '--global.checkNewVersion=false',
                '--global.sendAnonymousUsage=false',
            ],
            'additionalArguments': [
                '--api.debug=true',
                '--api.dashboard=true',
                '--api.insecure=false',
```

(continues on next page)

(continued from previous page)

```
                '--metrics.prometheus=true',
                '--metrics.prometheus.entryPoint=metrics',
                '--metrics.prometheus.addEntryPointsLabels=true',
            ],
            'ports': {
                'web': {
                    'expose': True,
                    'exposedPort': 80,
                },
                'websecure': {
                    'expose': False,
                },
                'api': {
                    'port': 8080,
                    'expose': True,
                },
                'metrics': {
                    'expose': True,
                    'port': 9090,
                },
            },
            'service': {
                'type': 'NodePort' if args.provider != 'k3d' else 'ClusterIP',
            },
            'resources': ValueData(value={
                'requests': {
                    'cpu': '100m',
                    'memory': '200Mi',
                },
                'limits': {
                    'cpu': '200m',
                    'memory': '300Mi',
                },
            }, enabled=not args.no_resource_limit),
        }))

    traefik_helmchart = helmreq.generate().process(helm_default_processor).split(
        ListSplitter({
            'crd': helm_splitter_crd,
            'config': helm_splitter_config,
            'service': helm_splitter_service,
        }, exactly_one_category=True))

    #
    # OUTPUTFILE: traefik-config-crd.yaml
    #
    file = OutputFile_Kubernetes('traefik-config-crd.yaml')

    file.append(traefik_helmchart['crd'].data)

    out.append(file)
    shell_script.append(OD_FileTemplate(f'kubectl apply -f ${{{FILE_{file.fileid}}}}'))

    #
    # OUTPUTFILE: traefik-config.yaml
    #
    file = OutputFile_Kubernetes('traefik-config.yaml')
```

(continues on next page)

(continued from previous page)

```
    file.append(traefik_helmchart['config'].data)

    file.append([{
        'apiVersion': 'traefik.containo.us/v1alpha1',
        'kind': 'IngressRoute',
        'metadata': {
            'name': 'traefik-api',
            'namespace': 'monitoring',
        },
        'spec': {
            'entryPoints': ['api'],
            'routes': [{
                'match': 'Method(`GET`)',
                'kind': 'Rule',
                'services': [{
                    'name': 'api@internal',
                    'kind': 'TraefikService'
                }]
            }]
        }
    }])

    out.append(file)
    shell_script.append(OD_FileTemplate(f'kubectl apply -f ${{FILE_{file.fileid}}}'))



    #
    # HELM: prometheus
    #
    helmreq = HelmRequest(repository='https://prometheus-community.github.io/helm-
↪charts',
                          chart='kube-prometheus-stack',
                          version='12.2.3', releasename='kube-prometheus-stack',
                          namespace='monitoring', values=BuildData({
            'alertmanager': {
                'resources': ValueData(value={
                    'requests': {
                        'cpu': '50m',
                        'memory': '100Mi'
                    },
                    'limits': {
                        'cpu': '100m',
                        'memory': '150Mi',
                    },
                }, enabled=not args.no_resource_limit),
            },
            'grafana': {
                'enabled': True,
                'adminPassword': 'grafana123',
                'resources': ValueData(value={
                    'requests': {
                        'cpu': '50m',
                        'memory': '100Mi'
                    },
                    'limits': {
```

(continues on next page)

(continued from previous page)

```
                    'cpu': '100m',
                    'memory': '128Mi',
                },
            }, enabled=not args.no_resource_limit),
        },
        'prometheusOperator': {
            'enabled': True,
            'resources': ValueData(value={
                'requests': {
                    'cpu': '50m',
                    'memory': '50Mi'
                },
                'limits': {
                    'cpu': '120m',
                    'memory': '120Mi'
                },
            }, enabled=not args.no_resource_limit),
        },
        'kube-state-metrics': {
            'resources': ValueData(value={
                'requests': {
                    'cpu': '10m',
                    'memory': '32Mi',
                },
                'limits': {
                    'cpu': '100m',
                    'memory': '64Mi',
                }
            }, enabled=not args.no_resource_limit),
        },
        'prometheus-node-exporter': {
            'resources': ValueData(value={
                'requests': {
                    'cpu': '150m',
                    'memory': '150Mi',
                },
                'limits': {
                    'cpu': '200m',
                    'memory': '200Mi'
                },
            }, enabled=not args.no_resource_limit),
        },
        'prometheus': {
            'service': {
                'port': 80,
            },
            'prometheusSpec': {
                'containers': [{
                    'name': 'prometheus',
                    'readinessProbe': {
                        'initialDelaySeconds': 30,
                        'periodSeconds': 30,
                        'timeoutSeconds': 8,
                    },
                }],
                'resources': ValueData(value={
                    'requests': {
```

(continues on next page)

(continued from previous page)

```
                            'cpu': '150m',
                            'memory': '350Mi'
                        },
                        'limits': {
                            'cpu': '300m',
                            'memory': '800Mi'
                        },
                    }, enabled=not args.no_resource_limit),
                    'storageSpec': {
                        'volumeClaimTemplate': {
                            'spec': pvc_prometheus_files.get_value()['spec'],
                        },
                    }
                }
            },
            'coreDns': {
                'enabled': args.provider != 'google-gke',
            },
            'kubeDns': {
                'enabled': args.provider == 'google-gke',
            },
        }))

    prometheus_helmchart = helmreq.generate().process(helm_default_processor).split(
        ListSplitter({
            'crd': helm_splitter_crd,
            'config': helm_splitter_config,
            'job': helm_splitter_job,
            'service': helm_splitter_service,
        }, exactly_one_category=True))

    #
    # OUTPUTFILE: prometheus-crd.yaml
    #
    file = OutputFile_Kubernetes('prometheus-crd.yaml')
    out.append(file)

    file.append(prometheus_helmchart['crd'].data)

    shell_script.append(OD_FileTemplate(f'kubectl apply -f ${{FILE_{file.fileid}}}'))

    #
    # OUTPUTFILE: prometheus-config.yaml
    #
    file = OutputFile_Kubernetes('prometheus-config.yaml')
    out.append(file)

    file.append(prometheus_helmchart['config'].data)

    shell_script.append(OD_FileTemplate(f'kubectl apply -f ${{FILE_{file.fileid}}}'))

    #
    # OUTPUTFILE: prometheus-job.yaml
    #
    file = OutputFile_Kubernetes('prometheus-job.yaml')
    out.append(file)
```

(continues on next page)

(continued from previous page)

```
    file.append(prometheus_helmchart['job'].data)

    shell_script.append(OD_FileTemplate(f'kubectl apply -f ${{FILE_{file.fileid}}}'))

    #
    # OUTPUTFILE: prometheus.yaml
    #
    file = OutputFile_Kubernetes('prometheus.yaml')
    out.append(file)

    file.append(prometheus_helmchart['service'].data)

    file.append([{
        'apiVersion': 'traefik.containo.us/v1alpha1',
        'kind': 'IngressRoute',
        'metadata': {
            'name': 'admin-prometheus',
            'namespace': 'monitoring',
        },
        'spec': {
            'entryPoints': ['web'],
            'routes': [{
                'match': f'Host(`admin-prometheus.localdomain`)',
                'kind': 'Rule',
                'services': [{
                    'name': 'kube-prometheus-stack-prometheus',
                    'namespace': 'monitoring',
                    'port': 80,
                }],
            }]
        }
    }, {
        'apiVersion': 'traefik.containo.us/v1alpha1',
        'kind': 'IngressRoute',
        'metadata': {
            'name': 'admin-grafana',
            'namespace': 'monitoring',
        },
        'spec': {
            'entryPoints': ['web'],
            'routes': [{
                'match': f'Host(`admin-grafana.localdomain`)',
                'kind': 'Rule',
                'services': [{
                    'name': 'kube-prometheus-stack-grafana',
                    'namespace': 'monitoring',
                    'port': 80,
                }],
            }]
        }
    }])

    shell_script.append(OD_FileTemplate(f'kubectl apply -f ${{FILE_{file.fileid}}}'))

    #
    # OUTPUTFILE: http-echo.yaml
    #
```

(continues on next page)

(continued from previous page)

```
    file = OutputFile_Kubernetes('http-echo.yaml')
    out.append(file)

    file.append([{
        'apiVersion': 'apps/v1',
        'kind': 'Deployment',
        'metadata': {
            'name': 'echo-deployment',
            'namespace': 'default',
            'labels': {
                'app': 'echo'
            }
        },
        'spec': {
            'replicas': 1,
            'selector': {
                'matchLabels': {
                    'app': 'echo'
                }
            },
            'template': {
                'metadata': {
                    'labels': {
                        'app': 'echo'
                    }
                },
                'spec': {
                    'containers': [{
                        'name': 'echo',
                        'image': 'mendhak/http-https-echo',
                        'ports': [{
                            'containerPort': 80
                        },
                        {
                            'containerPort': 443
                        }],
                    }]
                }
            }
        }
    },
    {
        'apiVersion': 'v1',
        'kind': 'Service',
        'metadata': {
            'name': 'echo-service',
            'namespace': 'default',
        },
        'spec': {
            'selector': {
                'app': 'echo'
            },
            'ports': [{
                'name': 'http',
                'port': 80,
                'targetPort': 80,
                'protocol': 'TCP'
```

(continues on next page)

(continued from previous page)

```
            }]
        }
    }, {
        'apiVersion': 'traefik.containo.us/v1alpha1',
        'kind': 'IngressRoute',
        'metadata': {
            'name': 'http-echo',
            'namespace': 'default',
        },
        'spec': {
            'entryPoints': ['web'],
            'routes': [{
                # 'match': f'Host(`http-echo.localdomain`)',
                'match': f'PathPrefix(`/`)',
                'kind': 'Rule',
                'services': [{
                    'name': 'echo-service',
                    'port': 80,
                }],
            }]
        }
    }])

    shell_script.append(OD_FileTemplate(f'kubectl apply -f ${{FILE_{file.fileid}}}'))

    #
    # OUTPUTFILE: traefik.yaml
    #
    file = OutputFile_Kubernetes('traefik.yaml')

    file.append(traefik_helmchart['service'].data)

    file.append({
        'apiVersion': 'traefik.containo.us/v1alpha1',
        'kind': 'IngressRoute',
        'metadata': {
            'name': 'admin-traefik',
            'namespace': 'monitoring',
        },
        'spec': {
            'entryPoints': ['web'],
            'routes': [{
                'match': f'Host(`admin-traefik.localdomain`)',
                'kind': 'Rule',
                'services': [{
                    'name': 'traefik-router',
                    'port': 8080,
                }],
            }]
        }
    })

    file.append({
        'apiVersion': 'monitoring.coreos.com/v1',
        'kind': 'ServiceMonitor',
        'metadata': {
            'name': 'traefik',
```

(continues on next page)

(continued from previous page)

```
            'namespace': 'monitoring',
            'labels': {
                'release': 'kube-prometheus-stack',
            },
        },
        'spec': {
            'selector': {
                'matchLabels': {
                    'app.kubernetes.io/name': 'traefik',
                    'app.kubernetes.io/instance': 'traefik-router',
                },
            },
            'namespaceSelector': {
                'matchNames': ['monitoring']
            },
            'endpoints': [{
                'port': 'metrics',
                'path': '/metrics',
            }],
        },
    })

    out.append(file)
    shell_script.append(OD_FileTemplate(f'kubectl apply -f ${{FILE_{file.fileid}}}'))

    #
    # OUTPUTFILE: ingress.yaml
    #
    file = OutputFile_Kubernetes('ingress.yaml')
    http_path = '/'
    if args.provider != 'k3d':
        http_path = '/*'

    ingress_chart = Chart(data=[
        {
            'apiVersion': 'extensions/v1beta1',
            'kind': 'Ingress',
            'metadata': {
                'name': 'ingress',
                'namespace': 'monitoring',
            },
            'spec': {
                'rules': [{
                    'http': {
                        'paths': [{
                            'path': http_path,
                            'backend': {
                                'serviceName': 'traefik-router',
                                'servicePort': 80,
                            }
                        }]
                    }
                }]
            }
        },
    ])
```

(continues on next page)

(continued from previous page)

```
    if args.provider == 'amazon-eks':
        ingress_chart = ingress_chart.process(DefaultProcessor(jsonpatches=[{
            'condition': [
                {'op': 'check', 'path': '/kind', 'cmp': 'equals', 'value': 'Ingress'},
            ],
            'patch': [
                {'op': 'merge', 'path': '/metadata', 'value': {'annotations': {
                    'kubernetes.io/ingress.class': 'alb',
                    'alb.ingress.kubernetes.io/scheme': 'internet-facing',
                    'alb.ingress.kubernetes.io/listen-ports': '[{"HTTP": 80}]',
                }}}
            ]
        }]))

    file.append(ingress_chart.data)

    out.append(file)
    shell_script.append(OD_FileTemplate(f'kubectl apply -f ${{FILE_{file.fileid}}}'))

    #
    # OUTPUT
    #
    out.output(OutputDriver_Print())

    # output_path = os.path.join(args.output_path, '{}-{}'.format(
    #     args.provider, datetime.datetime.today().strftime("%Y%m%d-%H%M%S")))
    # print('Saving files to {}'.format(output_path))
    # if not os.path.exists(output_path):
    #     os.makedirs(output_path)
    # out.output(OutputDriver_Directory(output_path))


if __name__ == "__main__":
    main()
```

Output:

```
****** BEGIN FILE: 001-namespace.yaml ********
apiVersion: v1
kind: Namespace
metadata:
  name: monitoring

****** END FILE: 001-namespace.yaml ********
****** BEGIN FILE: 002-storage.yaml ********
apiVersion: v1
kind: PersistentVolume
metadata:
  name: prometheus-storage
  labels:
    pv.role: prometheus
spec:
  hostPath:
    path: /var/storage/prometheus
  persistentVolumeReclaimPolicy: Retain
  capacity:
    storage: 50Gi
```

(continues on next page)

(continued from previous page)

```
  accessModes:
  - ReadWriteOnce

****** END FILE: 002-storage.yaml ********
****** BEGIN FILE: 003-traefik-config-crd.yaml ********
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: ingressroutes.traefik.containo.us
spec:
  group: traefik.containo.us
  version: v1alpha1
  names:
    kind: IngressRoute
    plural: ingressroutes
    singular: ingressroute
  scope: Namespaced
---
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
<...more...>

****** END FILE: 003-traefik-config-crd.yaml ********
****** BEGIN FILE: 004-traefik-config.yaml ********
kind: ServiceAccount
apiVersion: v1
metadata:
  name: traefik-router
  labels:
    app.kubernetes.io/name: traefik
    app.kubernetes.io/instance: traefik-router
  namespace: monitoring
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: traefik-router
<...more...>
****** END FILE: 004-traefik-config.yaml ********
****** BEGIN FILE: 005-prometheus-crd.yaml ********
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  annotations:
    controller-gen.kubebuilder.io/version: v0.2.4
  creationTimestamp: null
  name: alertmanagerconfigs.monitoring.coreos.com
spec:
  group: monitoring.coreos.com
  names:
    kind: AlertmanagerConfig
    listKind: AlertmanagerConfigList
    plural: alertmanagerconfigs
    singular: alertmanagerconfig
  scope: Namespaced
<...more...>
```

(continues on next page)

(continued from previous page)

```
****** END FILE: 005-prometheus-crd.yaml ********
****** BEGIN FILE: 006-prometheus-config.yaml ********
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: kube-prometheus-stack-grafana
  namespace: monitoring
  labels:
    app.kubernetes.io/name: grafana
    app.kubernetes.io/instance: kube-prometheus-stack
    app.kubernetes.io/version: 7.2.1
  annotations:
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: docker/default,runtime/
↪default
    seccomp.security.alpha.kubernetes.io/defaultProfileName: docker/default
    apparmor.security.beta.kubernetes.io/allowedProfileNames: runtime/default
    apparmor.security.beta.kubernetes.io/defaultProfileName: runtime/default
spec:
  privileged: false
  allowPrivilegeEscalation: false
  requiredDropCapabilities:
  - FOWNER
  - FSETID
  - KILL
  - SETGID
  - SETUID
  - SETPCAP
  - NET_BIND_SERVICE
  - NET_RAW
  - SYS_CHROOT
  - MKNOD
  - AUDIT_WRITE
  - SETFCAP
<...more...>

****** END FILE: 006-prometheus-config.yaml ********
****** BEGIN FILE: 007-prometheus-job.yaml ********
apiVersion: batch/v1
kind: Job
metadata:
  name: kube-prometheus-stack-admission-create
  namespace: monitoring
  annotations:
    helm.sh/hook: pre-install,pre-upgrade
    helm.sh/hook-delete-policy: before-hook-creation,hook-succeeded
  labels:
    app: kube-prometheus-stack-admission-create
    chart: kube-prometheus-stack-12.2.3
    release: kube-prometheus-stack
spec:
  template:
    metadata:
      name: kube-prometheus-stack-admission-create
      labels:
        app: kube-prometheus-stack-admission-create
        chart: kube-prometheus-stack-12.2.3
        release: kube-prometheus-stack
```

(continues on next page)

(continued from previous page)

```
        heritage: Helm
    spec:
      containers:
      - name: create
        image: jettech/kube-webhook-certgen:v1.5.0
        imagePullPolicy: IfNotPresent
        args:
        - create
        - --host=kube-prometheus-stack-operator,kube-prometheus-stack-operator.
↪monitoring.svc
        - --namespace=monitoring
        - --secret-name=kube-prometheus-stack-admission
        resources: {}
      restartPolicy: OnFailure
      serviceAccountName: kube-prometheus-stack-admission
      securityContext:
        runAsGroup: 2000
        runAsNonRoot: true
        runAsUser: 2000
---
apiVersion: batch/v1
kind: Job
metadata:
  name: kube-prometheus-stack-admission-patch
  namespace: monitoring
  annotations:
    helm.sh/hook: post-install,post-upgrade
    helm.sh/hook-delete-policy: before-hook-creation,hook-succeeded
  labels:
    app: kube-prometheus-stack-admission-patch
    chart: kube-prometheus-stack-12.2.3
    release: kube-prometheus-stack
spec:
  template:
    metadata:
      name: kube-prometheus-stack-admission-patch
      labels:
        app: kube-prometheus-stack-admission-patch
        chart: kube-prometheus-stack-12.2.3
        release: kube-prometheus-stack
        heritage: Helm
    spec:
      containers:
      - name: patch
        image: jettech/kube-webhook-certgen:v1.5.0
        imagePullPolicy: IfNotPresent
        args:
        - patch
        - --webhook-name=kube-prometheus-stack-admission
        - --namespace=monitoring
        - --secret-name=kube-prometheus-stack-admission
        - --patch-failure-policy=Fail
        resources: {}
      restartPolicy: OnFailure
      serviceAccountName: kube-prometheus-stack-admission
      securityContext:
        runAsGroup: 2000
```

(continues on next page)

(continued from previous page)

```
        runAsNonRoot: true
        runAsUser: 2000

****** END FILE: 007-prometheus-job.yaml ********
****** BEGIN FILE: 008-prometheus.yaml ********
apiVersion: v1
kind: Service
metadata:
  name: kube-prometheus-stack-grafana
  namespace: monitoring
  labels:
    app.kubernetes.io/name: grafana
    app.kubernetes.io/instance: kube-prometheus-stack
    app.kubernetes.io/version: 7.2.1
spec:
  type: ClusterIP
  ports:
  - name: service
    port: 80
    protocol: TCP
    targetPort: 3000
  selector:
    app.kubernetes.io/name: grafana
    app.kubernetes.io/instance: kube-prometheus-stack
---
apiVersion: v1
kind: Service
metadata:
  name: kube-prometheus-stack-kube-state-metrics
  namespace: monitoring
  labels:
    app.kubernetes.io/name: kube-state-metrics
    app.kubernetes.io/instance: kube-prometheus-stack
  annotations:
    prometheus.io/scrape: 'true'
spec:
  type: ClusterIP
  ports:
  - name: http
    protocol: TCP
    port: 8080
    targetPort: 8080
<...more...>
****** END FILE: 008-prometheus.yaml ********
****** BEGIN FILE: 009-http-echo.yaml ********
apiVersion: apps/v1
kind: Deployment
metadata:
  name: echo-deployment
  namespace: default
  labels:
    app: echo
spec:
  replicas: 1
  selector:
    matchLabels:
      app: echo
```

(continues on next page)

(continued from previous page)

```
  template:
    metadata:
      labels:
        app: echo
    spec:
      containers:
      - name: echo
        image: mendhak/http-https-echo
        ports:
        - containerPort: 80
        - containerPort: 443
---
apiVersion: v1
kind: Service
metadata:
  name: echo-service
  namespace: default
spec:
  selector:
    app: echo
  ports:
  - name: http
    port: 80
    targetPort: 80
    protocol: TCP
---
apiVersion: traefik.containo.us/v1alpha1
kind: IngressRoute
metadata:
  name: http-echo
  namespace: default
spec:
  entryPoints:
  - web
  routes:
  - match: PathPrefix(`/`)
    kind: Rule
    services:
    - name: echo-service
      port: 80

****** END FILE: 009-http-echo.yaml ********
****** BEGIN FILE: 010-traefik.yaml ********
apiVersion: apps/v1
kind: Deployment
metadata:
  name: traefik-router
  labels:
    app.kubernetes.io/name: traefik
    app.kubernetes.io/instance: traefik-router
  namespace: monitoring
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: traefik
      app.kubernetes.io/instance: traefik-router
```

(continues on next page)

(continued from previous page)

```
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
<...more...>

****** END FILE: 010-traefik.yaml ********
****** BEGIN FILE: 011-ingress.yaml ********
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingress
  namespace: monitoring
spec:
  rules:
  - http:
      paths:
      - path: /
        backend:
          serviceName: traefik-router
          servicePort: 80

****** END FILE: 011-ingress.yaml ********
****** BEGIN FILE: create_k3d.sh ********
#!/bin/bash

set -e
# k3d cluster create kg2sample-prometheus-stack --port 5051:80@loadbalancer --port␣
↪5052:443@loadbalancer -v /tmp/kubragen2_samples/prometheus_stack/output/storage:/
↪var/storage
kubectl apply -f 001-namespace.yaml
kubectl apply -f 002-storage.yaml
kubectl apply -f 003-traefik-config-crd.yaml
kubectl apply -f 004-traefik-config.yaml
kubectl apply -f 005-prometheus-crd.yaml
kubectl apply -f 006-prometheus-config.yaml
kubectl apply -f 007-prometheus-job.yaml
kubectl apply -f 008-prometheus.yaml
kubectl apply -f 009-http-echo.yaml
kubectl apply -f 010-traefik.yaml
kubectl apply -f 011-ingress.yaml

****** END FILE: create_k3d.sh ********
```

CHAPTER

THREE

AUTHOR

Rangel Reale (rangelreale@gmail.com)

3.1 Modules

3.1.1 The kubragen2.build module

kubragen2.build.**BuildData**(*data: Any*, *in_place: bool = True*) → Any

Cleanup all instances of Data classes, removing if not enabled or replacing by its value.

Parameters

- **data** – the data to mutate
- **in_place** – whether to modify the data in-place. If False, data will be duplicated using copy.deepcopy

Returns the same value passed, mutated, except if it is *Data{enabled=False}*, in this case it returns None.

3.1.2 The kubragen2.configfile module

3.1.3 The kubragen2.data module

class kubragen2.data.**Data**

Base class to represent data that can be disabled by a flag. The *get_value()* function allows for dynamic code generation if needed.

get_value() → Any

Returns the value of the data.

Returns the data value

is_enabled() → bool

Whether the data is enabled. If not, it will be removed by BuildData().

kubragen2.data.**DataGetValue**(*value: Any*, *raise_if_disabled: bool = False*) → Any

Returns the value. If value is an instance of *Data*, call its get_value() method, or return None if not enabled.

Parameters

- **value** – the value to check
- **raise_if_disabled** – whether to raise an exception if the value is disabled.

Returns the value

Raises ***InvalidParamError*** – if value is disabled

kubragen2.data.**DataIsNone**(*value: Any*) → bool
Checks if the value is None. If value is an instance of *Data*, check its *is_enabled()* method.

Parameters **value** – the value to check for None

Returns whether the value is None or disabled

class kubragen2.data.**DisabledData**
A *Data* class that is always disabled.

get_value() → Any
Returns the value of the data.

Returns the data value

is_enabled() → bool
Whether the data is enabled. If not, it will be removed by BuildData().

class kubragen2.data.**ValueConfiguration**(*value_path: str*)
Value configuration

class kubragen2.data.**ValueData**(*value: Any = None*, *enabled: bool = True*, *disabled_if_none: bool = False*)
A *Data* class with constant values.

Parameters

- **value** – the value to return in *get_value()*
- **enabled** – whether the data is enabled
- **disabled_if_none** – set enabled=False if value is None

get_value() → Any
Returns the value of the data.

Returns the data value

is_enabled() → bool
Whether the data is enabled. If not, it will be removed by BuildData().

3.1.4 The kubragen2.exception module

exception kubragen2.exception.**ConfigFileError**

exception kubragen2.exception.**InvalidOperationError**

exception kubragen2.exception.**InvalidParamError**

exception kubragen2.exception.**KG2Exception**

exception kubragen2.exception.**MergeError**

exception kubragen2.exception.**NotSupportedError**

3.1.5 The kubragen2.kdata module

3.1.6 The kubragen2.kdatahelper module

3.1.7 The kubragen2.kutil module

kubragen2.kutil.**secret_data_encode**(*data: Union[bytes, str]*) → str
Encode bytes or str Kuberentes secret. The encoding is done using base64, using the utf-8 charset.

Parameters **data** – Data to encode

Returns encoded secret

Raises ***KG2Exception*** – on error

kubragen2.kutil.**secret_data_encode_bytes**(*data: bytes*) → bytes
Encode bytes secret using the current provider. The encoding is done using base64, and raw bytes are returned

Parameters **data** – Data to encode

Returns encoded secret

Raises ***KG2Exception*** – on error

kubragen2.kutil.**unit_to_bytes**(*value: str*) → int
Convert Kubernetes units to byte.

Parameters **value** – a Kubernetes unit.

Returns the byte equivalent of the unit.

3.1.8 The kubragen2.merger module

3.1.9 The kubragen2.option module

3.1.10 The kubragen2.options module

3.1.11 The kubragen2.output module

3.1.12 The kubragen2.util module

kubragen2.util.**dict_flatten**(*d*, *parent_key=''*, *sep='.'*) → Mapping
Flatten a dict to a single level.

kubragen2.util.**dict_get_value**(*dict: Mapping*, *name: str*) → Any
Gets data from a dictionary using a dotted accessor-string

Parameters

- **dict** – source dictionary
- **name** – dotted value name

kubragen2.util.**dict_has_name**(*dict: Mapping*, *name: str*) → bool
Checks if the dict has a name using a dotted accessor-string

Parameters

- **dict** – source dictionary
- **name** – dotted value name

kubragen2.util.**urljoin**(**args: str*) → str
Join an array of strings using a forward-slash representing an url.

Parameters **args** – list of strings to join

Returns joined strings

3.1.13 The kubragen2.provider.aws module

3.1.14 The kubragen2.provider.digitalocean module

3.1.15 The kubragen2.provider.gcloud module

3.1.16 The kubragen2.provider.local module

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

k

INDEX