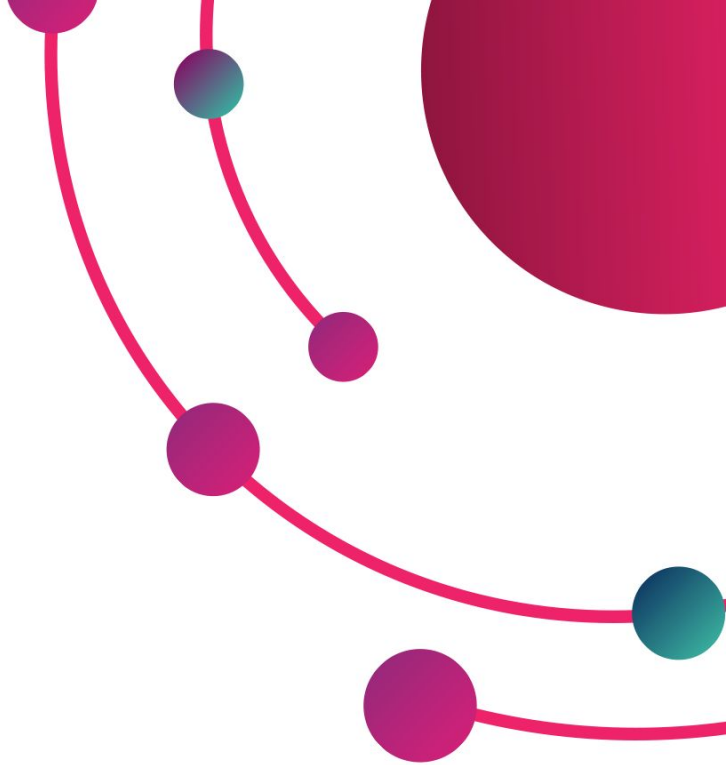


MIRANTIS

Introduction to YAML

... creating Deployments and other Kubernetes Objects

Who am I?

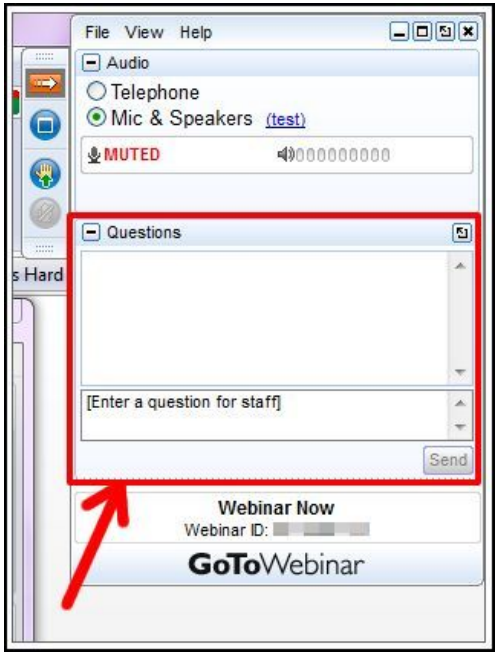
Nick Chase

Head of Technical Content at Mirantis

Nick Chase is Head of Technical Content for Mirantis and a member of the Kubernetes release team. He is a former software developer and author or co-author of more than a dozen books on various programming topics, including the *OpenStack Architecture Guide* and *Machine Learning for Mere Mortals*.

Housekeeping

- Please submit questions in the Questions panel.
- We'll provide a link where you can download the slides at the end of the webinar.



Who is this webinar for?

This webinar is for users who have a basic understanding of how Kubernetes works, but want to learn more about the YAML format often used to specify Kubernetes objects. We'll also cover some of the basic objects and how they fit together, but this won't be a Kubernetes tutorial.

Agenda

- Overall YAML structure
  - Workload-related objects
- Data types
  - Config-related objects
- Repeated nodes
  - Services-related objects
- Putting it together
  - Cluster-related objects

Overall YAML structure

A very simple YAML document: A Container

```
name: nginx
# Run the nginx:1.10 image
image: nginx:1.10
```

JSON version:

```
{
    "name": "nginx",
    "image": "nginx: 1.10"
}
```

Sequences: A Pod

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-example
  labels:
    - app: slingshot
    - status: approved
spec:
  containers:
  - name: ubuntu
    image: ubuntu:trusty
    command: ["echo"]
    args: ["Hello World"]
  - name: reporter
    image: sarahjane/reporter:v3
```

Sequences: A Pod

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    - app: slingshot
    - status: approved
  name: pod-example
spec:
  containers:
    - args:
        - "Hello World"
      command:
        - echo
      image: "ubuntu:trusty"
      name: ubuntu
    - image: "sarahjane/reporter:v3"
```

ReplicaSet

```
apiVersion: extensions/v1beta1
kind: ReplicaSet
metadata:
  # Unique key of the ReplicaSet instance
  name: replicaset-example
spec:
  # 3 Pods should exist at all times.
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      # Run the nginx image
      - name: nginx
        image: nginx:1.10
```

ReplicationController

```
apiVersion: v1
kind: ReplicationController
metadata:
  # Unique key of the ReplicationController instance
  name: replicationcontroller-example
spec:
  # 3 Pods should exist at all times.
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      # Run the nginx image
      - name: nginx
        image: nginx:1.10
```

Deployment

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: deployment-example
spec:
  replicas: 3
  template:
    metadata:
      labels:
        # Apply this label to pods and default
        # the Deployment label selector to this value
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.10
```

Multiple objects in a document: StatefulSet

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  ports:
  - port: 80
    name: web
  clusterIP: None
  selector:
    app: nginx
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
spec:
  serviceName: "nginx"
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: k8s.gcr.io/nginx-slim:0.8
        ports:
        - containerPort: 80
          name: web
        volumeMounts:
        - name: www
          mountPath: /usr/share/nginx/html
...
---
apiVersion: v1
kind: Pod
metadata:
  name: pod-example
   labels:
```

DaemonSet

```
apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
  # Unique key of the DaemonSet instance
  name: daemonset-example
spec:
  template:
    metadata:
      labels:
        app: daemonset-example
    spec:
```

```
containers:
# This container is run once on each
# Node in the cluster
- name: daemonset-example
  image: ubuntu:trusty
  command:
  - /bin/sh
  args:
  - -c
  # This script is run through
  # `sh -c <script>`
  - >-
    while [ true ]; do
    echo "DaemonSet running on $(hostname)" ;
    sleep 10 ;
    done
```

CronJob

```
apiVersion: batch/v1beta1
kind: CronJob
metadata:
 name: test
spec:
 schedule: "*/5 * * * *"
 jobTemplate:
   spec:
     template:
       spec:
         containers:
           - name: hello
             image: bash
             command:
                - echo
                - |
                  Hello world
                  It's good to see you!
         restartPolicy: OnFailure
```

Job

```
apiVersion: batch/v1
kind: Job
metadata:
  # Unique key of the Job instance
  name: example-job
spec:
  template:
    metadata:
      name: example-job
    spec:
      containers:
      - name: pi
        image: perl
        command: ["perl"]
        args: ["-Mbignum=bpi", "-wle", "print bpi(2000)"]
      # Do not restart containers after they exit
      restartPolicy: Never
```

Data Types

Data types: ConfigMap

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
  namespace: default
data:
  special.how: very
  weight: 42
  picture: |
    R0lGODdhDQAIAIAAAAAAANn
    Z2SwAAAAADQAIAAACF4SDGQ
    ar3xxbJ9p0qa7R0YxwzaFME
    1IAADs=
```

Data types: ConfigMap

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
  namespace: default
!!map data:
  special.how: !!str very
  weight: !!str 42
  picture: !!binary |
    R0lGODdhDQAIAIAAAAAAANn
    Z2SwAAAAADQAIAAACF4SDGQ
    ar3xxbJ9p0qa7R0YxwzaFME
    1IAADs=
```

Secret

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  username: !!base64 YWRtaW4=
  password: !!base64 MWYyZDFlMmU2N2Rm
```

Built in data types

- `str*`
- `map*`
- `int`
- `float`
- `boolean`
- `base64*`
- `binary*`
- `set`
- `timestamp`
- `hex`

Repeated nodes

Services

Service

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  labels:
 app: nginx
spec:
  ports:
  - port: 80
 name: web
  clusterIP: None
  selector:
 app: nginx
```

Service

```
kind: Service
apiVersion: v1
metadata:
  name: service-example
spec:
  ports:
  - name: http
      port: &openport 80
      targetPort: *openport
  selector:
  app: nginx
  type: LoadBalancer
```

Endpoints

```
apiVersion: v1
kind: Endpoints
metadata:
 name: mytest-cluster
subsets:
- addresses:
  - ip: 192.168.10.100
  ports:
  - name: myport
    port: 1
    protocol: TCP
- addresses:
  - ip: 192.168.10.101
  ports:
  - name: myport
    port: 1
    protocol: TCP
- addresses:
  - ip: 192.168.10.102
  ports:
  - name: myport
    port: 1
    protocol: TCP
```

Endpoints

```
apiVersion: v1
kind: Endpoints
metadata:
 name: glusterfs-cluster
subsets:
- addresses:
  - ip: 192.168.10.100
  ports: &stdport
  - name: myport
    port: 1
    protocol: TCP
- addresses:
  - ip: 192.168.10.101
  ports: *stdport
- addresses:
  - ip: 192.168.10.102
  ports: *stdport
```

Ingress

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /testpath
        backend:
          serviceName: test
          servicePort: 80
```

Ingress

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test-ingress
  annotations:
  nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /testpath
      backend: &stdbe
         serviceName: test
         servicePort: 80
      - path: /realpath
      backend: *stdbe
      - path: /hiddenpath
      backend: *stdbe
```

Ingress

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test-ingress
  annotations:
  nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /testpath
      backend: &stdbe
         serviceName: test
         servicePort: 80
      - path: /realpath
      backend: *stdbe
         servicePort: 443
      - path: /hiddenpath
      backend: *stdbe
```

Putting it together

Other objects you might encounter

StorageClass

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: standard
provisioner: kubernetes.io/aws-ebs
parameters:
  type: gp2
reclaimPolicy: Retain
mountOptions:
  - debug
volumeBindingMode: Immediate
```

Volume

```
apiVersion: v1
kind: Pod
metadata:
  name: test-ebs
spec:
  containers:
  - image: k8s.gcr.io/test-webserver
    name: test-container
    volumeMounts:
    - mountPath: /test-ebs
      name: test-volume
  volumes:
  - name: test-volume
    awsElasticBlockStore:
      volumeID: <volume-id>
      fsType: ext4
```

- awsElasticBlockStore
- azureDisk
- azureFile
- cephfs
- configMap
- csi
- downwardAPI
- emptyDir
- fc (fibre channel)
- flexVolume
- flocker
- gcePersistentDisk
- glusterfs
- hostPath
- iscsi
- local
- nfs
- persistentVolumeClaim
- projected
- portworxVolume
- quobyte
- rbd
- scaleIO
- secret
- storageos
- vsphereVolume

PersistentVolumeClaim

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc0001
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
```

PersistentVolume

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0003
spec:
  capacity:
    storage: 5Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: slow
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /tmp
    server: 172.17.0.2
```

Namespace

```
apiVersion: v1
kind: Namespace
metadata:
  name: my-namespace
```

Role

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""] # "" indicates the core API group
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```

RoleBinding

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: read-pods
  namespace: default
subjects:
- kind: User
  name: jane # Name is case sensitive
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role #this must be Role or ClusterRole
  name: pod-reader # this must match the name of the Role or ClusterRole
  apiGroup: rbac.authorization.k8s.io
```

ClusterRole

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
 name: restricted-psp-user
rules:
- apiGroups:
 - policy
 resources:
 - podsecuritypolicies
 resourceNames:
 - restricted
 verbs:
 - use
```

ClusterRoleBinding

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: read-secrets-global
subjects:
- kind: Group
  name: manager
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: secret-reader
  apiGroup: rbac.authorization.k8s.io
```

NetworkPolicy

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  - Egress
  egress:
  - to:
    - ipBlock:
        cidr: 10.0.0.0/24
    ports:
    - protocol: TCP
      port: 5978
```

```
ingress:
  - from:
    - ipBlock:
        cidr: 172.17.0.0/16
        except:
        - 172.17.1.0/24
    - namespaceSelector:
        matchLabels:
          project: myproject
    - podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 6379
```

Kubernetes Live Demo

Live Demo: Run a Multi-Node Kubernetes Cluster with kubeadm-dind-cluster

Date: Wednesday, February 20, 2019

Time: 9:00am PST / 17:00 UTC

Register at: http://bit.ly/kdc-demo

Thank You

Q&A

We will email you the slides and recording later this week.