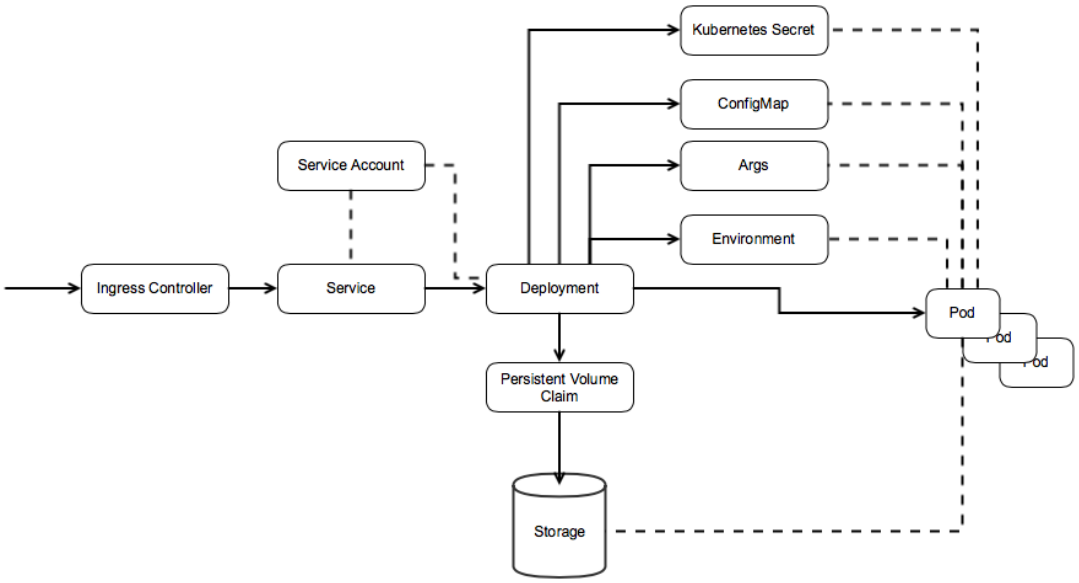
YAML Templates for GitOps Deployments

We want to create a set of YAML templates for the most common deployment components in Kubernetes and decide what parameters should be exposed for the teams to control.

Once we have a template for each different deployment components we can build a UI that reads the template and exposes the variables in the UI itself. When an application team wants to create/modify a deployment they can access the UI, change the variables point the git repo/folder/revision and the tool will generate the set of YAML files following the Kustomize format.

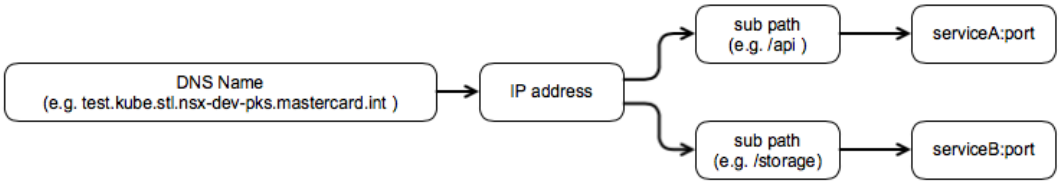
The application can also write the Application YAML format for ArgoCD allowing it to immediately point to the filled template and hence start synchronize the deployment on the target cluster.



Ingress Template

Fan Out Ingress

A fanout configuration routes traffic from a single IP address to more than one service, based on the HTTP URI being requested. An Ingress allows you to keep the number of loadbalancers down to a minimum.

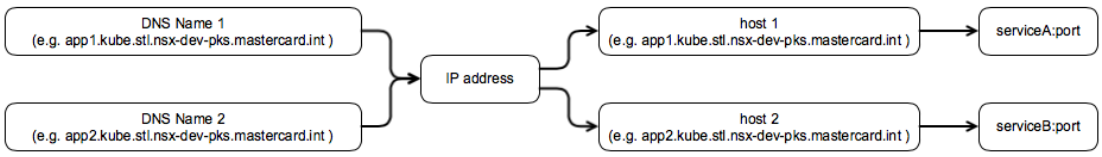


Ingress

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  annotations:
       nginx.ingress.kubernetes.io/rewrite-target: /
       kubernetes.io/ingress.class: {{application-name--ingress}}
  name: {{application-name-ingress}}
spec:
  rules:
  - host: {{host-api-fqdn}}
    http:
      paths:
      - backend:
          serviceName: {{service-name-A}}
          servicePort: {{service-port-A}}
        path: {{service-path-A}}
  - host: {{host-storage-fqdn}}
    http:
      paths:
      - backend:
          serviceName: {{service-name-B}}
          servicePort: {{service-port-B}}
        path: {{service-path-B}}
```

Name Based Virtual Hosting

Name-based virtual hosts support routing HTTP traffic to multiple host names at the same IP address.



Ingress

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: {{application-name-ingress}}
  annotations:
       kubernetes.io/ingress.class: {{application-name-ingress}}
spec:
  rules:
  - host: {{host-1-fqdn}}
    http:
      paths:
      - backend:
          serviceName: {{service-1-name}}
          servicePort: {{service-1-port}}
  - host: {{host-2-fqdn}}
    http:
      paths:
      - backend:
          serviceName: {{service-2-name}}
          servicePort: {{service-2-port}}
```

Service Account

Service Account

```
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app: {{application-name}}
    release: {{release-name-or-version}}
  name: {{application-name-sa}}
```

Service Template

Service

```
apiVersion: v1
kind: Service
metadata:
  labels:
    component: {{component-name}}
    app: {{application-name}}
    release: {{release-name-or-version}}
    tier: {{tier}}
  name: {{component-name-svc}}
spec:
  ports:
  - name: {{optional-name}}
    port: {{port}}
    protocol: TCP
    targetPort: {{target-port}}
  - name: {{optional-name}}
    port: {{port}}
    protocol: TCP
    targetPort: {{target-port}}
  selector:
    app: {{application-name}}
    component: {{component-name}}
  type: [[ExternalName,ClusterIP,LoadBalancer]]
```

Service Types

- **ClusterIP** is the default and is to expose things inside the cluster (Ingress will route from outside)
- **ExternalName** is for Kubernetes Service representation of things that are running outside Kubernetes but needs to be resolved inside.
- **LoadBalancer** can be used without Ingress to route traffic that does not use HTTP, e.g. gRPC or Kafka (Currently not supported because of firewall rule that accepts only port 443).

Common Labels

Common labels that can be used in the metadata labels sections are:

release : verN | stable | canary
environment : dev | qa | production
tier: frontend | backend | cache

Deployments

Deployment

```
apiVersion: apps/v1
kind: Deployment
```

```
metadata:
  name: {{application-name-deployment}}
spec:
  replicas: {{number-of-replicas}}
  selector:
    matchLabels:
      app: {{application-name}}
  strategy:
    type: [[RollingUpdate,Recreate]]
  template:
    metadata:
      labels:
        app: {{application-name}}
    spec:
      affinity:
        podAntiAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
          - podAffinityTerm:
              labelSelector:
                matchLabels:
                  app: {{application-name}}
                  component: {{component-name}}
                  release: {{release-name-or-version}}
              topologyKey: kubernetes.io/hostname
            weight: {{weight-value}}
      containers:
      - name: {{application-name}}
        image: {{AF-image-location>}}:{{version}}
        imagePullPolicy: [[Always,IfNotPresent]]
        env:
        - name: {{env-var-name-1}}
          value: {{env-var-value-1}}
        - name: {{env-var-name-N}}
          value: {{env-var-name-N}}

        # params and values need to have double quotes surrounding them
        args: [[{{param1}},{{value1}},{{paramN}},{{valueN}}]]

        resources:
           requests:
              memory: {{memory-request}}
              cpu: {{cpu-request}}
              ephemeral-storage: {{ephemeral-request}}
           limits:
              memory: {{memory-limit}}
              cpu: {{cpu-limit}}
              ephemeral-storage: {{ephemeral-limit}}
        livenessProbe:
          failureThreshold: 10
          httpGet:
            path: {{liveness-endpoint}}
            port: {{liveness-port}}
          initialDelaySeconds: 60
```

```
      timeoutSeconds: 30

    readinessProbe:
      httpGet:
        path: {{readiness-endpoint}}
        port: {{readiness-port}}

    volumeMounts:
    - mountPath: {{mount-path-1}}
      name: {{volume-name-1}}
      subPath: {{sub-path-1}}
    - mountPath: {{mount-path-2}}
      name: {{config-volume}}
      readOnly: true
    - mountPath: {{mount-path-N}}
      name: {{volume-name-N}}
      subPath: {{sub-path-N}}

    ports:
    - containerPort: {{port1}}
      name: {{name-of-connection}}
      protocol: [[TCP,UDP]]
    - containerPort: {{portN}}
      name: {{name-of-connection}}
      protocol: [[TCP,UDP]]

    volumes:
    - name: {{volume-name-1}}
      secret:
         items:
         - key: {{key-1}}
           path: {{value-1}}
         - key: {{key-N}}
           path: {{value-N}}
         secretName: {{secret-name}}
    - name: {{volume-name-N}}
      persistentVolumeClaim:
         claimName: {{persistent-volume-claim-name}}
    - name: {{config-volume-name}}
         configMap:
            name: {{configmap-name}}
```

Update Strategy

RollingUpdate works only if the container is stateless or it has a *ReadOnlyMany* or *ReadWriteMany*. If these conditions are not met and the container is attached to a host based volume and it is using the *ReadWriteOnce* it will need to use **Recreate**.

Persistent Volume Claims

Persistent Volume Claim

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: {{persistent-volume-claim-name-pvc}}
spec:
  accessModes:
  - [[ReadWriteOnce,ReadOnlyMany,ReadWriteMany]]
  resources:
    requests:
      storage: {{size-in-gigabytes}}
  storageClassName: {{available storage classes}}
```

Volume Strategies

ReadWriteOnce means that PVC can be attached to 1 pod at the time meaning cannot perform RollingUpgrade. ReadWriteMany uses NFS and it will allow RollingUpgrade