Testcase1

1. Testcase:
    a. {"field": "[\"spec\", \"workloads\", 0, \"env\", 0, \"envVars\"]", "testcase": "array-deletion"} #testrun-2024-02-28-00-04/trial-00-0002/0009
    b. {"field": "[\"spec\", \"workloads\", 0, \"env\", 0, \"envVars\"]", "testcase": "array-pop"} #testrun-2024-02-28-00-04/trial-00-0003/0003
    c. {"field": "[\"spec\", \"deployments\", 0, \"env\", 0, \"envVars\"]", "testcase": "array-deletion"} #testrun-2024-02-28-00-04/trial-01-0001/0005
    d. {"field": "[\"spec\", \"deployments\", 0, \"env\"]", "testcase": "array-deletion"} #testrun-2024-02-28-00-04/trial-09-0014/0009
2. What happened: Acto tried to remove the value from the envVars.
3. Root Cause: In the mergeEnv(src, tgt *[]v1.EnvVar), if the tgt already exists the key-value pairs, the value wouldn't be deleted.

```
23
24  func mergeEnv(src, tgt *[]v1.EnvVar) {
25      if len(*tgt) > 0 {
26          for _, srcV := range *src {
27              exists := false
28              for i, tgtV := range *tgt {
29                  if srcV.Name == tgtV.Name {
30                      (*tgt)[i] = srcV
31                      exists = true
32                  }
33              }
34              if !exists {
35                  *tgt = append(*tgt, srcV)
36              }
37          }
38      } else {
39          *tgt = *src
40      }
41  }
```

4. Expected behavior: This is the true alarm. While updating the cr.yaml, the operator should also check if the old key exists.

Testcase2

1. Testcase:
    a. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAffinity\", \"requiredDuringSchedulingIgnoredDuringExecution\", 0, \"topologyKey\"]", "testcase": "string-deletion"} #testrun-2024-02-28-00-04/trial-00-0007/0003
    b. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAffinity\", \"requiredDuringSchedulingIgnoredDuringExecution\", 0, \"topologyKey\"]", "testcase": "string-empty"} #testrun-2024-02-28-00-04/trial-00-0008/0003
    c. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAntiAffinity\", \"requiredDuringSchedulingIgnoredDuringExecution\", 0, \"topologyKey\"]", "testcase": "string-deletion"}
    d. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAntiAffinity\", \"requiredDuringSchedulingIgnoredDuringExecution\", 0, \"topologyKey\"]", "testcase": "string-empty"}

e. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAffinity\", \"preferredDuringSchedulingIgnoredDuringExecution\", 0, \"podAffinityTerm\", \"topologyKey\"]", "testcase": "string-deletion"}
f. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAffinity\", \"preferredDuringSchedulingIgnoredDuringExecution\", 0, \"podAffinityTerm\", \"topologyKey\"]", "testcase": "string-empty"}
g. {"field": "[\"spec\", \"workloads\", 0, \"affinity\", \"podAffinity\", \"preferredDuringSchedulingIgnoredDuringExecution\", 0, \"podAffinityTerm\", \"topologyKey\"]", "testcase": "string-change"}
h. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAntiAffinity\", \"preferredDuringSchedulingIgnoredDuringExecution\", 0, \"podAffinityTerm\", \"topologyKey\"]", "testcase": "string-empty"}
i. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAntiAffinity\", \"preferredDuringSchedulingIgnoredDuringExecution\", 0, \"podAffinityTerm\", \"topologyKey\"]", "testcase": "string-deletion"}
j. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAffinity\", \"requiredDuringSchedulingIgnoredDuringExecution\"]", "testcase": "array-pop"}
k. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAffinity\", \"preferredDuringSchedulingIgnoredDuringExecution\"]", "testcase": "array-pop"} # testrun-2024-02-28-00-04/trial-02-0022/0002
l. {"field": "[\"spec\", \"deployments\", 0, \"topologySpreadConstraints\", 0, \"topologyKey\"]", "testcase": "string-deletion"} #testrun-2024-02-28-00-04/trial-06-0015/0008

2. What happened: Acto tried to remove the value in the topologyKey. The topologyKey is empty. However, the operator couldn't find the specific node label specified in the topology key.
3. Root Cause: In kubernetes source code, ValidateTopologyKey() function will check if the value is empty or not.
4. Expected behavior: This is the true alarm. The operator should check if there is a value specified in topologykey.

Testcase3

1. Testcase:
    a. {"field": "[\"spec\", \"deployments\", 0, \"env\", 0, \"envVars\", 0, \"valueFrom\", \"resourceFieldRef\", \"divisor\"]", "testcase": "k8s-quantity_increase"} #testrun-2024-02-28-00-04/trial-00-0008/0003
    b. {"field": "[\"spec\", \"deployments\", 0, \"env\", 0, \"envVars\", 0, \"valueFrom\", \"resourceFieldRef\", \"divisor\"]", "testcase": "k8s-quantity_decrease"} #testrun-2024-02-28-00-04/trial-00-0009/0009
    c. {"field": "[\"spec\", \"workloads\", 0, \"env\", 0, \"envVars\", 0, \"valueFrom\", \"resourceFieldRef\", \"divisor\"]", "testcase": "k8s-quantity_increase"} # testrun-2024-02-28-00-04/trial-09-0005/0007

    d. {"field": "[\"spec\", \"workloads\", 0, \"env\", 0, \"envVars\", 0, \"valueFrom\", \"resourceFieldRef\", \"divisor\"]", "testcase": "k8s-quantity_decrease"} #testrun-2024-02-28-00-04/trial-09-0006/0002
2. What happened: Acto tried to increase and decrease the divisor, however, the name of the resource isn't valid.
3. Root Cause: In the validateContainerResourceFieldSelector() function, if the value in the resource field is set to the wrong value, the error occurs.
4. Expected behavior: This is the true alarm. The operator should check if the value specifics in the resourceFieldRef is correct in the validateEnvFrom()

Testcase4

1. Testcase:
    a. {"field": "[\"spec\", \"ingress\", \"kourier\", \"https-port\"]", "testcase": "integer-deletion"} #testrun-2024-02-28-00-04/trial-00-0013/0009
    b. {"field": "[\"spec\", \"ingress\", \"kourier\", \"http-port\"]", "testcase": "integer-deletion"} #testrun-2024-02-28-00-04/trial-10-0006/0002
2. What happened: Acto changed the kourier port from 5 to 0. However, the operator couldn't configure the http port.
3. Root Cause: By default, the Type in the kourier object is set to ClusterIP. However, if Type is not equal to NodePort, the error occurs in the func configureGatewayService(instance *v1beta1.KnativeServing) function.

```
// Configure HTTPPort/HTTPSPort if set.
if instance.Spec.Ingress.Kourier.HTTPPort > 0 || instance.Spec.Ingress.Kourier.HTTPSPort > 0 {
    if svc.Spec.Type != v1.ServiceTypeNodePort {
        return fmt.Errorf("cannot configure HTTP(S)Port for service type %q", svc.Spec.Type)
    }
    configureGatewayServiceTypeNodePort(instance, svc)
}
```

4. Expected behavior: This is the false alarm. While setting the http in kourier, Acto might need to set the nodeport in cr.yaml.

Testcase5

1. Testcase:
    a. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAntiAffinity\", \"preferredDuringSchedulingIgnoredDuringExecution\", 0, \"podAffinityTerm\"]", "testcase": "object-deletion"} #testrun-2024-02-28-00-04/trial-01-0005/0004
    b. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAffinity\", \"preferredDuringSchedulingIgnoredDuringExecution\", 0]", "testcase": "object-deletion"}
    c. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAntiAffinity\", \"preferredDuringSchedulingIgnoredDuringExecution\", 0]", "testcase": "object-deletion"}

d. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAffinity\", \"requiredDuringSchedulingIgnoredDuringExecution\"]", "testcase": "array-deletion"}
e. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAffinity\", \"preferredDuringSchedulingIgnoredDuringExecution\"]", "testcase": "array-deletion"} #testrun-2024-02-28-00-04/trial-02-0020/0009
f. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAntiAffinity\", \"preferredDuringSchedulingIgnoredDuringExecution\"]", "testcase": "array-deletion"} #testrun-2024-02-28-00-04/trial-03-0007/0001
g. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAntiAffinity\", \"preferredDuringSchedulingIgnoredDuringExecution\"]", "testcase": "array-pop"} # testrun-2024-02-28-00-04/trial-03-0009/0001
h. {"field": "[\"spec\", \"deployments\", 0, \"affinity\"]", "testcase": "object-deletion"} #testrun-2024-02-28-00-04/trial-03-0013/0006
i. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAffinity\", \"preferredDuringSchedulingIgnoredDuringExecution\", 0, \"podAffinityTerm\"]", "testcase": "object-deletion"} #testrun-2024-02-28-00-04/trial-07-0001/0001
j. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAntiAffinity\", \"requiredDuringSchedulingIgnoredDuringExecution\", 0]", "testcase": "object-deletion"} #testrun-2024-02-28-00-04/trial-07-0002/0004
k. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAntiAffinity\", \"requiredDuringSchedulingIgnoredDuringExecution\"]", "testcase": "array-deletion"} # testrun-2024-02-28-00-04/trial-10-0019/0001
l. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAntiAffinity\", \"requiredDuringSchedulingIgnoredDuringExecution\"]", "testcase": "array-pop"} #testrun-2024-02-28-00-04/trial-10-0021/0001
m. {"field": "[\"spec\", \"workloads\", 0, \"affinity\", \"podAntiAffinity\", \"preferredDuringSchedulingIgnoredDuringExecution\"]", "testcase": "array-deletion"} #testrun-2024-02-28-00-04/trial-11-0011/0008
n. {"field": "[\"spec\", \"deployments\", 0, \"topologySpreadConstraints\", 0]", "testcase": "object-deletion"} #testrun-2024-02-28-00-04/trial-05-0010/0001
o. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAffinity\"]", "testcase": "object-deletion"} #testrun-2024-02-28-00-04/trial-05-0011/0003
p. {"field": "[\"spec\", \"workloads\", 0, \"affinity\", \"podAffinity\"]", "testcase": "object-deletion"} #testrun-2024-02-28-00-04/trial-06-0000/0004
q. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAffinity\", \"requiredDuringSchedulingIgnoredDuringExecution\", 0]", "testcase": "object-deletion"} #testrun-2024-02-28-00-04/trial-08-0000/0009
r. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAntiAffinity\"]", "testcase": "object-deletion"} #testrun-2024-02-28-00-04/trial-11-0008/0003
s. {"field": "[\"spec\", \"deployments\"]", "testcase": "array-deletion"} #testrun-2024-02-28-00-04/trial-11-0001/0001
t. {"field": "[\"spec\", \"deployments\"]", "testcase": "array-pop"} #testrun-2024-02-28-00-04/trial-11-0003/0001

2. What's happened: Acto tried to delete the value from the spec.deployments[0].affinity.podAntiAffinity.preferredDuringSchedulingIgnoredDuringExecution[0].podAffinityTerm.labelSelector.matchLables.ACTOKEY path.
3. Root cause: In the withMatchLabel() function, the operator tried to update the label values. However, although the value updates to empty, the old keys still exist.

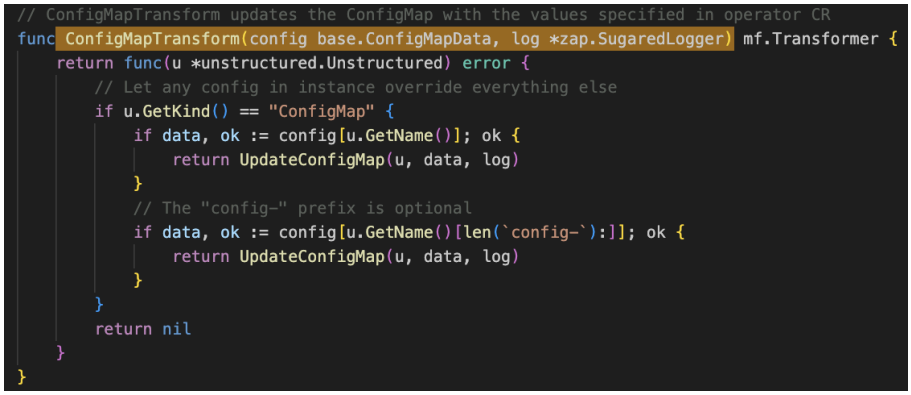
```go
func (b *LabelSelectorApplyConfiguration) WithMatchLabels(entries map[string]string) *LabelSelectorApplyConfiguration {
    if b.MatchLabels == nil && len(entries) > 0 {
        b.MatchLabels = make(map[string]string, len(entries))
    }
    for k, v := range entries {
        b.MatchLabels[k] = v
    }
    return b
}
```

4. Expected behavior: It is a true alarm. The operator should delete the old labels if it doesn't exist.

Testcase6

1. Testcase:
    a. {"field": "[\"spec\", \"config\", \"ACTOKEY\"]", "testcase": "object-deletion"} #testrun-2024-02-28-00-04/trial-01-0006/0004
    b. {"field": "[\"spec\", \"config\", \"ACTOKEY\"]", "testcase": "object-empty"} #testrun-2024-02-28-00-04/trial-01-0006/testrun-2024-02-28-00-04/trial-01-0007/0001
    c. {"field": "[\"spec\", \"config\", \"ACTOKEY\", \"ACTOKEY\"]", "testcase": "string-deletion"}
    d. {"field": "[\"spec\", \"config\", \"ACTOKEY\", \"ACTOKEY\"]", "testcase": "string-change"}
    e. {"field": "[\"spec\", \"config\", \"ACTOKEY\", \"ACTOKEY\"]", "testcase": "string-empty"}
    f. {"field": "[\"spec\", \"config\"]", "testcase": "object-deletion"} #testrun-2024-02-28-00-04/trial-10-0017/0001
2. What happened: Acto adds the ACTOKEY value to the spec.config.ACTOKEY.ACTOKEY path. However, it couldn't update the value successfully.
3. Root Cause: In the ConfigMapTransform(config base.ConfigMapData, log *zap.SugaredLogger), the operator iterates over all the configMap and then updates its

config value. However, the function couldn't find the configMap named ACTOKEY.

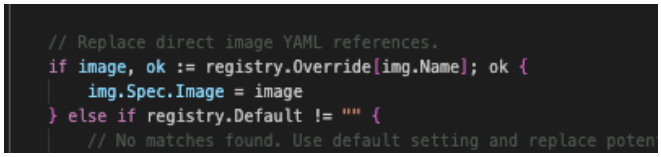


```go
// ConfigMapTransform updates the ConfigMap with the values specified in operator CR
func ConfigMapTransform(config base.ConfigMapData, log *zap.SugaredLogger) mf.Transformer {
    return func(u *unstructured.Unstructured) error {
        // Let any config in instance override everything else
        if u.GetKind() == "ConfigMap" {
            if data, ok := config[u.GetName()]; ok {
                return UpdateConfigMap(u, data, log)
            }
            // The "config-" prefix is optional
            if data, ok := config[u.GetName()[len(`config-`):]]; ok {
                return UpdateConfigMap(u, data, log)
            }
        }
        return nil
    }
}
```

4. Expected behavior: This is the false alarm. Inside the configmap, in the default value, there is no field named ACTOKEY. Perhaps, Acto needs to create the configmap with the specific label first.

Testcase7

1. Testcase:
    a. {"field": "[\"spec\", \"registry\", \"override\"]", "testcase": "object-deletion"}
    b. {"field": "[\"spec\", \"registry\", \"override\", \"ACTOKEY\"]", "testcase": "string-deletion"} #testrun-2024-02-28-00-04/trial-10-0000/0001
2. What happened: Acto tried to override the source of the image in the spec.registry.override.ACTOKEY path. However, the operator couldn't override it.
3. Root Cause: In the updateCachingImage() function, the operator tried to override the knative deployment images. However, it couldn't find the corresponding image name. Thus, the source of the image doesn't override.



```go
// Replace direct image YAML references.
if image, ok := registry.Override[img.Name]; ok {
    img.Spec.Image = image
} else if registry.Default != "" {
    // No matches found. Use default setting and replace poten
```

4. Expected behavior: This is a misoperation. When the operator does not find the image name, it doesn't execute override. However, the operator doesn't reject the wrong image name as well.

Testcase8

1. Testcase:
    a. {"field": "[\"spec\", \"deployments\", 0, \"env\", 0, \"envVars\"]", "testcase": "array-pop"}
2. What happened: Acto tried to remove all the environment variables. Before removing, Acto generated a bunch of envVars. However, the cli_status shows invalid because the divisor is set to an invalid value (divisor: +.5942).

3. Root Cause: The divisor is not a valid value, therefore, in kubernetes source code, the validateContainerResourceDivisor() occurs error.

```
func (b *ResourceFieldSelectorApplyConfiguration) WithDivisor(value resource.Quantity) *ResourceFieldSelectorApplyConfiguration {
    b.Divisor = &value
    return b
}
```

4. Expected behavior: This is the true alarm. The operator should check if the divisor is set to the valid value first in the validateEnvFrom() function.

Testcase9

1. Testcase:
    a. {"field": "[\"spec\", \"deployments\", 0, \"topologySpreadConstraints\"]", "testcase": "array-deletion"}
    b. {"field": "[\"spec\", \"deployments\", 0, \"topologySpreadConstraints\"]", "testcase": "array-push"}
    c. {"field": "[\"spec\", \"deployments\", 0, \"topologySpreadConstraints\"]", "testcase": "array-pop"}
2. What happened: Acto tried to delete the parameters inside the topologySpreadConstraints.
3. Root cause: If topologySpreadConstraints is specified, in kubernetes source code, the validateTopologySpreadConstraints() function couldn't find the required parameters. Thus, the error occurs.
4. Expected behavior: This is the true alarm. The operator should check if topologySpreadConstraints has the required parameters.

Testcase10

1. Testcase:
    a. {"field": "[\"spec\", \"services\", 0, \"selector\", \"ACTOKEY\"]", "testcase": "string-deletion"} #testrun-2024-02-28-00-04/trial-02-0007/0008
    b. {"field": "[\"spec\", \"services\", 0, \"selector\", \"ACTOKEY\"]", "testcase": "string-empty"} #testrun-2024-02-28-00-04/trial-02-0009/0001
2. What happened: Acto tried to delete the value in the spec.services[0].selector.ACTOKEY.
3. Root cause: In the overrideSelectors(), while removing the value, the old key still exists.

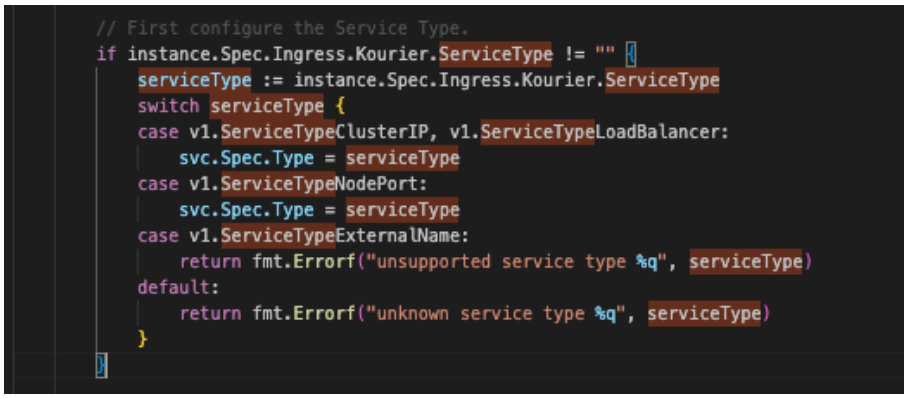
```
func overrideSelectors(override *base.ServiceOverride, service *corev1.Service) {
    if service.Spec.Selector == nil {
        service.Spec.Selector = map[string]string{}
    }

    for key, val := range override.Selector {
        service.Spec.Selector[key] = val
    }
}
```

4. Expected behavior: This is the true alarm. The operator should remove the key if the value is not presented.

Testcase11

1. Testcase:
    a. {"field": "[\"spec\", \"ingress\", \"kourier\", \"service-type\"]", "testcase": "string-empty"} #testrun-2024-02-28-00-04/trial-03-0002/0005
2. What happened: Acto tried to deploy the ingress, however, the service type does not provide inside cr.yaml.
3. Root Cause: In the configureGatewayService, the operator doesn't check whether the server-type is an empty string or not. Therefore, the envoy service couldn't create successfully, and the error message showed 'deployment: 3scale-kourier-gateway replicas [1] ready_replicas [None], 3scale-kourier-gateway condition [Available] status [False] message [Deployment does not have minimum availability.]\npod: 3scale-kourier-gateway-58c8cb46c6-8kffx container [kourier-gateway] restart_count [6]'.



```
// First configure the Service Type.
if instance.Spec.Ingress.Kourier.ServiceType != "" {
    serviceType := instance.Spec.Ingress.Kourier.ServiceType
    switch serviceType {
    case v1.ServiceTypeClusterIP, v1.ServiceTypeLoadBalancer:
        svc.Spec.Type = serviceType
    case v1.ServiceTypeNodePort:
        svc.Spec.Type = serviceType
    case v1.ServiceTypeExternalName:
        return fmt.Errorf("unsupported service type %q", serviceType)
    default:
        return fmt.Errorf("unknown service type %q", serviceType)
    }
}
```

4. Expected behavior: This is a misoperation. If the service-type isn't specified correctly, the operator should reject the undesired state.

Testcase12

1. Testcase:
    a. {"field": "[\"spec\", \"workloads\", 0, \"annotations\"]", "testcase": "object-deletion"} #testrun-2024-02-28-00-04/trial-04-0000/0007
    b. {"field": "[\"spec\", \"deployments\", 0, \"annotations\"]", "testcase": "object-deletion"} #testrun-2024-02-28-00-04/trial-04-0019/0007
    c. {"field": "[\"spec\", \"services\", 0, \"annotations\"]", "testcase": "object-deletion"} #testrun-2024-02-28-00-04/trial-05-0018/0007
    d. {"field": "[\"spec\", \"services\", 0, \"annotations\", \"ACTOKEY\"]", "testcase": "string-deletion"} # testrun-2024-02-28-00-04/trial-11-0014/0007
    e. {"field": "[\"spec\", \"services\", 0, \"annotations\", \"ACTOKEY\"]", "testcase": "string-empty"} #testrun-2024-02-28-00-04/trial-11-0016/0001
    f. {"field": "[\"spec\", \"services\", 0]", "testcase": "object-deletion"} #testrun-2024-02-28-00-04/trial-06-0001/0007
    g. {"field": "[\"spec\", \"deployments\", 0, \"annotations\", \"ACTOKEY\"]", "testcase": "string-deletion"} #testrun-2024-02-28-00-04/trial-09-0009/0001
    h. {"field": "[\"spec\", \"deployments\", 0, \"annotations\", \"ACTOKEY\"]", "testcase": "string-empty"} #testrun-2024-02-28-00-04/trial-09-0011/0001

i. {"field": "[\"spec\", \"services\"]", "testcase": "array-pop"} #testrun-2024-02-28-00-04/trial-10-0010/0001
j. {"field": "[\"spec\", \"services\"]", "testcase": "array-deletion"} #testrun-2024-02-28-00-04/trial-10-0008/0005

2. What happened: Actos tried to delete and empty the value in the annotations field.
3. Root Cause: In the replaceAnnotations function, the old key still exists while removing the value of the specific key.

```
func replaceAnnotations(override *base.WorkloadOverride, obj metav1.Object, ps *corev1.PodTemplateSpec) {
    if obj.GetAnnotations() == nil {
        obj.SetAnnotations(map[string]string{})
    }
    if ps.GetAnnotations() == nil {
        ps.SetAnnotations(map[string]string{})
    }
    for key, val := range override.Annotations {
        obj.GetAnnotations()[key] = val
        ps.Annotations[key] = val
```

4. Expected behavior: This is a true alarm. The operator should remove the old key.

Testcase13

1. Testcase:
    a. {"field": "[\"spec\", \"ingress\", \"kourier\", \"bootstrap-configmap\"]", "testcase": "string-deletion"} #testrun-2024-02-28-00-04/trial-04-0008/0002.
    b. {"field": "[\"spec\", \"ingress\", \"kourier\", \"bootstrap-configmap\"]", "testcase": "string-change"} #testrun-2024-02-28-00-04/trial-04-0009/0001
2. What happened: Acto tried to delete, and change the bootstrap-configmap value.
3. Root Cause: In the configureBootstrapConfigMap() function, the operator directly configures the bootstrapName. However, the config-map named ACTOKEY doesn't exist or the config-map i. Therefore, the error "MountVolume.SetUp failed for volume "config-volume" : configmap "ACTOKEY" not found" occurred.

```
    for i := range deployment.Spec.Template.Spec.Volumes {
        v := &deployment.Spec.Template.Spec.Volumes[i]
        if v.VolumeSource.ConfigMap.Name == kourierDefaultVolumeName {
            v.VolumeSource.ConfigMap = &v1.ConfigMapVolumeSource{
                LocalObjectReference: v1.LocalObjectReference{
                    Name: bootstrapName,
                },
```

4. Expected behavior: This is a misoperation. The operator should validate the value of the bootstrapName.

Testcase14

1. 1. Testcase
    a. {"field": "[\"spec\", \"ingress\", \"kourier\", \"bootstrap-configmap\"]", "testcase": "string-empty"} #testrun-2024-02-28-00-04/trial-04-0010/0001
2. What happened: Acto tried to remove the value in the bootstrap-configmp field.
3. Root Cause: In the configureGatewayService() function, because there is no information specified in the kourier field, the envoy service uses the state registration lookup by

default. However, the static registration lookup doesn't have any information. Therefore, the operator failed to create the kourier-gateway.

4. Expected behavior: This is a misoperation. The operator should check if the gateway info has been provided if the kourier is enabled.

Testcase15

1. Testcase
    a. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAntiAffinity\", \"preferredDuringSchedulingIgnoredDuringExecution\", 0, \"weight\"]", "testcase": "integer-deletion"} #testrun-2024-02-28-00-04/trial-04-0015/0002
    b. {"field": "[\"spec\", \"deployments\", 0, \"labels\", \"ACTOKEY\"]", "testcase": "string-deletion"} #testrun-2024-02-28-00-04/trial-10-0014/0001
    c. {"field": "[\"spec\", \"deployments\", 0, \"labels\", \"ACTOKEY\"]", "testcase": "string-empty"} #testrun-2024-02-28-00-04/trial-10-0016/0001
2. What happened: Acto tried to change the weight value.
3. Root Cause: The operator simply set the weight value of the last call in WithWeight(), but in kubernetes source code, validateWeightedPodAffinityTerms() occurs an error because the weight is set to zero.
4. Expected behavior: This is the true alarm. The operator should check if the weight value is correct.

Testcase16

1. Testcase:
    a. {"field": "[\"spec\", \"services\", 0, \"labels\", \"ACTOKEY\"]", "testcase": "string-deletion"} #testrun-2024-02-28-00-04/trial-06-0005/0001
    b. {"field": "[\"spec\", \"services\", 0, \"labels\", \"ACTOKEY\"]", "testcase": "string-empty"} #testrun-2024-02-28-00-04/trial-06-0007/0001
    c. {"field": "[\"spec\", \"deployments\", 0, \"labels\"]", "testcase": "object-deletion"} #testrun-2024-02-28-00-04/trial-07-0005/0002
    d. {"field": "[\"spec\", \"services\", 0, \"labels\"]", "testcase": "object-deletion"} #testrun-2024-02-28-00-04/trial-08-0017/0004
2. What happened: Acto tried to delete and empty the value in the spec.services[0].labels.ACTOEKY path.
3. Root Cause: In the overrideLables() function, the operator override the new value but doesn't remove the old keys.

```go
func overrideLabels(override *base.ServiceOverride, service *corev1.Service) {
    if service.GetLabels() == nil {
        service.Labels = map[string]string{}
    }

    for key, val := range override.Labels {
        service.Labels[key] = val
    }
}
```

4. Expected behavior: This is the true alarm. The operator should remove the old keys.

Testcase17

5. Testcase
    a. {"field": "[\"spec\", \"ingress\", \"kourier\", \"enabled\"]", "testcase": "boolean-deletion"} #testrun-2024-02-28-00-04/trial-06-0008/0005
    b. {"field": "[\"spec\", \"ingress\", \"kourier\", \"enabled\"]", "testcase": "boolean-toggle-off"} #testrun-2024-02-28-00-04/trial-06-0009/0001
    c. {"field": "[\"spec\", \"ingress\", \"kourier\", \"enabled\"]", "testcase": "boolean-toggle-on"} #testrun-2024-02-28-00-04/trial-06-0010/0001
6. What happened: Acto tried to enable kourier.
7. Root Cause: In the configureGatewayService() function, because there is no information specified in the kourier field, the envoy service uses the state registration lookup by default. However, the static registration lookup doesn't have any information. Therefore, the operator failed to create the kourier-gateway.
8. Expected behavior: This is a misoperation. The operator should check if the gateway info has been provided if the kourier is enabled.

Testcase18

1. Testcase:
    a. {"field": "[\"spec\", \"ingress\", \"kourier\", \"service-load-balancer-ip\"]", "testcase": "string-deletion"} #testrun-2024-02-28-00-04/trial-06-0017/0001
    b. {"field": "[\"spec\", \"ingress\", \"kourier\", \"service-load-balancer-ip\"]", "testcase": "string-change"} #testrun-2024-02-28-00-04/trial-06-0018/0001
    c. {"field": "[\"spec\", \"ingress\", \"kourier\", \"service-load-balancer-ip\"]", "testcase": "string-empty"} #testrun-2024-02-28-00-04/trial-06-0019/0001
2. What happened: Acto tried t o delete, change, and empty the service-load-balancer-ip value.
3. Root Cause: The operator failed to create the kourier-gateway because the ip is incorrect.
4. Expected behavior: This is a misoperation. The operator should check if the ip address is valid before creating the envoy.

Testcase19

1. Testcase:
    a. {"field": "[\"spec\", \"workloads\", 0, \"replicas\"]", "testcase": "k8s-overload"}
2. What happened: Acto tried to increase the replicas.

3. Root Cause: The current cluster state couldn't satisfy the number of replicas. Therefore, the error occurs "0/4 nodes are available: 1 node(s) had untolerated taint {node-role.kubernetes.io/control-plane: }, 3 Too many pods. preemption: 0/4 nodes are available: 1 Preemption is not helpful for scheduling, 3 No preemption victims found for incoming pod."
4. Expected behavior: This is the misoperation. The operator should update the cluster state.

Testcase20

1. Testcase:
    a. {"field": "[\"spec\", \"deployments\", 0, \"topologySpreadConstraints\", 0, \"maxSkew\"]", "testcase": "integer-deletion"} # testrun-2024-02-28-00-04/trial-08-0001/0004
2. What happened: Acto tried to change the maxskew value to 0.
3. Root Cause: 0 is an invalid value for maxskew. However, in the WithMaxSkew() function, this field is set to the value of the last call.
4. Expected behavior: This is a misoperation. The operator should validate if the maxskew value is valid.

Testcase21

1. Testcase:
    a. {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAffinity\", \"preferredDuringSchedulingIgnoredDuringExecution\", 0, \"weight\"]", "testcase": "integer-deletion"} #testrun-2024-02-28-00-04/trial-08-0005/0002
2. What happened: Acto tried to change the weight value to 0.
3. Root Cause: 0 is an invalid value to weight. However, in the WithWeight() function, this field is set to the value of the last call.
4. Expected behavior: This is a misoperation. The operator should validate if the weight value is valid.

Testcase22

1. Testcase:
    a. {"field": "[\"spec\", \"deployments\", 0, \"nodeSelector\"]", "testcase": "object-deletion"} # testrun-2024-02-28-00-04/trial-08-0015/0001
    b. {"field": "[\"spec\", \"deployments\", 0, \"nodeSelector\", \"ACTOKEY\"]", "testcase": "string-deletion"} #testrun-2024-02-28-00-04/trial-11-0005/0002
    c. {"field": "[\"spec\", \"deployments\", 0, \"nodeSelector\", \"ACTOKEY\"]", "testcase": "string-empty"} #testrun-2024-02-28-00-04/trial-11-0007/0001
2. What happened: Acto tried to remove the value from the spec.deployments[0].nodeSelector.

3. Root Cause: In the replaceNodeSelector() function, the operator wouldn't remove the old key-value in the nodeselector field.

```
func replaceNodeSelector(override *base.WorkloadOverride, ps *corev1.PodTemplateSpec) {
    if len(override.NodeSelector) > 0 {
        ps.Spec.NodeSelector = override.NodeSelector
    }
}
```

4. Expected behavior: This is the true alarm. The operator should remove the old keys.

Testcase23:

1. Testcase:
    a. {"field": "[\"spec\", \"high-availability\", \"replicas\"]", "testcase": "k8s-invalid_replicas"} #testrun-2024-02-28-00-04/trial-10-0002/0009
2. What happened: Acto tried to scale the replicas to 0.
3. Root Cause: In the system state, it showed that the operator attempted to scale to at least one replica. The error "Deployment does not have minimum availability" occurs.
4. Expected behavior: This is a misoperation. The operator should check if the replica value is valid.

Testcase24

1. Testcase:
    a. {"field": "[\"spec\", \"deployments\", 0, \"resources\", 0, \"limits\", \"memory\"]", "testcase": "string-change"} #testrun-2024-02-28-00-04/trial-10-0022/0006
2. What happened: Acto tried to change the value in memory.
3. Root Cause: The value 1926nTuGTGE0020723 specired in the memory field is invalid.
4. Expected behavior: This is the true alarm. The operator should check if the value is valid.

Testcase25:

1. Testcase:
    a. {"field": "[\"spec\", \"deployments\", 0, \"tolerations\"]", "testcase": "array-deletion"} #testrun-2024-02-28-00-04/trial-10-0023/0004
    b. {"field": "[\"spec\", \"deployments\", 0, \"tolerations\"]", "testcase": "array-pop"} #testrun-2024-02-28-00-04/trial-11-0018/0002
2. What happened: Acto tried to remove the key in the tolerations field.
3. Root Cause: In the validvalidateTolerations() function, if there is no key inside the tolerations field, the operator wouldn't remove the old key in this field.
4. Expected behavior: This is the true alarm. The operator should remove the old keys.

Testcase26:

1. Testcase:

    a. {"field": "[\"spec\", \"high-availability\", \"replicas\"]", "testcase": "k8s-overload"} #testrun-2024-02-28-00-04/trial-11-0000/0005
2. What happened: Acto tried to increase the replicas to 1000.
3. Root Cause: The current cluster state couldn't apply the request. The error " 0/4 nodes are available: 1 node(s) had untolerated taint {node-role.kubernetes.io/control-plane: }, 3 Too many pods. preemption: 0/4 nodes are available: 1 Preemption is not helpful for scheduling, 3 No preemption victims found for incoming pod.." occurs.
4. Expected behavior: This is a misoperation. The operator should update the cluster state.