Testcase1

1. Testcase: {"field": "[\"spec\", \"workloads\", 0, \"env\", 0, \"envVars\"]", "testcase": "array-deletion"} #testrun-2024-02-28-00-04/trial-00-0002/0009
2. Alarm: True
3. Acto's behavior: Removes the value from the path.
4. Reason: In the mergeEnv(src, tgt *[]v1.EnvVar), if the tgt already exists the key-value pairs, the value wouldn't be deleted.

Testcase2

1. Testcase: {"field": "[\"spec\", \"workloads\", 0, \"env\", 0, \"envVars\"]", "testcase": "array-pop"} #testrun-2024-02-28-00-04/trial-00-0007/0003
2. Alarm: True
3. Acto's behavior: Removes the value from the path.
4. Reason: In the mergeEnv(src, tgt *[]v1.EnvVar), if the tgt already exists the key-value pairs, the value wouldn't be deleted. (The same as Testcase1.)

Testcase3

1. Testcase: {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAffinity\", \"requiredDuringSchedulingIgnoredDuringExecution\", 0, \"topologyKey\"]", "testcase": "string-deletion"} #testrun-2024-02-28-00-04/trial-00-0007/0003
2. Alarm: False
3. Acto's behavior: Removes the topologyKey. The topologyKey is empty.
4. Reason: It looks like the tolologyKey is packed in a Selector object. Then, the operator calls the Matches() function to see if there are values associated with the tolologyKey. Since topologyKey has no default value and according to the document, the empty string is not allowed. Thus, the error occurs, but the code operates correctly.

Testcase4

1. Testcase: {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAffinity\", \"requiredDuringSchedulingIgnoredDuringExecution\", 0, \"topologyKey\"]", "testcase": "string-empty"} # testrun-2024-02-28-00-04/trial-00-0008/0003
2. Alarm: False
3. Acto's behavior: Empties the topologyKey.
4. Reason: It looks like the tolologyKey is packed in a Selector object. Then, the operator calls the Matches() function to see if there are values associated with the tolologyKey. Since topologyKey has no default value and according to the document, the empty string is not allowed. Thus, the error occurs, but the code operates correctly. (The same as Testcase3.)

Testcase5

1. Testcase: {"field": "[\"spec\", \"deployments\", 0, \"env\", 0, \"envVars\", 0, \"valueFrom\", \"resourceFieldRef\", \"divisor\"]", "testcase": "k8s-quantity_increase"} #testrun-2024-02-28-00-04/trial-00-0009/0009

2. Alarm: False
3. Acto's behavior: Increases the divisor from '2000m' to '4'.
4. Reason: In the validateEnvValueFrom(ctx context.Context, source *corev1.EnvVarSource) function, operator checks if the source contains invalid fields with CheckDisallowedFields() function. Acto sets the name of the resource to grznanvnsr in CR. However, the resource only supports the name such as limits.cpu, limits.memory. Thus, the error occurs.

Testcase6

1. Testcase: {"field": "[\"spec\", \"ingress\", \"kourier\", \"https-port\"]", "testcase": "integer-deletion"} #testrun-2024-02-28-00-04/trial-00-0013/0009
2. Alarm: False
3. Acto's behavior: Changes the kourier port from 5 to 0.
4. Reason: 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.

Testcase7

1. Testcase: {"field": "[\"spec\", \"deployments\", 0, \"env\", 0, \"envVars\"]", "testcase": "array-deletion"} #testrun-2024-02-28-00-04/trial-01-0001/0005
2. Alarm: True
3. Acto's behavior: Removes the value from the path.
4. Reason: In the mergeEnv(src, tgt *[]v1.EnvVar), if the tgt already exists the key-value pairs, the value wouldn't be deleted. (The same as Testcase1.)

Testcase8

1. Testcase: {"field": "[\"spec\", \"deployments\", 0, \"affinity\", \"podAntiAffinity\", \"preferredDuringSchedulingIgnoredDuringExecution\", 0, \"podAffinityTerm\"]", "testcase": "object-deletion"} #testrun-2024-02-28-00-04/trial-01-0005/0004
2. Alarm: False
3. Acto's behavior: Acto tried to delete the label from the path, but in this case, it wanna create the object with the label first. However, the error happens during the creating process.
4. Reason: The operator calls the Matches() function to check if there are values associated with matchLabels, but it couldn't find any. Another possible reason might be that in the CR file, the operator is set to ACTOKEY, which is an invalid value. However, the error message does not show in the alarm file. Therefore, it is speculated that the first reason is the root cause.

Testcase9

1. Testcase: {"field": "[\"spec\", \"config\", \"ACTOKEY\"]", "testcase": "object-deletion"} #testrun-2024-02-28-00-04/trial-01-0006/0004
2. Alarm: False

3. Acto’s behavior: Adds the ACTOKEY value to the spec.config.ACTOKEY.ACTOKEY path.
4. Reason: In the ConfigMapTransform(config base.ConfigMapData, log *zap.SugaredLogger), the operator iterate over all the configMap and then update its config value. However, the function could find the configMap named ACTOKEY.

Testcase10

1. Testcase: {"field": "[\"spec\", \"config\", \"ACTOKEY\"]", "testcase": "object-empty"} #testrun-2024-02-28-00-04/trial-01-0007/0001
2. Alarm: False
3. Acto’s behavior: Specifies empty string to the spec.config.ACTOKEY path.
4. Reason: The config type is map[string]map[string]string. Thus, the configMapTransform() function returns an error while updating the values specified in operator CR.