Winter Soldier : Scale down your infrastructure in the easiest way

Running Kubernetes can be very expensive, especially when done inefficiently. This is often the case when companies have just started to roll out Kubernetes in their organizations for different environments such as Staging, QA, Dev, Prod. etc. Still, over time these environments end up with workloads(k8s resources) that have outlived their utility and add to Total Cost of Ownership of infrastructure. Removing every unused resource manually can be a tedious task.

Now think out a situation where you want to scale down the non-prod infra automatically during the night or maybe weekends to avoid extra resource consumption, again hibernating the resources is a task that administrators avoid doing manually.

In this post, we'll focus on how to remove or scale down the orphaned (unused resources)

This can be achieved using **Winter Soldier**, no it's not Bucky, Captain America's best friend. It is an open-source command line utility tool by Devtron that is used to -

1.Delete k8s resource based on conditions.

2.Scale down the Workload to Zero at a Specific period of date & time.

Winter Soldier is an Operator which expects conditions to be defined using a CRD hibernator, it contains three config files.

1. **CRD** - An extension of the Kubernetes API that contains a custom definition of a hibernator.
2. **Winter-soldier deployment** - It contains cluster roles, role binding, service account & deployment required by the hibernator.
3. **Hibernator**- It contains custom policies based on which operations are done.

How to install winter soldier in K8S Cluster -

Note :- Currently, winter soldier supports k8s version less than or equal to v1.21

Step-1

```
kubectl apply -f
https://raw.githubusercontent.com/devtron-labs/winter-soldier/main/config/crd/bases/pincher.devtron.ai_hibernators.yaml
```

Step-2 : Create winter-soldier deployment

```
vim winter-soldier.yaml
```

```
And paste this content inside the Deployment
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/name: winter-soldier
    app.kubernetes.io/part-of: winter-soldier
    app: winter-soldier
  name: winter-soldier
  namespace: demo
spec:
  selector:
    matchLabels:
      app: winter-soldier
  replicas: 1
  template:
    metadata:
      labels:
        app: winter-soldier
    spec:
      containers:
      - command:
        - /manager
        args:
        - --enable-leader-election
        image: quay.io/devtron/winter-soldier:3f10a62b-196-5753
        name: manager
      terminationGracePeriodSeconds: 10
      serviceAccount: winter-soldier
      serviceAccountName: winter-soldier
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/name: winter-soldier
```

```
    app.kubernetes.io/part-of: winter-soldier
  name: winter-soldier
  namespace: demo
rules:
- apiGroups:
  - pincher.devtron.ai
  resources:
  - hibernators
  verbs:
  - create
  - delete
  - get
  - list
  - patch
  - update
  - watch
- apiGroups:
  - "*"
  resources:
  - "*"
  verbs:
  - get
  - patch
  - update
  - create
  - delete
  - list
  - watch
---
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/name: winter-soldier
    app.kubernetes.io/part-of: winter-soldier
  name: winter-soldier
  namespace: demo
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
```

```
metadata:
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/name: winter-soldier
    app.kubernetes.io/part-of: winter-soldier
  name: winter-soldier
  namespace: demo
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: winter-soldier
subjects:
- kind: ServiceAccount
  name: winter-soldier

  namespace: demo
```

Create the deployment in the demo namespace

```
kubectl apply -f winter-soldier.yaml -n demo
```

This will automatically create cluster-roles, role binding, Service account & deployment in demo namespace which are required by the hibernator.

Step-3 :

```
apiVersion: pincher.devtron.ai/v1alpha1
kind: Hibernator
metadata:
  name: k8s-hibernator
spec:
  timeRangesWithZone:
    timeZone: "Asia/Kolkata"
    timeRanges:
    - timeFrom: 00:00
      timeTo: 23:59:59
```

```
      weekdayFrom: fri
      weekdayTo: Sun
  selectors:
  - inclusions:
    - objectSelector:
        name: ""
        type: "deployment,statefulset"
        fieldSelector:
          - AfterTime(Now(),
AddTime(ParseTime({{metadata.creationTimestamp}},
'2006-01-02T15:04:05Z'), '1d'))
      namespaceSelector:
        name: "all"
    exclusions:
    - objectSelector:
        name: ""
        type: "deployment,statefulset,rollout"
      namespaceSelector:
        name:
"kube-system,demo,devtroncd,kube-node-lease,kube-public,argo"

  action: sleep
```

The above hibernator policy will reduce the replicas to zero as action set is sleep of all Deployments in all namespace excluding (kube-system, demo, devtroncd, kube-node-lease, kube-public, argo) namespace from Friday to Sunday at 00:00 to 23:59:59

At the end of the hibernation cycle, it sets the replica count of workload to the same number as before hibernation.

Configurations :

Actions :

Winter Soldier supports two type of actions on the workload

1. Delete : This action can be used to delete any Kubernetes object. eg

```
spec:
 action: delete
```

2. Sleep : This condition can be used to change replicas of workload to 0. eg

```
spec:
  action: sleep
```

Conditions :

Objects can be included and excluded based on

1. Label Selector
2. Object Kind
3. Name
4. Namespace
5. Any field in the kubernetes object

```
selectors:
- inclusions:
  - objectSelector:
      name: ""
      type: "deployment"
      fieldSelector:
      - AfterTime(Now(), AddTime(ParseTime({{metadata.creationTimestamp}},
'2006-01-02T15:04:05Z'), '10h'))
    namespaceSelector:
      name: "all"
  exclusions:
  - objectSelector:
      name: ""
      type: "deployment"
    namespaceSelector:
      name: "kube-system"
```

The above example will select `Deployment` kind objects which have been created 10 hours ago across all namespaces excluding `kube-system` namespace. Winter soldier exposes following functions to handle time, cpu and memory

- ParseTime - This function can be used to parse time. For eg to parse creationTimestamp use
- AddTime - This can be used to add time. For eg `AddTime(ParseTime({{metadata.creationTimestamp}}, '2006-01-02T15:04:05Z'), '-10h')` ll add 10h to the time. Use `d` for day, `h` for hour, `m` for minutes and `s` for seconds. Use negative number to get earlier time.

CpuToNumber - This can be used to compare CPU. For eg `any({{spec.containers.#.resources.requests}}, { MemoryToNumber(.memory) < MemoryToNumber('60Mi')})` will check if any `resource.requests` is less than `60Mi`

Time Range :

```
spec:
  action: sleep
  timeRangesWithZone:
    reSyncInterval: 300 # in minutes
    timeZone: "Asia/Kolkata"
    timeRanges:
      - timeFrom: 00:00
        timeTo: 23:59:59
        weekdayFrom: Sat
        weekdayTo: Sun
      - timeFrom: 00:00
        timeTo: 08:00
        weekdayFrom: Mon
        weekdayTo: Fri
      - timeFrom: 20:00
        timeTo: 23:59:59
        weekdayFrom: Mon
        weekdayTo: Fri
```

Above settings will take action on Sat and Sun from 00:00 to 23:59:59, and on Mon-Fri from 00:00 to 08:00 and 20:00 to 23:59:59. If `action:sleep` then runs hibernate at

`timeFrom` and unhibernate at `timeTo`. If `action: delete` then it will delete workloads at `timeFrom` and `timeTo`