```
    Scaling custom-deployment-1 down to 0
--> Success
Complete
```

If the custom deployment strategy process requires access to the OpenShift Container Platform API or the Kubernetes API the container that executes the strategy can use the service account token available inside the container for authentication.

3.3.5. Lifecycle hooks

The rolling and recreate strategies support *lifecycle hooks*, or deployment hooks, which allow behavior to be injected into the deployment process at predefined points within the strategy:

Example pre lifecycle hook

```
pre:
  failurePolicy: Abort
  execNewPod: {} 1
```

1 **execNewPod** is a pod-based lifecycle hook.

Every hook has a *failure policy*, which defines the action the strategy should take when a hook failure is encountered:

Abort	The deployment process will be considered a failure if the hook fails.
Retry	The hook execution should be retried until it succeeds.
Ignore	Any hook failure should be ignored and the deployment should proceed.

Hooks have a type-specific field that describes how to execute the hook. Currently, pod-based hooks are the only supported hook type, specified by the **execNewPod** field.

Pod-based lifecycle hook

Pod-based lifecycle hooks execute hook code in a new pod derived from the template in a **DeploymentConfig** object.

The following simplified example deployment uses the rolling strategy. Triggers and some other minor details are omitted for brevity:

```
kind: DeploymentConfig
apiVersion: v1
metadata:
  name: frontend
spec:
  template:
    metadata:
      labels:
        name: frontend
    spec:
      containers:
        - name: helloworld
```