

# Feature Deployment im Kop (OCP4)

- [Abstract](#)
  - [Hilfe-Skripts für Feature-Deployment-Prozess im Deployment Repository](#)
    - [deploy-feature.sh](#)
    - [clean-feature.sh](#)
  - [Jenkins Pipeline mit autoDeployFeatBranch Funktion](#)

## Abstract

Im OCP4 wird der **Deployment-Prozess** für neue Features oder Bugfixes angepasst. Anstatt Änderungen direkt in die Testumgebung ( wie Openshift in OCP3) zu deployen, werden die Deployment-Sets nun in ein spezielles [Deployment-Bitbucket-Repository](#) gepusht.

ArgoCD überwacht dieses Repository und synchronisiert die Änderungen mit der Ziel-Container-Plattform. Da dieser manuelle Prozess aufwändig sein kann, beschreibt dieses Wiki den automatisierten Vorgang mithilfe von Jenkins und Hilfskripten.

## Hilfe-Skripts für Feature-Deployment-Prozess im Deployment Repository

Im Deployment-Repository (elpa-elpa4) gibt es zwei Skript zur Automatisierung: [deploy-feature.sh](#) und [clean-feature.sh](#).

### deploy-feature.sh

Dieses Skript ist dafür zuständig, eine neue Feature-Umgebung im [dev](#)-Verzeichnis des Deployment-Repositories vorzubereiten.

1. **Parameter:** Es erwartet den Service-Namen, ein Suffix für den Service, die Jira-Ticketnummer und den vollständigen Image-Namen als Eingabe.
2. **Klonen:** Es kopiert die bestehende Konfiguration des Basis-Services (z.B. von [envs/dev/hint-service](#)) in ein neues Verzeichnis, das nach dem Muster ``SERVICE_NAME-JIRA_TICKET`` benannt ist (z.B. [envs/dev/hint-service-ELPA4-123](#)).
3. **Anpassen:**

- In der [kustomization.yaml](#) des neuen Verzeichnisses wird der [nameSuffix](#) angepasst, um das Jira-Ticket einzuschließen.
- Der Image-Tag wird mit dem übergebenen Image-Namen aktualisiert ( `kustomize edit set image app-image=<Image-Name>`).
- Istio-Konfigurationen (wie [VirtualService](#), [DestinationRule](#)) werden angepasst, um den neuen Feature-spezifischen Hostnamen zu verwenden.

4. **Registrieren:** Das neue Feature-Verzeichnis wird zur [kustomization.yaml](#) im [dev](#)-Verzeichnis hinzugefügt, damit Kustomize es erkennt.

### clean-feature.sh

Dieses Skript räumt eine Feature-Umgebung wieder auf.

1. **Parameter:** Es erwartet den Service-Namen und die Jira-Ticketnummer.
2. **Entfernen aus Ressourcen:** Der Eintrag für das Feature-Verzeichnis wird aus der [kustomization.yaml](#) im [dev](#)-Verzeichnis entfernt.
3. **Verzeichnis löschen:** Das gesamte Verzeichnis der Feature-Umgebung (z.B. [envs/dev/hint-service-ELPA4-123](#)) wird gelöscht

## Jenkins Pipeline mit autoDeployFeatBranch Funktion

In der [Jenkins-Shared-Library](#) (in [elpa-copsi.groovy](#)) gibt es die Funktion [autoDeployFeatBranch](#). Diese Funktion automatisiert den Prozess, um Änderungen aus einem Feature-Branch in das Deployment-Repository zu pushen.

1. **Parameter:** Die Funktion erwartet den Service-Namen (z.B. "hint-service") und den Image-Namen (inklusive Tag).
2. **Branch-Prüfung:** Sie prüft, ob der aktuelle Git-Branch einem gültigen Muster folgt (z.B. ELPA4-123-...). Wenn nicht, wird ein Fehler ausgegeben.
3. **Pull Request erstellen ( [si\\_copsi.createChangeAsPullRequest](#) ):**
  - Es wird ein neuer Branch im Deployment-Repository ([SDASVCDEPLOY/elpa-elpa4](#)) erstellt (z.B. [autodeploy/elpa4-123-job-XYZ](#)).
  - Innerhalb dieses Branches werden die Skripts [clean-feature.sh](#) und [deploy-feature.sh](#) ausgeführt, um die Kustomize-Konfiguration für das Feature zu erstellen/aktualisieren.
  - Die Änderungen (im [envs/dev](#)-Verzeichnis) werden committet.
  - Ein Pull Request von diesem neuen Branch zum ``nop``-Branch des Deployment-Repositories wird erstellt. Die Funktion gibt die ID des Pull Requests zurück.
4. **Auf Merge warten ( [si\\_copsi.waitForMergeChecksAndMerge](#) ):**
  - Jenkins wartet darauf, dass die Merge-Checks für den erstellten Pull Request erfolgreich sind.
  - Sobald die Checks erfolgreich sind, wird der Pull Request automatisch gemerged.
  - Der Source-Branch ([autodeploy/...](#)) wird danach optional gelöscht.
  - Die Funktion gibt zurück, ob der Merge erfolgreich war.

Nachdem der Pull Request im Deployment-Repository gemerged wurde, erkennt Argo-CD die neuen Kustomize-Manifeste und startet das Deployment des Feature-Stands in der entsprechenden Testumgebung. Nachdem der Pull Request im Deployment-Repository gemerged wurde, erkennt Argo-CD die neuen Kustomize-Manifeste und startet das Deployment des Feature-Stands in der entsprechenden Testumgebung.