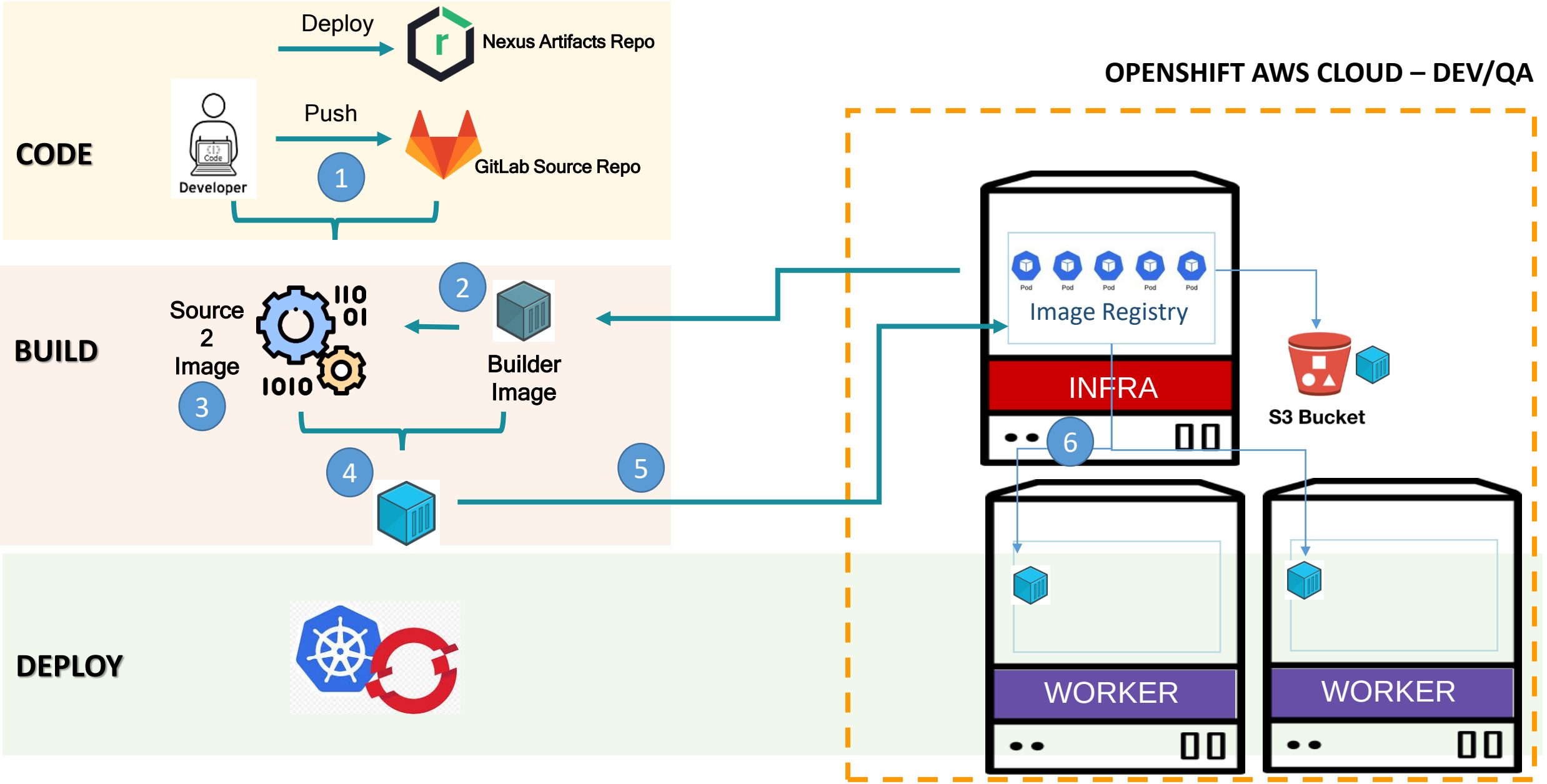
Crear Aplicación en Openshift
Guía Básica

Arquitectura TI

Elaborado por: *Kléber Franco Calderón*

Proceso Source-To-Image (S2I)



Pasos previos al S2I

- Esta guía aplica para el escenario en que Banco Bolivariano es propietario del fuente del microservicio, el proceso source-to-image requiere del fuente para ejecutar el la etapa de BUILD que consiste en generar el artefacto del microservicio.

- Antes de iniciar con el proceso de S2I es necesario lo siguiente:

a. El fuente del microservicio debe estar cargado en el repositorio GITLAB (https://gitlab.apps.ocpmsbbdevqa.bolivariano.fin.ec/).

b. Las dependencias que requiere el microservicio y que no se encuentran en el repositorio MAVEN CENTRAL (https://repo.maven.apache.org/maven2/), deben previamente cargarse en nuestro repositorio NEXUS (https://nexus.apps.ocpmsbbdevqa.bolivariano.fin.ec/). Dependencias como clientes de servicios, drivers de jdbc que no estén en MAVEN CENTRAL.

c. Debe existir un proyecto o namespace creado en Openshift, dentro de este proyecto se crearán los recursos necesarios para que el proceso de Source-To-Image se ejecute y los recursos que requiere el microservicio para funcionar.

Pasos previos a crear la Aplicación

1. Conectarse al clúster de Openshift por línea de comandos tal como lo detalla el manual “[TI-ARQ-MAN]Ambiente Desarrollo Openshift - CLI.pdf”

2. Ubicarse en el proyecto sobre el cual se creará la nueva aplicación:

```
oc project bb-dev-bmovil
```

3. Se debe crear, sólo en caso de que no exista, un recurso tipo SECRET llamado “git-secret” que contenga las credenciales para poder conectarse a GITLAB

```
oc create secret generic git-secret --from-literal="username=ocp-reader" --from-literal="password=op3nsh1ft"
```

Crear Recursos

- El pod del microservicio necesita de ciertos recursos externalizados para desplegarse de manera exitosa como por ejemplo el archivo de configuraciones application.yml y el almacén de certificados para esto se deben crear recursos config map y secret respectivamente.

1. Crear config map para el archivo de propiedades “application.yml”:

```
oc create cm usuarioms-application-yml --from-file=application.yml
```

2. Crear secret para el almacén de certificados “identity”, contiene únicamente el certificado tipo Servidor para exponer el servicio con HTTPS, esto se hace una sola vez por namespace/proyecto:

```
oc create secret generic identity-jks --from-file=identity.jks=identity.jks
```

3. Crear secret para el almacén de certificados “trust”, contiene certificados tipo Cliente para consumir servicios de terceros, esto se hace una sola vez por namespace/proyecto :

```
oc create secret generic truststore-jks --from-file=truststore.jks=truststore.jks
```

Crear Aplicación

1. Crear la nueva aplicación:

```
oc new-app --as-deployment-config --build-env
MAVEN_MIRROR_URL=https://nexus.apps.ocpmsbbdevqa.bolivariano.fin.ec/repository/maven-public --build-
  env MAVEN_OPTS=-Dmaven.wagon.http.ssl.insecure=true --name usuarioms -i bb-images/bb-openjdk-8 -e
            TZ=America/Guayaquil -e LOGGING_LEVEL_ROOT=INFO --source-secret=git-secret
             https://gitlab.apps.ocpmsbbdevqa.bolivariano.fin.ec/bmovil/usuarioms.git
```

Asociar Recursos

- Los recursos creados deben asociarse con el deployment config mediante volúmenes:

1. Asociar config map para el archivo de propiedades “application.yml”:

```
oc set volume dc/usuarioms --add --name=application-yml --type configmap --configmap-name=usuarioms-application-yml --mount-path=/deployments/application.yml --sub-path=application.yml
```

2. Asociar secrets para almacenes de certificados “identity” y “trust:

```
oc set volume dc/usuarioms --add --name=identity-jks --type secret --secret-name=identity-jks --mount-path=/secret/identity.jks --sub-path=identity.jks

oc set volume dc/usuarioms --add --name=truststore-jks --type secret --secret-name=truststore-jks --mount-path=/secret/truststore.jks --sub-path=truststore.jks
```

3. En caso de modificar el config map o algún secret, se debe ejecutar rollout del deployment config para que la etapa de DEPLOY se vuelva a ejecutar y validar que se despliegue satisfactoriamente.

Exponer servicio

- En el caso que aplique se debe exponer el servicio:

1. Crear la ruta de tipo “passthrough” para la aplicación creada:

```
oc create route passthrough --service usuarioms --port 8443
```

```
oc annotate route usuarioms --overwrite haproxy.router.openshift.io/balance=roundrobin
```

Siguientes pasos

- Los siguientes pasos:

1. Definir variables de Entorno en DeploymentConfig
2. Especificar Reserva y Límite de memoria del POD en DeploymentConfig
3. Configurar HealthChecks en DeploymentConfig
4. Especificar AntiAfinidad en DeploymentConfig para que los PODS no se creen en el mismo Worker