

Connect

¿JBoss es legacy?

Qué va... lo acabo de ver desplegarse en OpenShift como si nada. De cero a GitOps en un plis plas

Carlos Vicens
AppDev SSA EMEA

José A. González Prada
OpenShift SSA

Introducción

JBoss Cowboys vs Space

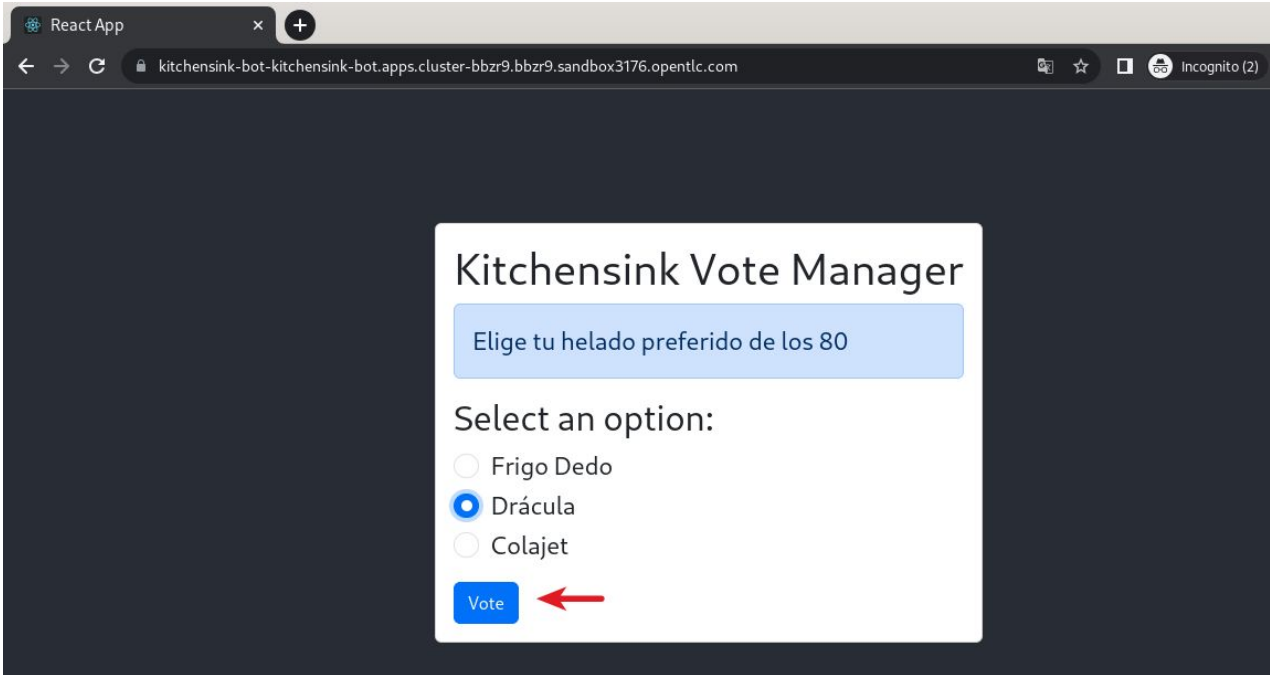
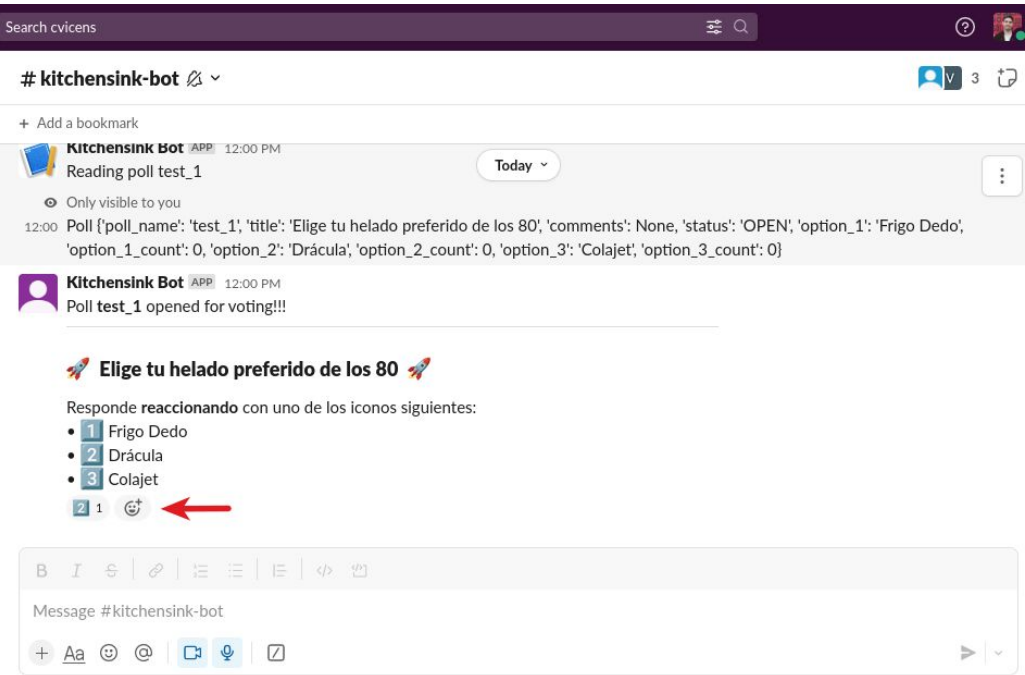
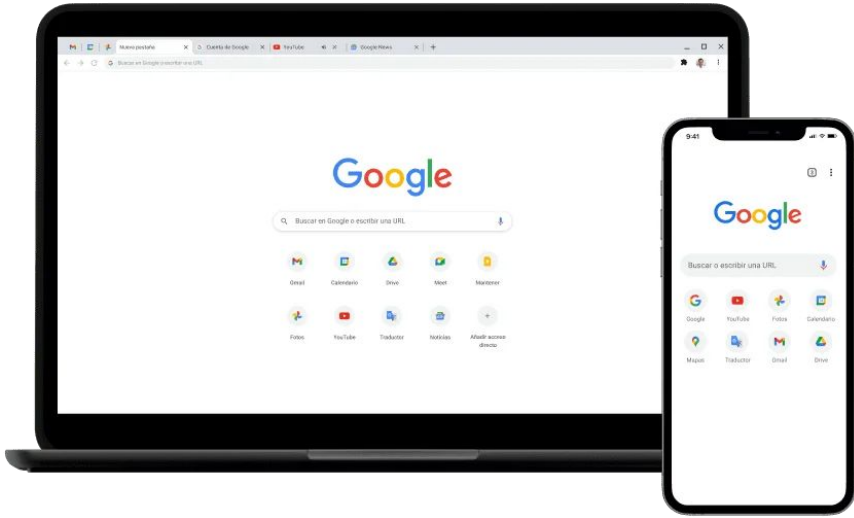
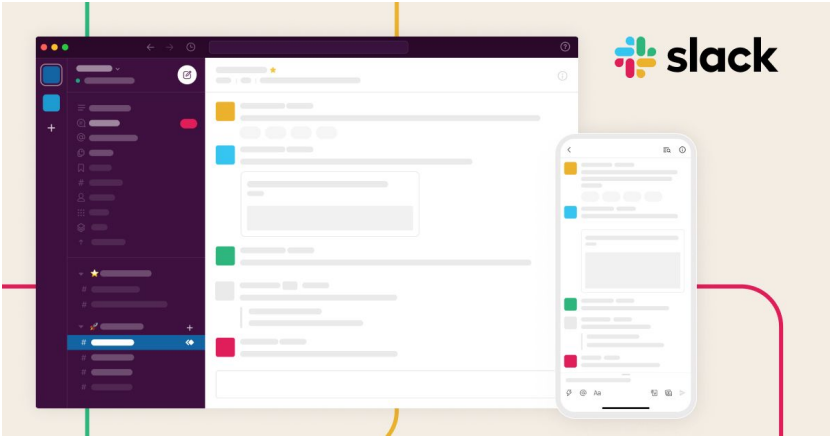
Cowboys



***Legacy proviene de legado... No de viejo decrépito.
Un respeto!***



CONTROL DE MISIÓN

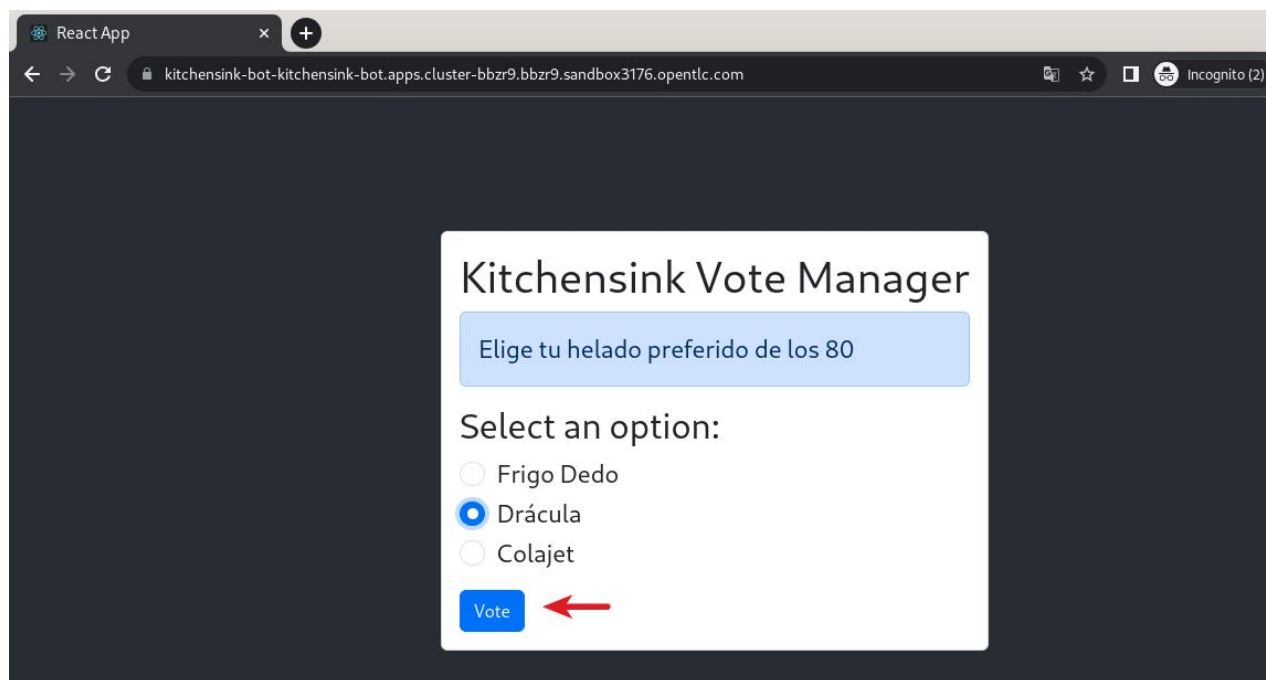




Hagamos una prueba!



<https://red.ht/jboss-cowboys>



Punto de partida

NUESTRA
CARGA ÚTIL

Aplicaciones JBoss EAP (Jakarta EE) **que soportan lógica fundamental** para nuestro negocio **pero que:**

- **Nos cuesta desplegar** nuevas versiones
- **Nos cuesta subir de versión** JBoss EAP, SO, ...

Esto reduce nuestra velocidad de despliegue,
limita nuestra innovación

¿A dónde vamos?

Misión STI-400

La misión **STI-400** tiene por **objetivo** el **simplificar el ciclo de vida de esas aplicaciones JBoss EAP**.

Despliegues más rápidos y seguros, más a menudo

Por cierto nuestra se llama **STI-400** en lugar de la denominación regular: **STS-### ¿por qué será?**



Vamos!





PLAN DE MISIÓN

Vista completa: ON-PREMISE => OPENSIFT

PUNTO PARTIDA

Construcción y Pruebas



Categorizar aplicaciones por tipo, topología, etc.
Migración a JBoss
Externalizar el contexto, ficheros de propiedades, certificados, tokens, etc.
Adaptar de configuración de JBoss con scripts de jboss-cli.sh

Despliegue manual con STI sobre OpenShift

Despliegue automático con GitOps

Despliegue en dos namespaces con GitOps

Despliegue GitOps en dos overlays con kustomize

Despliegue GitOps utilizando helm

Despliegue GitOps de pipelines CI/CD

Despliegue GitOps mixto helm+kustomize

Versiones y Monitorización

Actualización de versión 7.2 a 7.4

Actualización de versión 7.4 a 8.1

Actualización de versión 8.x a 9.x

DEMO HOY

Navegación y Llegada / Experimentación

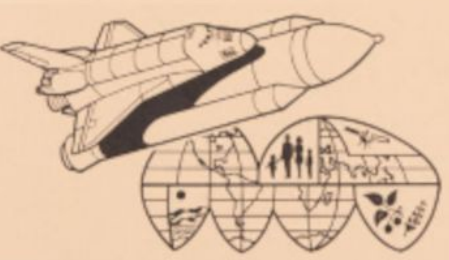
Mantenimiento rutinario (CI/CD Pipelines)

FUTURO

Operaciones extendidas

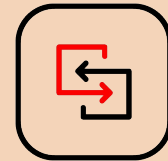
Construcción y Pruebas





PLAN DE MISIÓN 1/4 – Construcción y Pruebas

Construcción y Pruebas



Migration Toolkit for Applications

Categorizar aplicaciones
por tipo, topología, etc.

Migración desde
p. ej. WLS a JBOSS

Externalizar el contexto,
ficheros de propiedades,
certificados, tokens, etc.

Adaptar la configuración de
JBOSS con **scripts** de
jboss-cli.sh

Se pretende **definir**
unas **plantillas** para
que **migrar por**
grupos

Se **detectan los puntos a**
resolver (Story Points) y
se **apuntan soluciones**
que servirán para el
grupo de aplicaciones

El **contexto se define con**
variables de entorno y/o
ficheros que residen en
ConfigMaps y/o Secrets

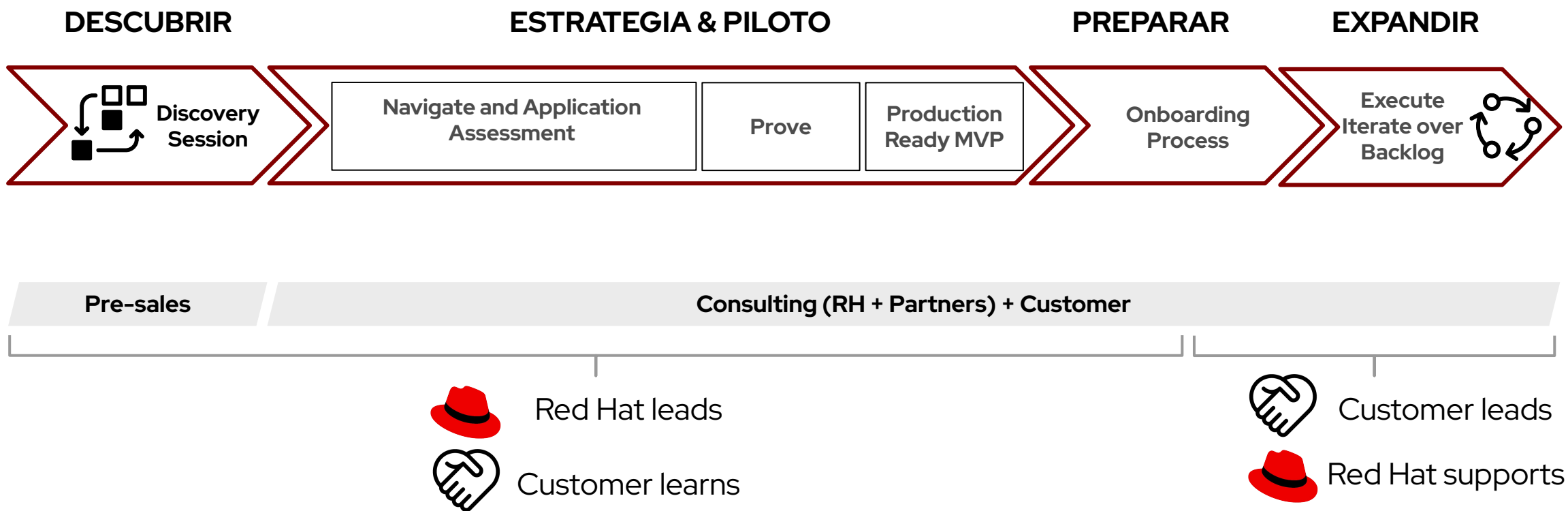
S2I permite la ejecución de
scripts de JBOSS en la
construcción y posterior
ejecución de la imagen

Ya se han ejecutado en tierra



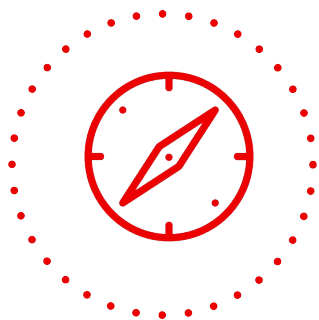
Application Modernization and Migration (AMM)

Nuestro enfoque y metodología





Migration Toolkit for Applications (MTA)



Guía de Migración



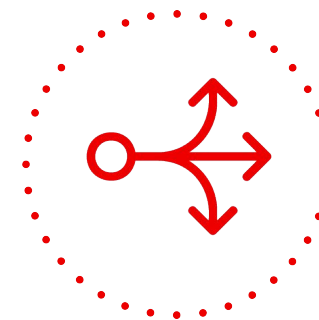
**Facilitar la adopción de
OpenShift**



Recopilar Información



Reducir Riesgos



**Ampliar el ámbito de
alcance**



**Aportar valor en cada fase
de la adopción**



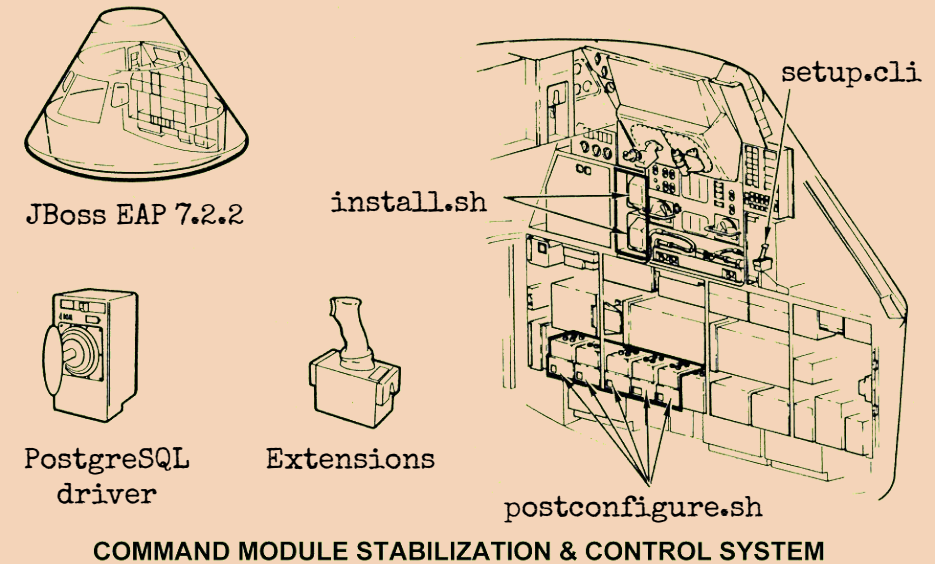
Ignición !



TELETIPO DE CONTROL DE MISIÓN - I

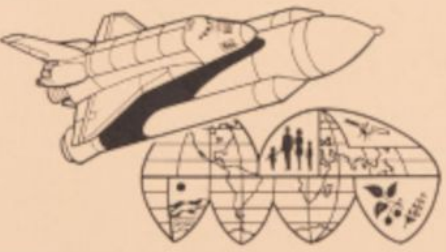
Recomendaciones para generar una imagen de aplicación JBoss EAP

1. Partir de un JBoss limpio de la versión necesaria
2. Aplicar cambios de configuración de forma automatizada
3. Utilizar comandos de `jboss.cli`
4. Construir la imagen con el código de la aplicación



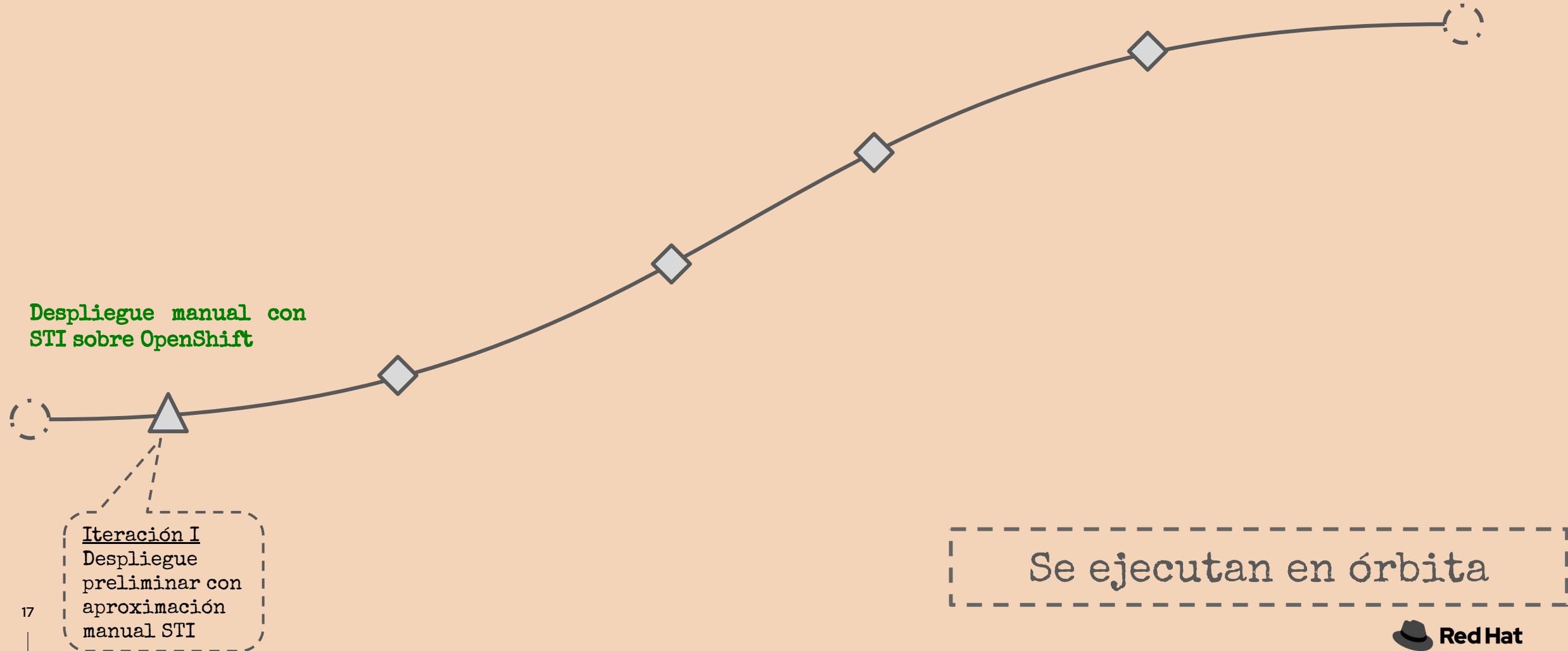
De este modo **simplificamos el proceso de actualización de JBoss EAP.**

Denominemos a esta implementación de las recomendaciones **STI** por Source To Image



PLAN DE MISIÓN 2/4 - Operaciones

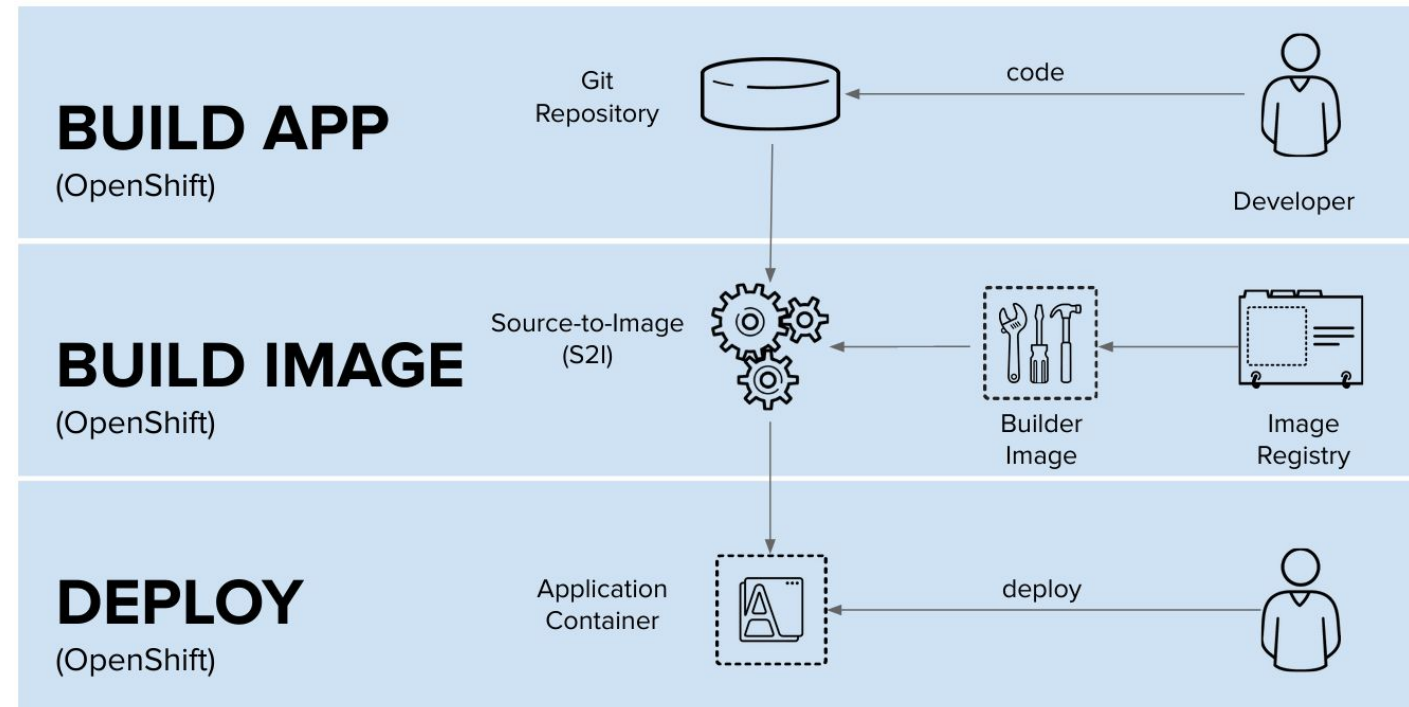
Navegación y Llegada $\triangle$ / Experimentación $\diamond$

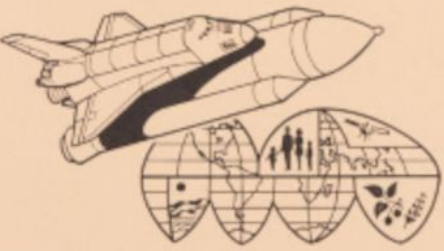


Despliegue de aplicaciones JBoss EAP con STI

Red Hat OpenShift STI es un **framework genérico**

1. Punto de origen => GIT
2. Partimos de una imagen con herramientas: JDK, JBOSS, javac, dotnet, etc.
3. Construimos una imagen con nuestra aplicación lista para ser ejecutada





A321HO
ZYYLBYL

URGENTE

=====

LANZAMIENTO - I

=====

JJ1032pcs 17 de octubre



Despliegue de aplicaciones JBoss EAP con STI

La imagen 'builder' STI contiene scripts que permiten:

1. **Construir** nuestra aplicación a partir de código y colocar el binario o similar en un path conocido $\Rightarrow$ **assemble**
2. **Ejecutar** la aplicación desde ese path conocido $\Rightarrow$ **run**



Despliegue de aplicaciones JBoss EAP con S2I

Ficheros ejecutados en construcción (assemble)

extensions/install.sh

```
#!/bin/bash
echo ">>>>>> Running install.sh <<<<<<"
injected_dir=$1
source /usr/local/s2i/install-common.sh
configure_drivers ${injected_dir}/driver-oracle.env
# configure_drivers ${injected_dir}/driver-postgresql.env

# copy any needed files into the target build.
cp -rf ${injected_dir} $JBoss_HOME/extensions
```

WATCH OUT!

extensions/driver-oracle.env

```
DRIVERS=ORACLE

ORACLE_DRIVER_NAME=oracle
ORACLE_DRIVER_MODULE=com.oracle
ORACLE_DRIVER_CLASS=oracle.jdbc.OracleDriver
```



Despliegue de aplicaciones JBoss EAP con S2I

Ficheros ejecutados en ejecución (run)

extensions/postconfigure.sh

```
#!/usr/bin/env bash
echo ">>>>> Executing postconfigure.sh <<<<<<<"
if [ "${SCRIPT_DEBUG}" = "true" ]; then
  set -x echo "Script debugging is enabled"
fi
```

WATCH OUT!

```
echo ">>>>> Executing setup.cli <<<<<<<"
```

```
$JBoss_HOME/bin/jboss-cli.sh --file=$JBoss_HOME/extensions/setup.cli
```

WATCH OUT!

extensions/setup.cli

```
embed-server --std-out=echo --server-config=standalone-openshift.xml
```

```
/subsystem=datasources/data-source=kitchensink:add( \
  jndi-name="java:jboss/jdbc/kitchensink", \
  connection-url="jdbc:postgresql://${env.DB_HOST}:${env.DB_PORT:5432}/${env.DB_NAME:kitchensink}", \
  driver-name=postgresql, \
  user-name=${env.DB_USERNAME},password=${env.DB_PASSWORD}, \
  validate-on-match=true, \
  valid-connection-checker-class-name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker", \
  exception-sorter-class-name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter" \
)
```

Datasource kitchensink

```
if (outcome == success) of /subsystem=datasources/jdbc-driver=mysql:read-attribute(name=driver-name)
  /subsystem=datasources/jdbc-driver=mysql:remove
end-if
```

Eliminar datasource

```
quit
```



A321H0
ZYYLBYL

INFORME

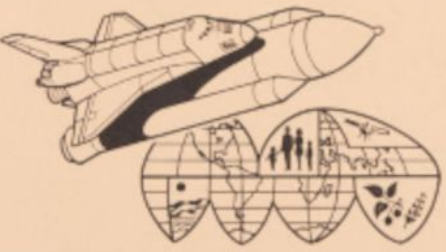
=====

Descriptores desplegados **manualmente** con STI

- > Imagen construida con STI: **nominal**
- > Ajustes en construcción: **nominal**
 - ~ drivers, módulos
- > Ajuste JBoss 7.2 en ejecución: **nominal**
 - ~ DataSource creado
 - ~ Driver MySQL eliminado
- > Ejecución de aplicación: **nominal**
- (*) **No es automático, ¿qué pasa con las actualizaciones, entornos, etc.?**

=====

JJ1032pcs 17 de octubre

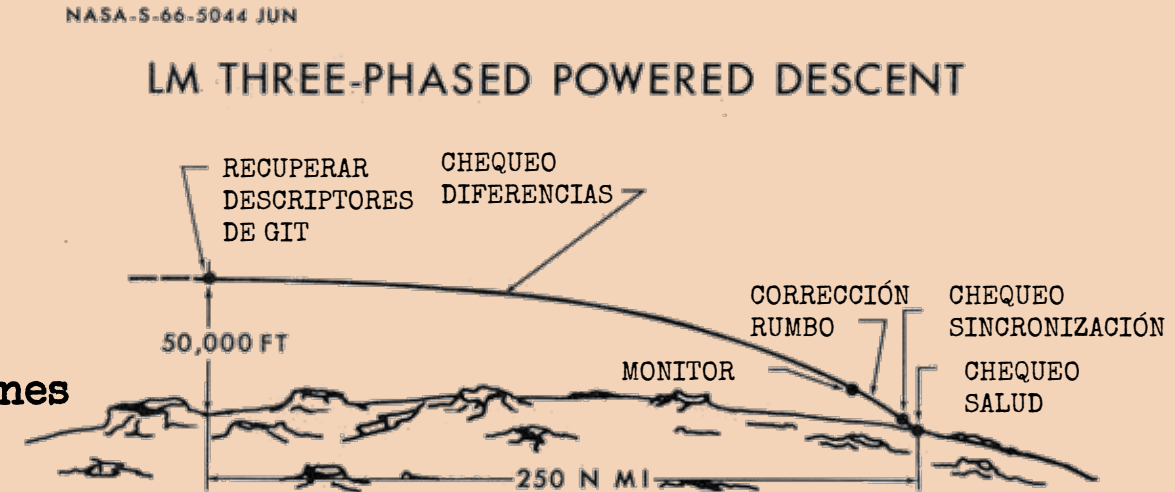


TELETIPO DE CONTROL DE MISIÓN - II

Recomendaciones para un sistema de despliegue automático basado en git o **GitOps**

Recomendaciones:

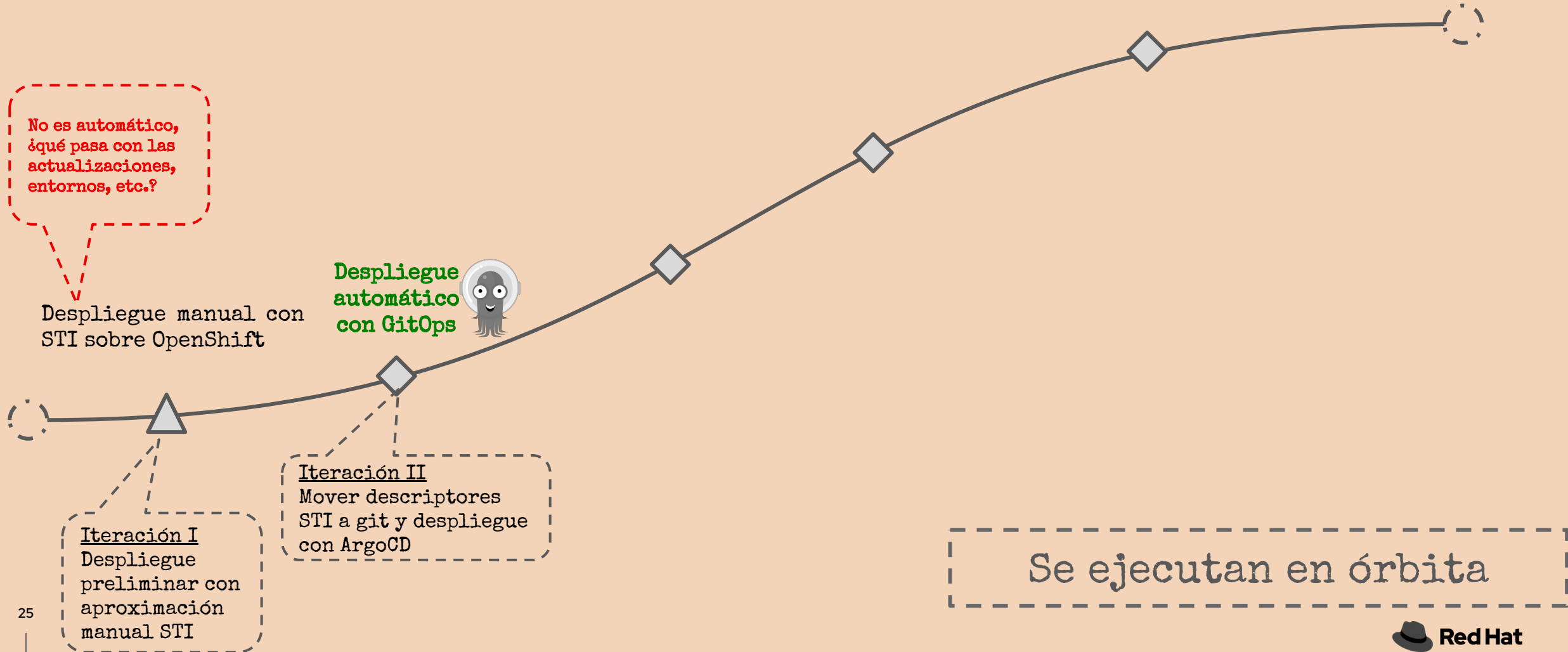
1. Configuración de despliegue **centralizada** en Git
2. Debe ser posible **definir un aplicación** en base a:
 - a. **Origen:** dónde está la configuración
 - b. **Destino:** dónde de despliega
3. Permite **definir conjuntos de aplicaciones**
4. El sistema **debe corregir el rumbo** si hay desviaciones



Probamos una implementación de estas recomendaciones, el proyecto ArgoCD

PLAN DE MISIÓN 2/4 - Operaciones

Navegación y Llegada $\triangle$ / Experimentación $\diamond$





Objeto clave de ArgoCD: Application

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: kitchensink-basic-app
  namespace: openshift-gitops
  labels:
    kitchensink-root-app: 'true'
Spec:
```

```
destination:
  name: in-cluster
  namespace: argo-userX
```

```
source:
  path: basic
  repoURL: 'https://github.com/atarazana/kitchensink-conf'
  targetRevision: main
```

```
project: default
```

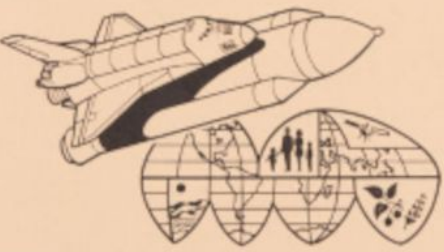
```
syncPolicy:
  automated:
    selfHeal: true
```

Destino, cluster y namespace donde desplegar

Fuente de verdad, de donde se obtienen los descriptores de despliegue

Proyecto, permite definir restricciones

Parametrización de la **sincronización**



A321HO
ZYYLBYL

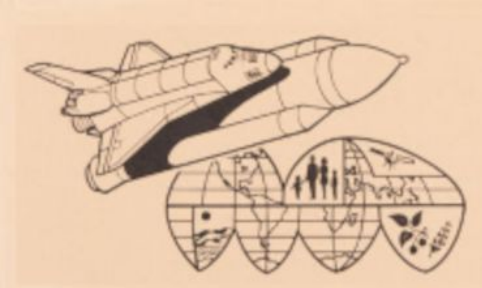
URGENTE

=====

LANZAMIENTO - II

=====

JJ1032pcs 17 de octubre



A321HO
ZYYLBYL

INFORME

=====

Mismos descriptores STI de etapa previa
desplegados con **ArgoCD**

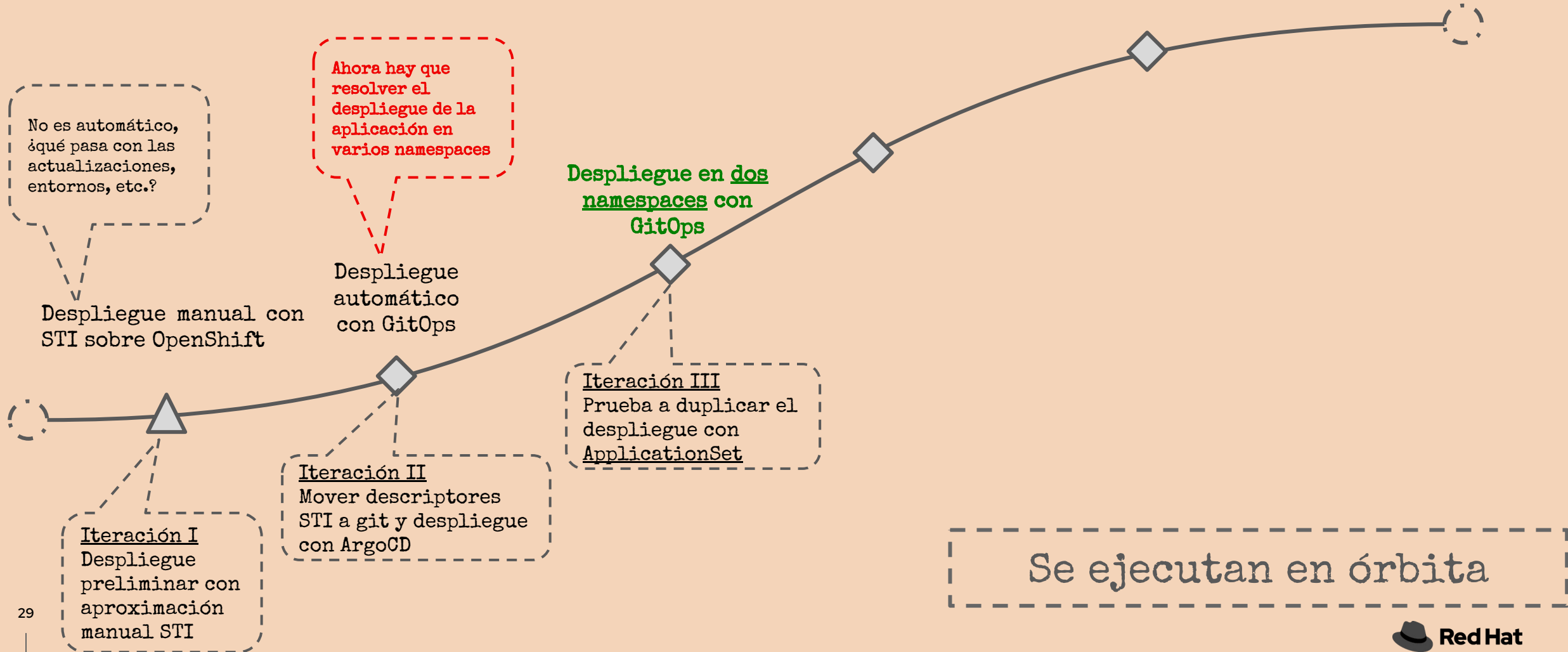
- > Construcción imagen: **nominal**
- > Ejecución de aplicación: **nominal**
- + **Mejoras en corrección de rumbo**
- + **Mejoras en auditoría**
- **No desplegamos en varios namespaces**

=====

JJ1032pcs 17 de octubre

PLAN DE MISIÓN 2/4 - Operaciones

Navegación y Llegada $\triangle$ / Experimentación $\diamond$



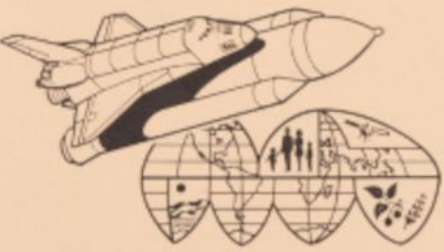


Para lo cual vamos a utilizar un ApplicationSet

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: kitchensink-basic
  namespace: openshift-gitops
  labels:
    argocd-root-app: "true"
spec:
  generators:
    - list:
        elements:
          - env: appset-a-userX
          - env: appset-b-userX
  template:
    metadata:
      name: kitchensink-basic-app-{{ env }}
      namespace: openshift-gitops
      labels:
        kitchensink-root-app: "true"
    ...
```

Generadores, definiciones que permiten generar objetos definidos con la plantilla `template`

Plantilla que emplea el generador como modelo para generar objetos de tipo Application



A321HO
ZYYLBYL

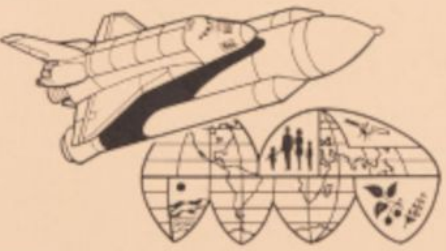
URGENTE

=====

LANZAMIENTO - III

=====

JJ1032pcs 17 de octubre



A321HO
ZYYLBYL

INFORME

=====

Descriptores STI desplegados con **ArgoCD**

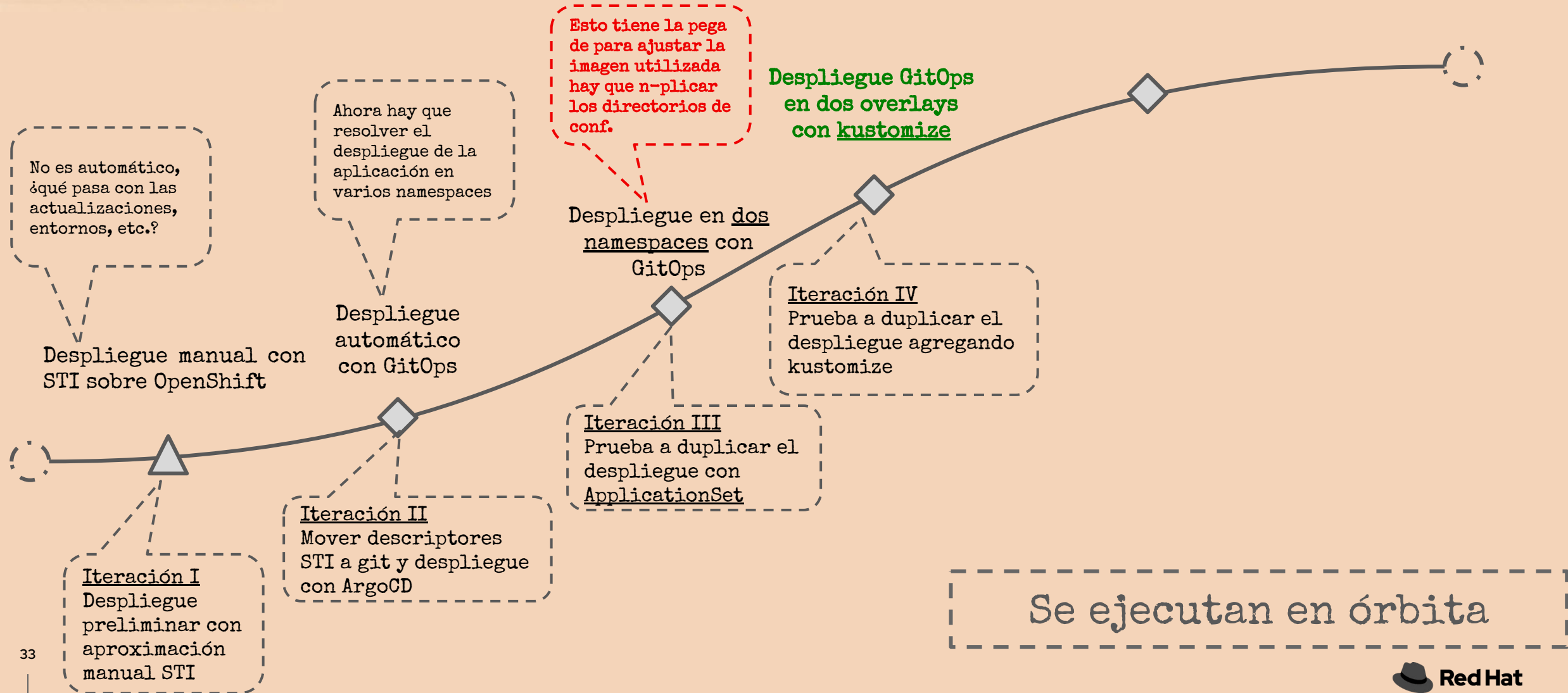
- > Despliegue en 2 namespaces: **nominal**
- > Construcción imagen: **nominal**
- > Ejecución de aplicación: **nominal**
- + **Mejoras en corrección de rumbo**
- + **Mejoras en auditoría**
- + **Despliegue de grupos de aplicaciones**
- **Habría que n-plicar los directorios de conf.**

=====

JJ1032pcs 17 de octubre

PLAN DE MISIÓN 2/4 - Operaciones

Navegación y Llegada $\triangle$ / Experimentación $\diamond$





PLUGINS

Overlay

K PATCH

It was a wet, cold November day. I woke up to the d *sound of thundering hoofbeats down the street,* rs.



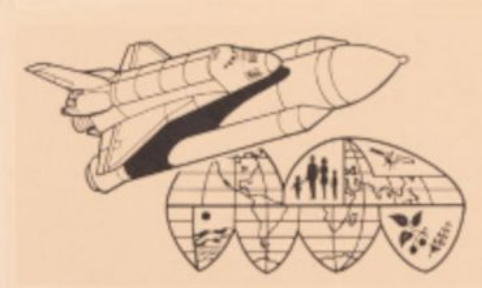
DIFFERENT ENVIRONMENTS, DEV, STAGING, QA, ...



```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
...
patchesJson6902:
- target:
    group: apps
    version: v1
    kind: Deployment
    name: kitchensink
    path: patch/deployment_patch.yml
```



```
- op: replace
  path: /spec/replicas
  value: 1
```



A321HO
ZYYLBYL

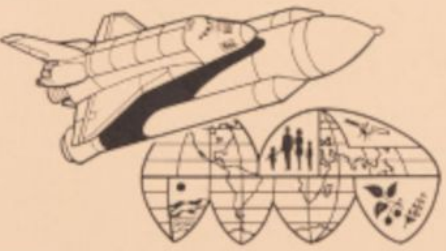
URGENTE

=====

LANZAMIENTO - IV

=====

JJ1032pcs 17 de octubre



A321HO
ZYYLBYL

INFORME

=====

Despliegue con **ArgoCD + kustomize**

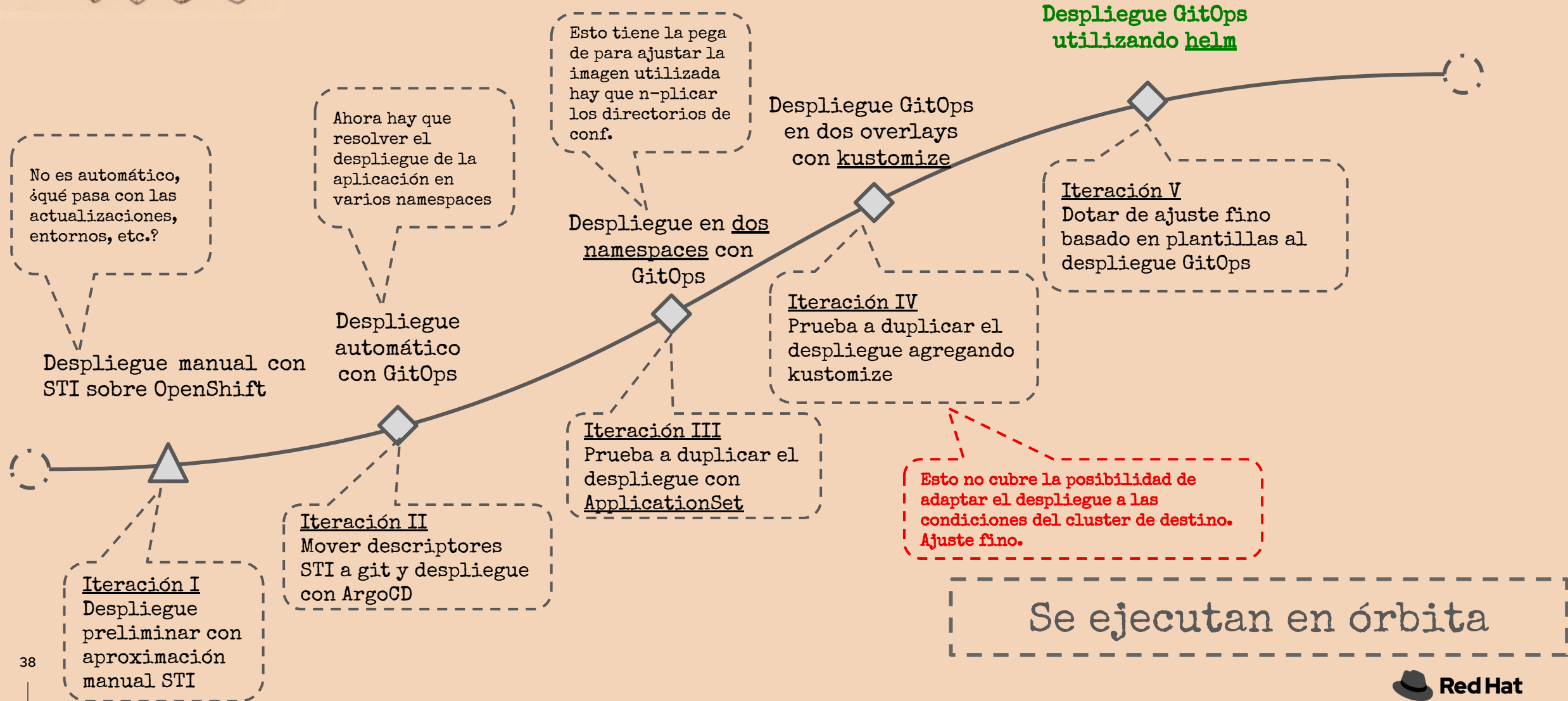
- > Despliegue en 2 entornos: **nominal**
- > Construcción imagen: **nominal**
- > Ejecución de aplicación: **nominal**
- + **Ajuste de etiquetas y namespace**
- + **Parcheo de objetos por entorno**
- **No se adapta el despliegue a las condiciones del cluster de destino.**

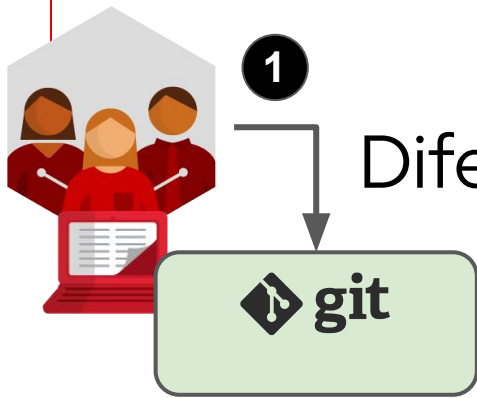
=====

JJ1032pcs 17 de octubre

PLAN DE MISIÓN 2/4 - Operaciones

Navegación y Llegada / Experimentación





Diferencia entre Template/Helm y Overlay/Kustomize

GIT IS THE SOURCE OF TRUTH
WE WANT TO DEPLOY ON DIFFERENT PLATFORMS
AND DIFFERENT ENVIRONMENTS



PLUGINS

Template



DIFFERENT PLATFORMS, AZURE, AWS, BARE METAL

HELM DETAILS

It was a wet , cold November day. I woke up to the sticky smell of ostrich roasting in the dungeon downstairs.
adjective adjective type of bird room in a house

Overlay



DIFFERENT ENVIRONMENTS, DEV, STAGING, QA, ...

K PATCH

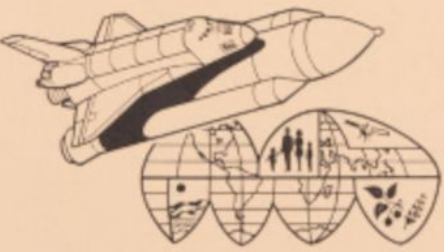
It was a wet, cold November day. I woke up to the d sound of thundering hoofbeats down the street, rs.



Despliegue GitOps en dos overlays con kustomize

```
spec:
...
  {{ if or (gt .Values.ocpVersion "4.11") (eq .Values.ocpVersion "4.11") }}
  securityContext:
    capabilities:
      add:
        - SETFCAP
  {{ end }}
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  storageClassName: |
    {{- if eq .Values.cloudVendor "aws" }}
    {{ .Values.storageClassAWS }}
    {{- else if eq .Values.cloudVendor "gcp" }}
    {{ .Values.storageClassGCP }}
    {{- else }}
    default-storage-class
    {{- end }}
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```



A321HO
ZYYLBYL

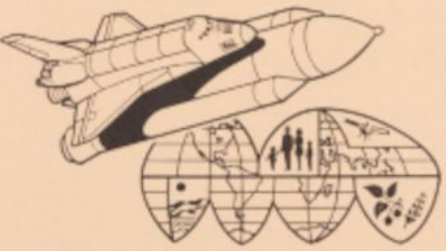
URGENTE

=====

LANZAMIENTO – V

=====

JJ1032pcs 17 de octubre



A321HO
ZYYLBYL

INFORME

=====

Despliegue con **ArgoCD + helm**

- > Despliegue basado en helm: **nominal**
- > Construcción imagen: **nominal**
- > Ejecución de aplicación: **nominal**
- + **Ajuste basado en valores y plantillas**
- (*) **No es fácil desplegar en diferentes overlays/entornos utilizando sólo helm o sólo kustomize.**

=====

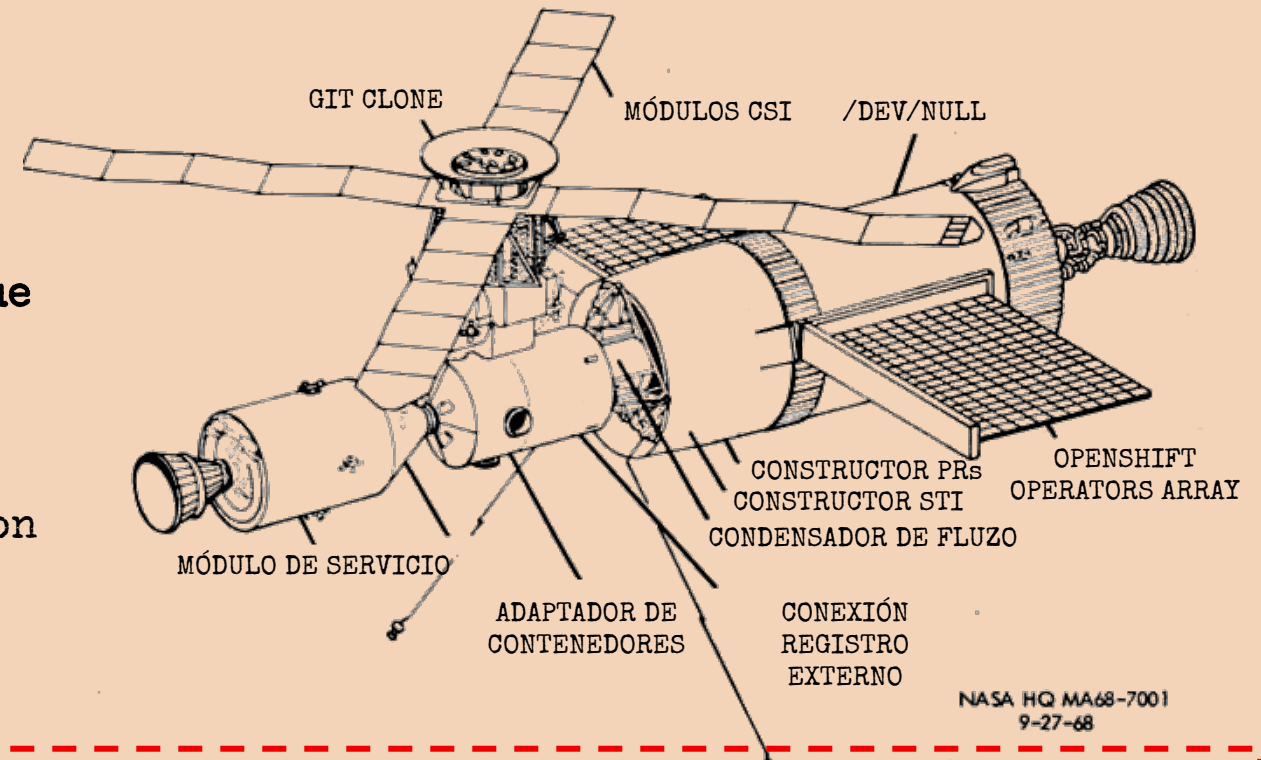
JJ1032pcs 17 de octubre

TELETIPO DE CONTROL DE MISIÓN - III

Recomendaciones para tareas rutinarias de mantenimiento mediante pipelines

Recomendaciones:

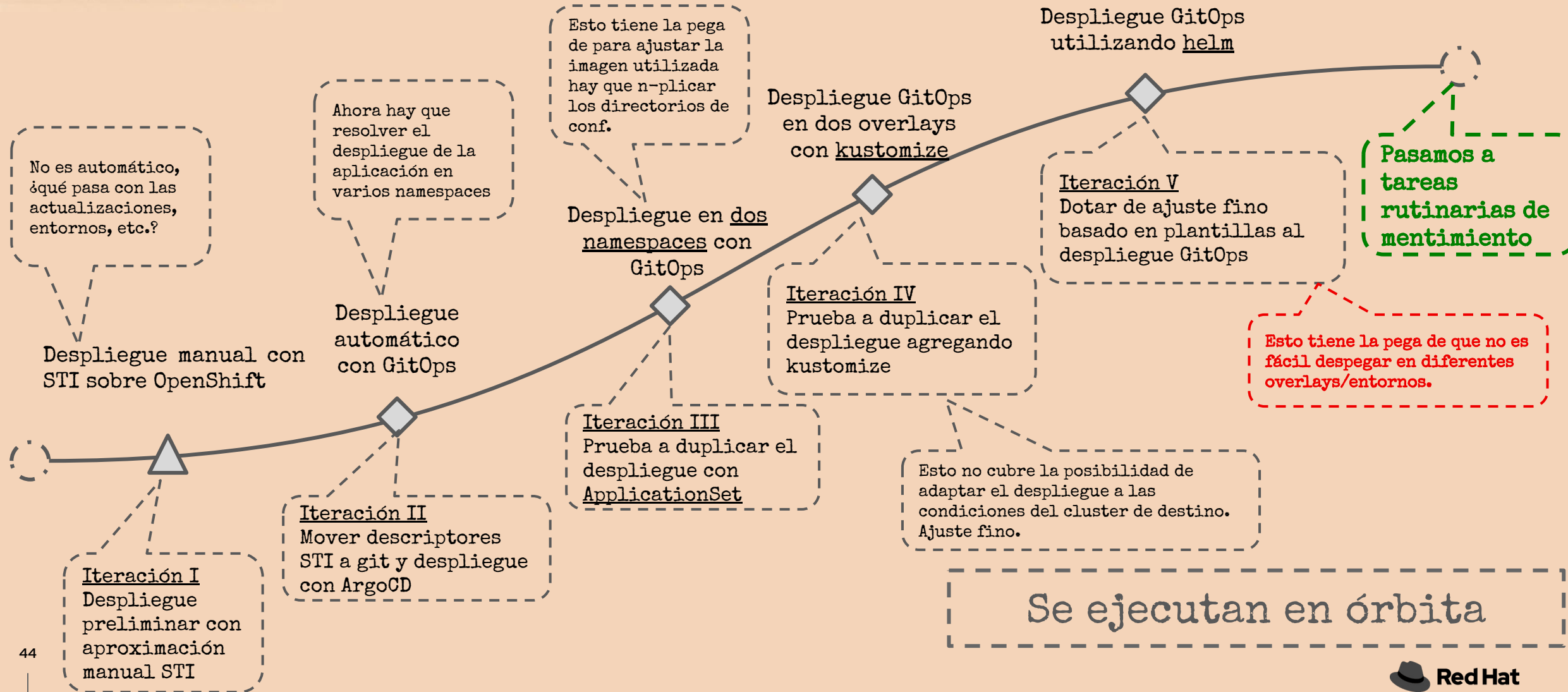
1. Los pipelines de integración y despliegue continuo deben ser nativos k8s
2. Desplegables en órbita mediante operador
3. Debe poderse desplegar con Argo
4. El paso entre entornos debe ser coherente con GitOps => Pull Request

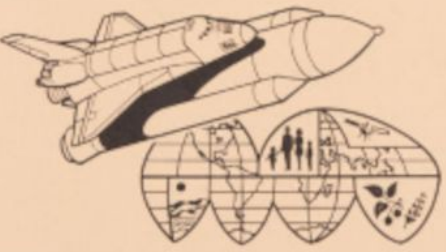


Probamos una implementación de estas recomendaciones, el proyecto Tekton

PLAN DE MISIÓN 2/4 - Operaciones

Navegación y Llegada $\triangle$ / Experimentación $\diamond$





PLAN DE MISIÓN 3/4 - Exploración

Mantenimiento rutinario (CI/CD Pipelines)

Versiones y
Monitorización

Despliegue
GitOps mixto
helm+kustomize

Iteración VII
Acciones rutinarias
en órbita

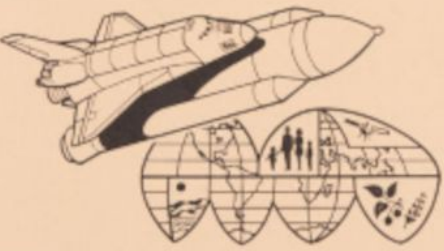
Despliegue GitOps de
pipelines CI/CD



Iteración VIb
Despliegue completo
con ajuste fino en
varios entornos

Iteración VIa
Despliegue de
pipelines CI/CD
con ArgoCD

Se ejecutan en órbita



A321HO
ZYYLBYL

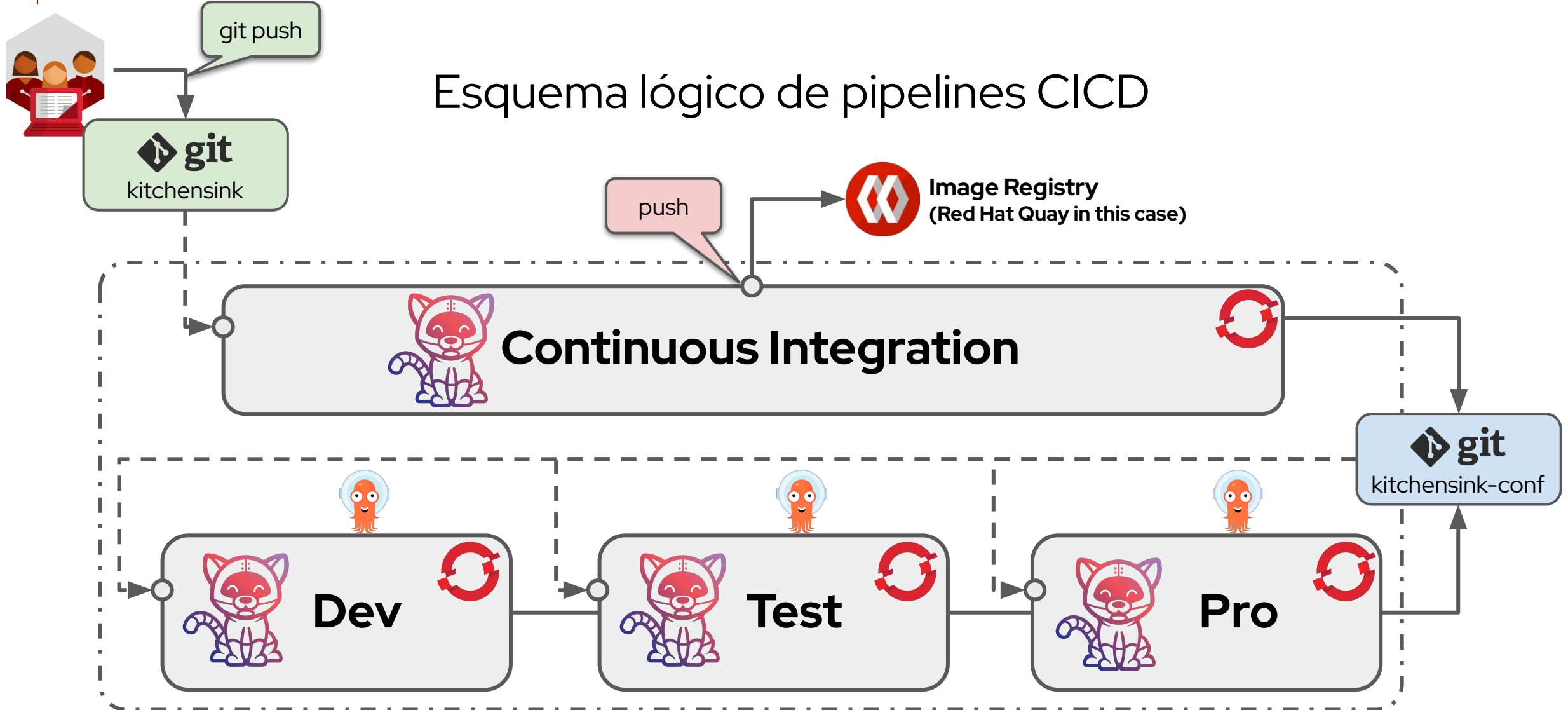
URGENTE

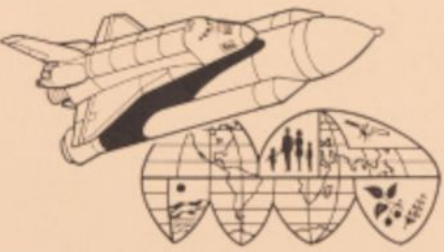
=====

LANZAMIENTO - VIa

=====

JJ1032pcs 17 de octubre





A321HO
ZYYLBYL

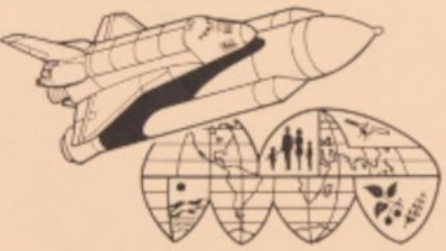
URGENTE

=====

LANZAMIENTO - VIb

=====

JJ1032pcs 17 de octubre



A321HO
ZYYLBYL

INFORME

=====

Despliegue con **ArgoCD + kustmized-helm**

- > Despliegue híbrido: **nominal**
- > Construcción imagen: **nominal**
- > Ejecución de aplicación: **nominal**
- > Pipelines CICD: **nominal**

- + **Ajuste basado en valores y plantillas**
- + **Ajuste de etiquetas y namespace**
- + **Parcheo de objetos por entorno**

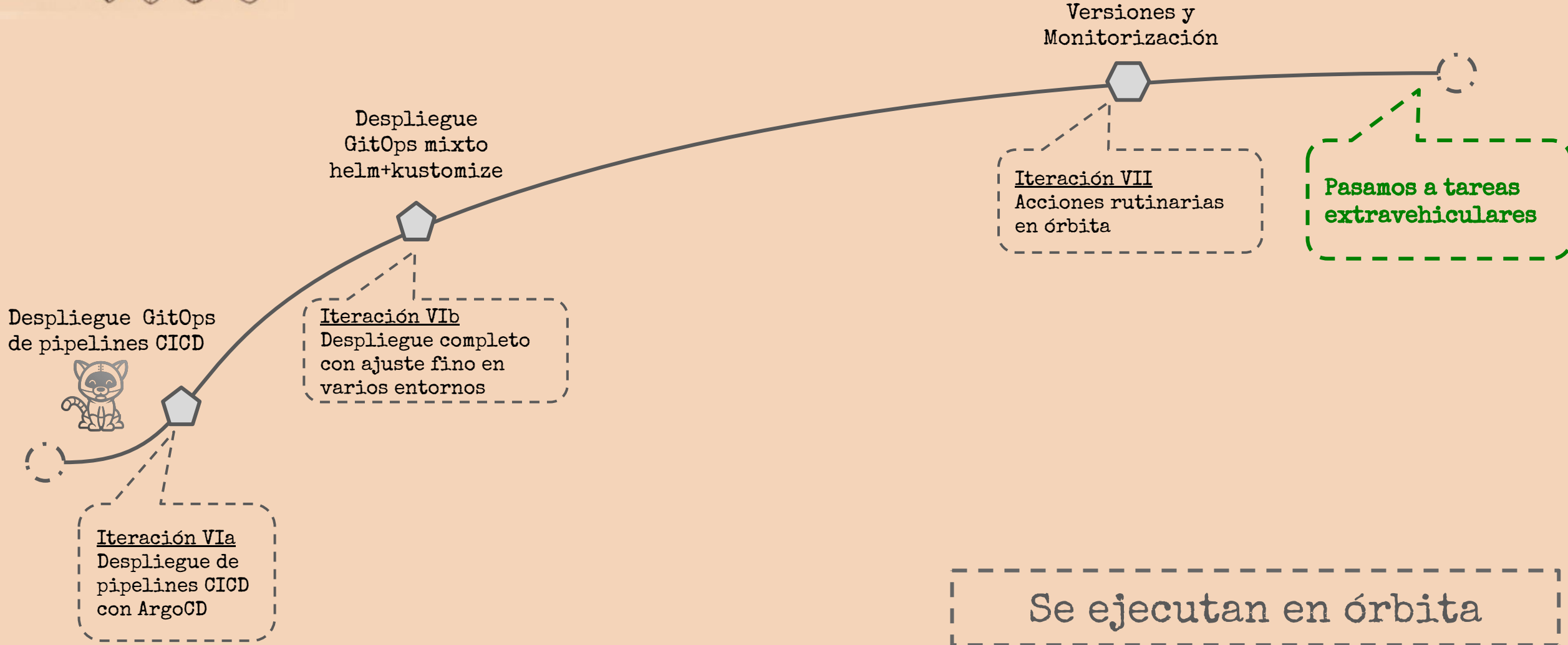
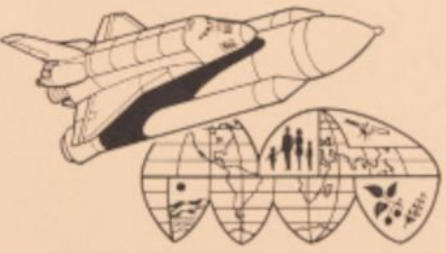
=====

JJ1032pcs 17 de octubre



PLAN DE MISIÓN 3/4 - Exploración

Mantenimiento rutinario (CI/CD Pipelines)



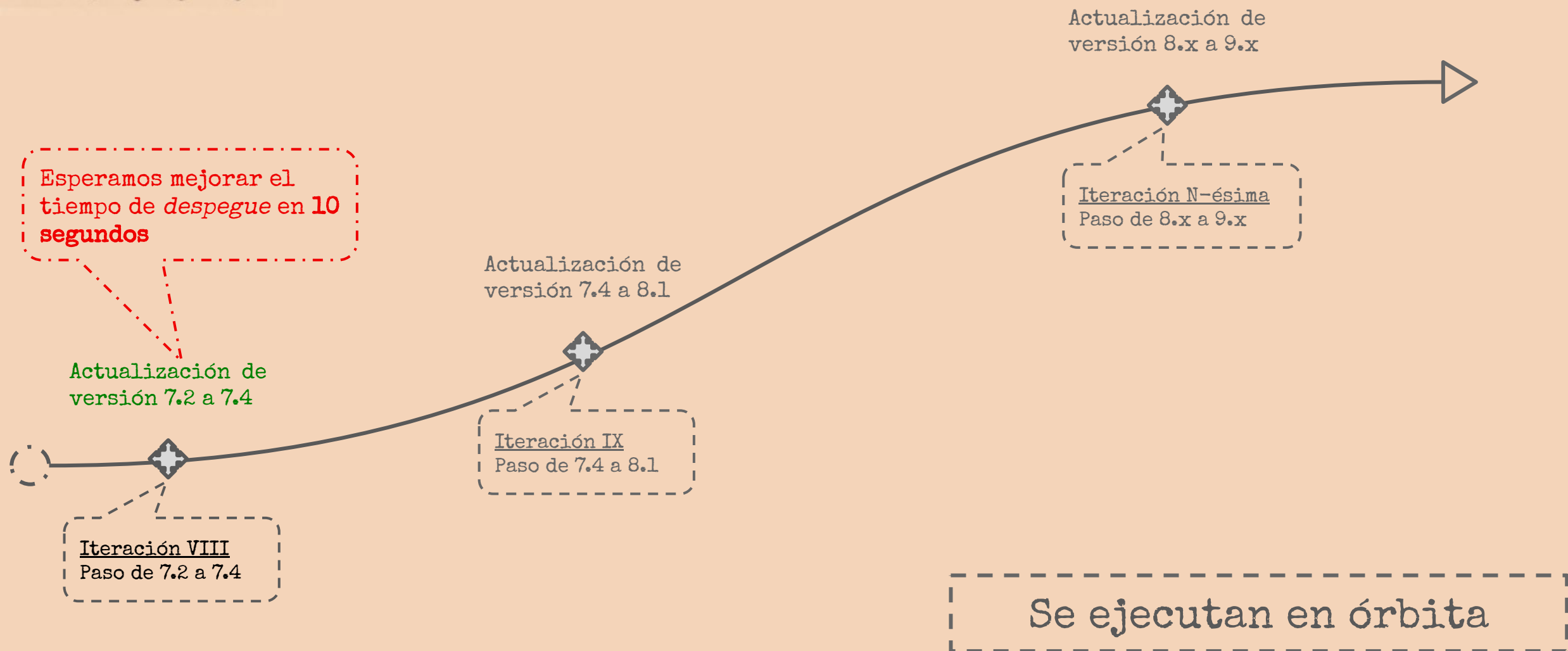
Actividades Extravehiculares!

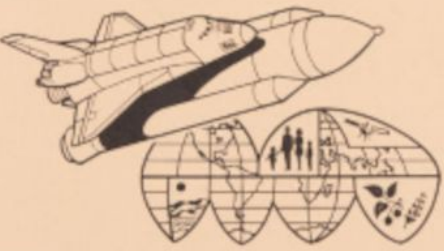




PLAN DE MISIÓN 4/4 - Operaciones Extendidas

Actividades Extravehiculares





A321HO
ZYYLBYL

URGENTE

=====

LANZAMIENTO - VII

=====

JJ1032pcs 17 de octubre



A321HO
ZYYLBYL

INFORME

=====

Despliegue con **ArgoCD + kustmized-helm**

- > Despliegue híbrido: **nominal**
- > Construcción imagen 7.4: **nominal**
- > Ejecución de aplicación: **nominal**
- > Pipelines CI/CD: **nominal**
- + **Ajuste basado en valores y plantillas**
- + **Ajuste de etiquetas y namespace**
- + **Parcheo de objetos por entorno**
- + **Reducción de tiempo**

=====

JJ1032pcs 17 de octubre

Touch down!



Resources

- [\[*\] Workshop Guide](#): Step-by-step guide with everything you need to know to perform the WorkAthon. ***It is highly recommended to review it prior to the event.***

The following links can be found within the Workshop guide itself[*]:

- [Git Repo: kitchensink-conf](#): Configuration GitHub repository. Contains all the Kubernetes objects needed to deploy Kitchensink.
- [Git Repo: kitchensink \(app\)](#): GitHub repository with the Kitchensink app code.



Red Hat
Summit

Connect

Thank you



linkedin.com/company/red-hat



facebook.com/redhatinc



youtube.com/user/RedHatVideos



twitter.com/RedHat