**IBM APP CONNECT & OPENSHIFT PIPELINE   
(TEKTON) - INTEGRACIÓN**

|  |  |
| --- | --- |
| **DOCUMENTO DE INSTALACIÓN, CONFIGURACIÓN & MANEJO** |  |
| **Versión 1.0** |

|  |  |
| --- | --- |
| **Contact Detail IBM** | Cesar Guerra - cesar.guerra@ibm.com |
| **Fecha** | Septiembre 2023 |

# ÍNDICE:

# INTRODUCCIÓN………………………………………………………………………………………………………………………………………………………….…….3

# INSTALACIÓN: *OPENSHIFT PIPELINES*……………………………………………………………………………………………………………………….…….……5

# INSTALACIÓN: *TEKTON DASHBOARD*……………………………..……………………………………………………………………………………….………..…..9

# DEMO: *INTEGRACIÓN*……………………………………………………………………………………………………………………………………………………....13

**IBM APP CONNECT & OPENSHIFT PIPELINES  
(TEKTON) - INTEGRACIÓN**

# Logotipo de TektonA blue and white logo Description automatically generated

# 

# El presente documento abarca la INTEGRACIÓN de IBM APP CONNECT 12 con *OPENSHIFT PIPELINES* (TEKTON):

|  |  |
| --- | --- |
| **DESCRIPCIÓN** | **DETALLE** |
| 1. **INTRODUCCIÓN:** | |
| * **TEKTON** es una capacidad de infraestructura nativa de [***KUBERNETES***](https://www.ibm.com/pe-es/cloud/learn/kubernetes), para la creación de sistemas de integración & entrega/despliegue continuo **(CI/CD)**. Así mismo, ayuda a [modernizar dicha ***continuidad***](https://www.ibm.com/cloud/blog/tekton-a-modern-approach-to-continuous-delivery) ofreciendo especificaciones de la industria para el manejo de: ***pipelines***, en múltiples entorno: ***On-Cloud, On-Premise*** e ***Híbrido***, supervisados por un panel de control integrado ***(DASHBOARD)***. * **OPENSHIT PIPELINEs** es una capacidad de [***OPENSHIT***](https://www.ibm.com/pe-es/cloud/learn/kubernetes)basada en **TEKTON**, para proporcionar una experiencia de **CI/CD**, a través de una excelente integración con las herramientas  de ***OPENSHIFT (RECURSOS)***. | |
| En **TEKTON** se manejan varios ***CONCEPTOS*** asociados  a sus ***RECURSOS***, que son utilizados en ***KUBERNETES*** durante el ciclo de vida de ejecución, estos son: | |  |  | | --- | --- | | **STEP:** | Es una operación independiente del flujo de **CI/CD**, donde en cada **STEP** se puede correr una **IMAGEN** contenerizada. | | **TASK** | Es una colección de **STEPs** en orden, donde cada **TASK** se representa como un **POD**.   * *Compilar código.* * *Ejecutar test.* * *Crear imagen de contenedor.* | | **TASK-RUN** | Es el recurso asociado a la acción de ejecutar & mostrar el resultado una ***TASK***. | | **PIPELINE** | Es una colección de ***TASKs*** en orden. | | **PIPELINE-RUN** | Es el recurso asociado a la acción de ejecutar & mostrar el resultado una ***PIPELINE***. | | **PIPELINE -RESOURCE** | Este recurso define un objeto de ***ENTRADA*** *(como un repositorio de Git)* o ​​una ***SALIDA***  *(como una imagen de Docker)* del ***PIPELINE***. | | **PARAMs** | Es la información que requiere un ***PIPELINE*** o ***TASK*** para su ejecución. | | **WORKSPACES** | Es un ***FileSystem*** utilizado en ***RUNTIME*** (***TASK-RUN***), para compartir información  entre ***TASKs***. | | **EVENT-LISTENER** | Este encargará de escuchar los ***EVENTOS* automáticos**, recibir las ***PETICIONES*** & llamar  a los ***TRIGGERs***. | | **TRIGGER** | Este define lo que se ejecutará cuando el recurso: ***EVENT-LISTENER*** detecta un evento.  Así mismo, internamente hace referencia e invoca a los recursos: ***TRIGGER-TEMPLATE****,* ***TRIGGER-BINDING***, opcionalmente a: ***INTERCEPTOR***. | | **TRIGGER-TEMPLATE** | Es utilizado para crear ***RECURSOS*** como ***PIPELINE-RUNs*** & ***TASK-RUNs****.* | | **TRIGGER-BINDING** | Es utilizado para capturar los campos (***PAYLOAD***), de un ***EVENTO*** & guardarlo como ***PARÁMETRO***. | | **INTERCEPTOR** | Permite realizar un filtrado del ***WORKLOAD***, para la verificación*,* definir, probar condiciones &  otros procesos útiles. Modifican el comportamiento de ***TRIGGERs****.* | |
| Luego, en las ***IMÁGENES*** se muestra la relación  entre los ***RECURSOS*** de tipo: ***TASK, PIPELINE, PIPELINE-RUN, TASK-RUN*** asociados a los ***PODs***. Así como la secuencia de **STEPs** que se maneja a nivel de: **TASKs**. |  |
| Finalmente, vale mencionar la existencia de: **TEKTON-HUB**, donde se podrá encontrar un gran listado con la ***DOCUMENTACIÓN*** de ***TASK*** predefinidas & listas para ser utilizadas: <https://hub.tekton.dev/?kind=Task> |  |
| **2. INSTALACIÓN: *OPENSHIFT PIPELINES*** | |
| Ingresar en **OPERATOR-HUB** & filtrar por: ***Openshift Pipelines****.* |  |
| Luego, desde aquí antes de ***INSTALAR***, se requiere ***DESCARGAR*** el cliente de **TEKTON**.  Otra ***OPCIÓN*** es realizar la ***DESCARGA*** desde aquí:  [*https://docs.openshift.com/container-platform/4.10/cli\_reference/tkn\_cli/installing-tkn.html*](https://docs.openshift.com/container-platform/4.10/cli_reference/tkn_cli/installing-tkn.html) |  |
| **CLIENTE TEKTON:**  En este caso para fines de la presentación, ***DESCARGAR***  para ***WINDOWS*** el archivo: **tkn-windows-amd64.zip**,  & descomprimir en un ***FILESYSTEM*** & referenciarlo  a nivel de ***VARIABLES DE ENTORNO***, como se muestra en ***IMAGEN***.  Así mismo, luego de instalar se ***VALIDA*** el cliente de **TEKTON** instalado:  *$ tkn version* |  |
| Luego, regresando a ***OPENSHIFT***, se procede a ***INSTALAR*** el**OPERADOR**, aplicando la configuración como se muestra en la ***IMAGEN*** *(all namespaces).*  **IMPORTANTE:** *“Siempre utilizar la versión mayor del* ***CHANNEL*** *existente”.* |  |
| Luego, se valida el estado de ***OPERADOR*** que figure **EXITOSO**. |  |
| Luego, se valida la existencia del **SECRET** asociado llamado: **pipeline**, si figura quiere decir que ya se tiene los ***PERMISOS*** para el manejo de ***PIPELINES*** en ***OPENSHIFT***:  *$ oc get serviceaccount pipeline* |  |
| Finalmente, se aprecia a nivel del ***MENÚ PRINCIPAL***,  que se ha generado un ***BLOQUE*** nuevo llamado: **PIPELINES** *(antes* ***NO*** *existía).* |  |
| **3. INSTALACIÓN: *TEKTON DASHBOARD*** | |
| **TEKTON-DASHBOARD** es una ***CONSOLA WEB*** con el propósito de la administración & ejecución ***RECURSOS***: [**TEKTON**,](https://github.com/tektoncd/pipeline) esta permite lo siguiente:   * *Filtrar recursos.* * *Vista en runtime de los recursos:* ***PipelineRun*** *& ​​****TaskRun*** * *Ver detalles de recursos existentes &* ***YAMLs****.* * *Agregar funcionalidad a través de extensiones.* |  |
| Luego, para la instalación del ***DASHABORD*** se deberá  realizar los siguientes pasos:  En este punto debe ***EXISTIR*** ya el ***NAMESPACE***: *openshift-pipelines (este será creado por el* ***OPERATOR*** *en la instalación).*  *$ oc get all -n openshift-pipelines*   * **TEKTON-DASHBOARD:** contiene el **GITHUB** el ***script YAML*** con los recursos requeridos para la instalación del ***DASHBOARD***.     Después, se procede con la ***CREACIÓN***:  *$ oc create -f https://raw.githubusercontent.com/maktup/dummy-tekton-ace/main/scripts/0\_dummy-appconnect-tekton-dashboard.yaml* |  |
| Luego, se sebe validar todos los ***RECURSOS*** creados  en el ***NAMESPACE***, resaltando los asociados al: ***DASHBOARD*** como se muestra en ***IMAGEN***:    *$ oc get pods,deployments,services,routes -n openshift-pipelines* |  |
| Luego, se ingresa gráficamente en **OPENSHIFT /** **ROUTES** filtrando por ***NAMESPACE***: *openshift-pipelines*, & seleccionar: *tekton-dashboard*, ingresando al ***LINK*** similar generado:  [*http://tekton-dashboard-openshift-pipelines.apps.rey.coc-ibm.com*](http://tekton-dashboard-openshift-pipelines.apps.rey.coc-ibm.com) |  |
| Finalmente, se aprecia como debería visualizarse el **TEKTON** ***DASHBOARD*** desplegado. |  |
| **4. DEMO: *INTEGRACIÓN*** | |
| **TIPOS DE DESPLIEGUE:**   1. Recurso **PIPELINE**: *Utilizado para la ejecución* ***MANUAL*** *(****PIPELINE RUN, PIPELINE START****).* 2. Recurso **TRIGGER**: *Complemento utilizado para la ejecución* ***AUTOMATIZADA****.* | |
| Por tema de orden, se recomienda manejar un ***REPOSITORIO***: **GIT**independiente por ***MicroServicio***,  en el cual se ubiquen los ***Script*** utilizados.  En este caso se maneja:  **REPOSITORIO:** *https://github.com/maktup/dummy-tekton-ace.git* |  |
| Los ***DIRECTORIOS*** internos en **GITHUB**que se manejan  en este punto, contienen:   * **scripts:** contiene los ***scripts YAML*** para ser instalados ***ORDENADAMENTE***, para la construcción del flujo ***DEVOPs*** con **TEKTON**. |  |
| |  |  | | --- | --- | | **NOMBRE** | **DETALLE** | | **0\_dummy-appconnect-tekton-dashboard.yaml** | *Script* ***YAML*** *utilizado para la instalación del* ***DASHBOARD*** *de* ***TEKTON****.* | | **1\_dummy-appconnect-requerimientos.yaml** | *Script* ***YAML*** *utilizado para la instalación de* ***TODOS*** *los* ***REQUERIMIENTOS*** *previos al manejo de* ***OPENSHIFT PIPELINE*** *(acceso, seguridad, etc).* | | **2\_dummy-appconnect-pipeline.yaml** | *Script* ***YAML*** *utilizado para la instalación del* ***RECURSO*** *con la estructura de* ***PIPELINE****.* | | **3\_dummy-appconnect-task.yaml** | *Script* ***YAML*** *utilizado para la instalación de los* ***RECURSOS*** *de tipo* ***TASK*** *con la* ***LÓGICA*** *de flujo* ***(+ IMPORTANTE)****.* | | **4\_dummy-appconnect-trigger.yaml** | *Script* ***YAML*** *utilizado para la instalación de los* ***RECURSOS*** *para la* ***AUTOMATIZACIÓN*** *deldespliegue.* | | **5\_dummy-appconect-ejecucion.yaml** | *Script* ***YAML*** *utilizado para el* ***TEST*** *por medio del* ***PIPILINE RUN****.* | | |
| * **dummy\_csm\_microservice:** consiste en el ***DIRECTORIO*** del proyecto de **ACE12**, con las ***FUENTES*** del ***MICROSERVICIO***. |  |
| Luego, toca ahora proceder a la instalación de cada ***RECURSO***, referenciando a los ***scripts*** existentes en **GITHUB**, en este caso solo estos se ***EJECUTARÍAN***:  **IMPORTANTE:** *“El* ***NAMESPACE BASE*** *que se utilizará para la* ***DEMO*** *será: dummy-tekton-appconnect”*  *$ oc create ns dummy-tekton-appconnect*  ***//Creación REQUERIMIENTOS:***  *$ oc create -f https://raw.githubusercontent.com/maktup/dummy-tekton-ace/main/scripts/1\_dummy-appconnect-requerimientos.yaml*    ***//Creación PERMISOS:***  *$ oc adm policy add-scc-to-user privileged -z pipeline -n dummy-tekton-appconnect*  *$ oc adm policy add-role-to-user edit -z pipeline -n dummy-tekton-appconnect*  ***//Creación de PIPELINE:***  *$ oc create -f https://raw.githubusercontent.com/maktup/dummy-tekton-ace/main/scripts/2\_dummy-appconnect-pipeline.yaml*  ***//Creación de TASKs:***  *$ oc create -f https://raw.githubusercontent.com/maktup/dummy-tekton-ace/main/scripts/3\_dummy-appconnect-task.yaml*  ***//Creación de TRIGGERs:***  *$ oc create -f https://raw.githubusercontent.com/maktup/dummy-tekton-ace/main/scripts/4\_dummy-appconnect-trigger.yaml* |  |
| Luego, de ejecutados los ***SCRIPTs***, se procede a validar la existencia de algunos ***RECURSOS*** importantes de **TEKTON**:  ***//Listar PIPELINES existentes en el NAMESPACE:***  *$ tkn pipeline ls -n dummy-tekton-appconnect*  ***//Listar TAREAS (CREADAS MANUALMENTE) existentes en el NAMESPACE:***  *$ tkn task ls -n dummy-tekton-appconnect* |  |
| 1. **EJECUCIÓN “MANUAL”:** | |
| El ***TEST*** de ejecución**MANUAL** del flujo **PIPELINE**, se puede realizar de **2** formas:   * 1. **PIPELINE RUN:** *en base a un Script* ***YAML***   *$ oc create -f* [*https://raw.githubusercontent.com/maktup/dummy-tekton-ace/main/scripts/5\_dummy-appconect-ejecucion.yaml*](https://raw.githubusercontent.com/maktup/dummy-tekton-ace/main/scripts/5_dummy-appconect-ejecucion.yaml) | A screen shot of a computer  Description automatically generated |
| 1. **PILEPINE START:** *en base a* ***COMANDOS****.*   *$ tkn pipeline start* ***pipeline-build-and-deploy*** *-w name=****workspace-pipeline****,claimName=****source-pvc-pipe*** *-p nombre-integration-server=csm-microservicio -p nombre-namespace-appconnect=dummy-tekton-appconnect -p url-repositorio-git=https://github.com/maktup/dummy-tekton-ace.git -p nombre-subdirectorio-git=dummy\_csm\_microservice -p nombre-repositorio-git=dummy-tekton-ace -p branch-repositorio-git=main -p nombre-proyecto-appconnect=dummy\_csm\_microservice -p version-image-appconnect="12.0.7.0-r3" -p licencia-image-appconnect="L-APEH-CJUCNR" -p ruta-image-appconnect="docker.io/maktup/ibm-appconnect-12:latest" -p tipo-licencia-appconnect="CloudPakForIntegrationNonProductionFREE" -p tipo-despliegue-servidor="3" -n dummy-tekton-appconnect*  **IMPORTANTE: “***los campos* ***RESALTADOS*** *hacen referencia a los* ***RECURSOS*** *&* ***PARÁMETROS*** *para enviar, propiamente al* ***RECURSO******PIPELINE*** *existente  en el* ***NAMESPACE”***.  *“Solo deberán ser enviarán los* ***PARÁMETROS*** *que se requiere que sean* **DINÁMICOS***, los que* ***NO*** *sean enviados serán seteados en* ***HardCode*** *desde: pipeline-build-and-deploy”.*  Además, los ***LOGs*** generados sería ***SIMILARES*** a este:  *$ tkn pipelinerun logs pipeline-build-and-deploy-run-k5rnx -f -n dummy-tekton-appconnect* |  |

|  |  |
| --- | --- |
| Luego, se procede a validar el **REQUEST** enviado  ***¿QUÉ PARÁMETROS FUERON ENVIADO AL PIPELINE?*** en: **RUNTIME**:  *$ tkn pipelinerun list -n dummy-tekton-appconnect*  *$ tkn pipelinerun describe pipeline-build-and-deploy-run-4xt75 -n dummy-tekton-appconnect*  **IMPORTANTE: *“****Reemplazar el valor del* ***PIPELINE*** *en la* ***2da*** *sentencia de código”.* |  |
| Luego, ingresando en: ***OPERATOR / INSTALLED OPERATOR / IBM App Connect***, se aprecia que la ***AUTOMATIZACIÓN*** ha logrado ***DESPLEGAR*** el **.bar** en las modalidades:   * ***INTEGRATION SERVER.*** * ***INTEGRATION RUNTIME.***   **IMPORTANTE:** *“Este* ***INTEGRATION RUNTIME*** *es  una nueva característica de* ***IBM APP CONNECT****, que habilita crear* ***SERVERLESS***integration runtimes*”.* |  |
|  |
| Finalmente, el resultado del ***TEST*** es el siguiente:   * ***INTEGRATION SERVER.***   *http://****csm-microservicio-is****-http-dummy-tekton-appconnect.apps.rey.coc-ibm.com/csm-microservice/get/empleados*   * ***INTEGRATIO RUNTIME.***   *http://****csm-microservicio-ir****-http-dummy-tekton-appconnect.apps.rey.coc-ibm.com/csm-microservice/get/empleados* |  |
| 1. **EJECUCIÓN “AUTOMÁTICA”:** | |
| Para el ***TEST*** de ejecución**AUTOMÁTICA** del flujo **PIPELINE**, es necesario el uso de un recurso: ***ROUTE*** para asociado al: **EVENT-LISTENER**.  Este servirá en la comunicación con el ***REPOSITORIO*** (**GITHUB**), & se obtienen ingresando en **OPENSHIFT**: ***NETWORKING / ROUTES*** se selecciona el ***NAMESPACE*** manejado, se selecciona el ***ROUTE*** con el nombre: (el-listener-\*), se validar que esté activo & se muestre el ***JSON***.    **IMPORTANTE:** *“se debe de guardar la* ***URL*** *del* ***EVENT-LISTENER*** *generada*: [*http://el-listener-dummy-micro-01-dummy-tekton-appconnect.apps.rey.coc-ibm.com*](http://el-listener-dummy-micro-01-dummy-tekton-appconnect.apps.rey.coc-ibm.com), *ya que será requerido más adelante como* ***PARÁMETRO****: Payload URL”.* |  |
| Luego, se procede a obtener el valor del ***SECRET*** registrado para ***REGISTRADO*** en el repositorio **GIT**, para ***CONFIGURAR*** la ***CONEXIÓN***:    *$ oc get secrets github-interceptor-secret -n dummy-tekton-appconnect -o yaml*  *$ echo -n MTIzNDU2Nzg= | base64 --decode*  **IMPORTANTE:** *“El* ***SECRET*** *debe ser actualizado  en la* ***2da*** *sentencia de comando, como se muestra en* ***IMAGEN****”.* |  |
| Luego, la ***URL*** anterior del ***EVENT-LISTENER*** se ingresa en la ruta de ***GITHUB: SETTINGs/WEBHOOKs*** para crear una ***CONEXION***. Además, el valor del **SECRET** obtenido se ingresa, como se muestra en ***IMAGEN***.  Esto hará que toda modificación que ***IMPACTE*** contra  el ***REPOSITORIO*** de ***GIT***, desencadenará & gatillará  la activación del flujo: **PIPELINE**.  **IMPORTANTE:** *“Esta conexión se debe realizar por* ***CLÚSTER*** *que se desee conectar”.* | A screenshot of a computer screen  Description automatically generated |
| **EJECUCIÓN AUTOMÁTICA:**  Para gatillar el flujo **PIPELINE**, se necesitará una ***MODIFICACIÓN*** en cualquier ***FUENTE*** de **ACE12**  dentro del ***REPOSITORIO*** de**GITHUB**,  En la ***IMAGEN*** se aprecia como se procede a ***MODIFICAR*** un campo en el ***ESQL*** de **ACE12**. Luego, el directorio completo del ***PROYECTO*** es reemplazado locamente.  Después los ***COMANDOS*** **GIT** para la ejecutar son:  *$ git init*  *$ git clone https://github.com/maktup/dummy-tekton-ace.git*  *$ cd dummy\_csm\_microservice*  *$ git add .*  *$ git status*  *$ git commit -m "ACTUALIZANDO REPOSITORIO!"*  *$ git push -f origin main*  **IMPORTANTE:** *“Este funcionará para el escenario  de este* ***MICROSERVICIO****, para otros de debe* ***ADAPTAR*** *(URL, Directorio, etc)”.* | A screenshot of a computer  Description automatically generated |
|  |
| Luego, como se muestra en las ***IMÁGENES*** el ***DESPLIEGUE*** **AUTOMÁTICO** del ***MICROSERVICIO*** se ha iniciado & después de un rato este será ***COMPLETADO*** a nivel del: **PIPELINE**. |  |
| Finalmente, luego de **TERMINADO** el procesamiento del **PIPELINE**, se realiza el ***TEST*** de validación del ***MICROSERVICIO*** desplegado funcione correctamente, con la **MODIFICACIÓN** realizada:   * ***INTEGRATION SERVER.***   *http://****csm-microservicio-is****-http-dummy-tekton-appconnect.apps.rey.coc-ibm.com/csm-microservice/get/empleados*   * ***INTEGRATIO RUNTIME.***   *http://****csm-microservicio-ir****-http-dummy-tekton-appconnect.apps.rey.coc-ibm.com/csm-microservice/get/empleados* |  |

**REFERENCIAS:**

* [*https://tekton.dev/docs*](https://tekton.dev/docs)
* [*https://www.ibm.com/docs/en/app-connect/containers\_cd?topic=releases-release-notes-app-connect-operator-820*](https://www.ibm.com/docs/en/app-connect/containers_cd?topic=releases-release-notes-app-connect-operator-820)
* [*https://www.ibm.com/docs/en/app-connect/containers\_cd?topic=obtaining-app-connect-enterprise-server-image-from-cloud-container-registry*](https://www.ibm.com/docs/en/app-connect/containers_cd?topic=obtaining-app-connect-enterprise-server-image-from-cloud-container-registry)
* [*https://dalelane.co.uk/blog/?p=4676#image-builder-pipeline*](https://dalelane.co.uk/blog/?p=4676#image-builder-pipeline)