Documentación Pipelines

Cliente: Banco Bolivariano de Ecuador

Proyecto: Integración continua para las líneas 24Móvil y OnBoarding

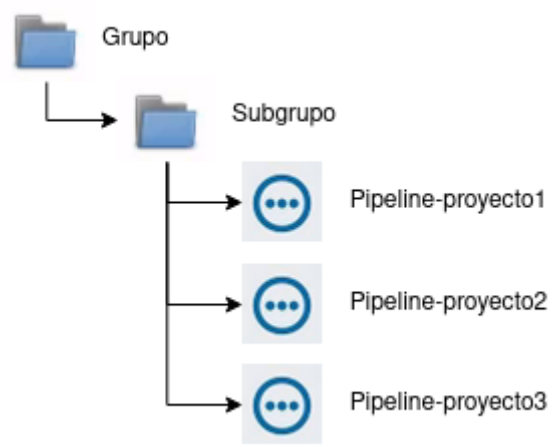
Equipo DevOps

Aprovisionamiento Pipelines Proyectos **3**
1.1. Estructura de Directorios 3
1.2. Flujos de trabajo 3
Proyectos ubicados en los subgrupos con sufijo "Microservicios" y "Cliente" 4
Flujo de integración continua desde la plataforma de Jenkins 4
Flujo de despliegue 5
Proyectos ubicados en el subgrupo "BaseDeDatos" 6
Flujo de integración continua desde la plataforma de Jenkins 6
Flujo de despliegue 7
Proyectos ubicados en los subgrupos con sufijo "Frontend" 7
Flujo de integración continua desde la plataforma de Jenkins 7
Flujo de despliegue 8
1.2. Ejecución Pipeline 9
Ejecución automática 10

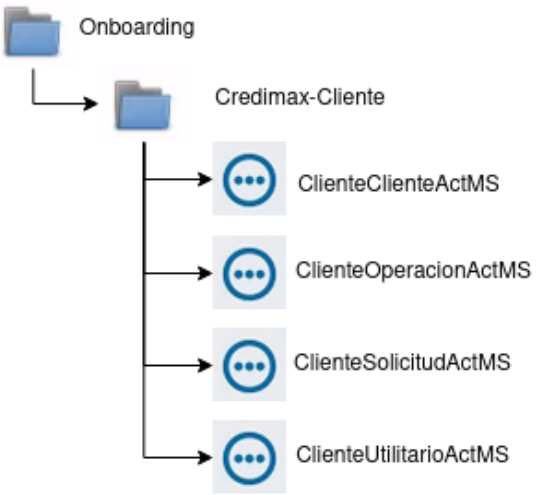
1. Aprovisionamiento Pipelines Proyectos

1.1. Estructura de Directorios

En la plataforma de Jenkins se realizó el aprovisionamiento de los pipelines correspondientes a los proyectos de los grupos de trabajo Onboarding y 24MovilRetail, quedando la estructura de directorios de la siguiente manera:



Ejemplo: ubicación de los pipelines de los proyectos del subgrupo **Credimax-Cliente.**



1.2. Flujos de trabajo

Para los proyectos Onboarding y 24MovilRetail se definieron los siguientes flujos de trabajo:

Proyectos ubicados en los subgrupos con sufijo "Microservicios" y "Cliente"

Flujo de integración continua desde la plataforma de Jenkins

Se definió el siguiente ciclo de integración continua para estos proyectos:

- **Obtener código fuente:** Jenkins realiza la descarga del código fuente almacenado en el repositorio del proyecto en Gitlab.

- **Compilación del código:** Utilizando maven versión 3.8.4, se ejecutará el siguiente comando para compilar el proyecto:

  mvn package -Dmaven.test.skip=true -Dmaven.wagon.http.ssl.insecure=true
  -Dmaven.wagon.http.ssl.allowall=true
  -Dmaven.wagon.http.ssl.ignore.validity.dates=true

- **Ejecución de pruebas unitarias:** Utilizando maven versión 3.8.4, se ejecutará el siguiente comando para ejecutar las pruebas unitarias del proyecto:

  mvn test

- **Ejecución de análisis estático de código:** El código es analizado con base a las reglas definidas en la plataforma de Sonarqube para cada lenguaje. Dependiendo del resultado del análisis Sonarqube retorna una respuesta a Jenkins para continuar o detener el proceso.

  El resultado del análisis y el reporte de los hallazgos encontrados queda almacenado en la plataforma de Sonarqube.

  Nota: Cada proyecto que se ejecuta desde la plataforma de Jenkins crea su propio proyecto, no es necesario crear el proyecto manualmente en la plataforma de Sonarqube. Además, se crea de forma automática el archivo sonar-project.properties con los siguientes datos:

  sonar.projectName = <nombre proyecto>
  sonar.projectKey = <nombre proyecto>
  sonar.sources = src/main
  sonar.projectBaseDir = ./

sonar.test = src/test
sonar.java.binaries = target/classes

- **Análisis de fuentes:** Se ejecuta el siguiente comando para realizar el análisis con Veracode del artefacto del proyecto (archivo .jar) generado en el paso de compilación:

  java -Dhttps.proxyHost=172.17.1.205 -Dhttps.proxyPort=9091 -jar /root/Jenkins/pipeline-scan.jar --file <archivo .jar a scannear> --fail_on_severity='Very High, High'

  Las opciones -Dhttps.proxyHost y -Dhttps.proxyPort se utilizan para especificar el proxy empleado para realizar la conexión con la plataforma de Veracode.

Flujo de despliegue

El flujo de despliegue comprende los siguientes pasos:

- **Generación de reportes:** El reporte de las pruebas unitarias realizadas sobre el proyecto se genera en un archivo .pdf empleando el comando: mvn site.

- **Análisis en Veracode:** El proceso ejecuta el pipeline ubicado en la siguiente url: https://jenkinsdes.bolivariano.fin.ec/job/tests/job/veracode/, con información relacionada al proyecto, como el nombre del grupo en Veracode, el nombre del proyecto y la ruta de las fuentes a analizar con base a las reglas de seguridad definidas en la plataforma de Veracode. Dependiendo del resultado del análisis, Veracode retorna una respuesta a Jenkins para continuar o detener el proceso.

- **Publicación de reporte en Jira:** El reporte de las pruebas unitarias es publicado en la tarea de Jira indicada al ejecutar el pipeline.

- **Aprobación análisis Veracode:** Notificación y aprobación de los resultados del análisis del proyecto en Veracode por parte del equipo de ciberseguridad.

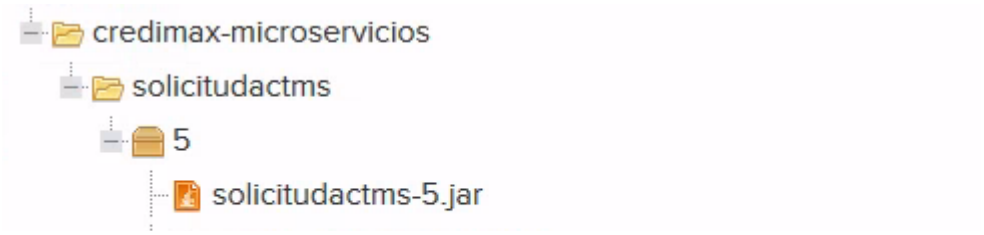
- **Aprobación de Matriz Unitaria:** Notificación y aprobación de los resultados de las pruebas unitarias del proyecto (se adjunta el reporte de las pruebas) por parte del equipo de administración y control.

- **Aprobación análisis SonarQube:** Notificación y aprobación de los resultados del análisis estático del código (se adjunta url del reporte en Sonarqube) por parte del equipo correspondiente.

- **Merge request a producción:** Se envía una petición a Gitlab de merge a la rama main del repositorio del proyecto. Está solicitud es asignada a Iván Vasquez Galarza para su aprobación.

- **Publicación Artefacto:** El artefacto generado por el proyecto es publicado en el repositorio respectivo (onboarding ó 24-movil-retail) en la plataforma de Nexus bajo el siguiente esquema:
  groupId: <subgrupo>
  artifactId: <nombreproyecto>
  version: <número de ejecución>

  Ejemplo de la publicación de un artefacto del proyecto SolicitudActMS:



- **Publicación de Documentación a Confluence (si se presenta un fallo en este paso, no afecta el proceso):** El reporte de las pruebas unitarias es publicado a la plataforma de confluence empleando el comando:

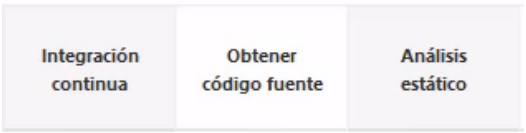
  java -jar /root/Jenkins/ClienteConfluence/ClienteConfluence-1.0-SNAPSHOT.jar --usuario=<usuario de Jira> --urlconfluence=https://bancobolivariano.atlassian.net --token=<token del usuario> --anexopruebas=<ruta reporte pdf de las pruebas unitarias> --grupo=<grupo respectivo (onboarding ó 24movil)>

- **Notificación:** Este paso envía un correo electrónico informando el estado final de la ejecución e información asociada al despliegue, como los aprobadores de cada paso, la tarea de Jira y su descripción.

Nota: Si el pipeline presenta alguna falla en cualquier parte del proceso, se enviará una notificación vía correo electrónico a los equipos correspondientes a cada proyecto, con información del pipeline donde se presentó el error.

Proyectos ubicados en el subgrupo "BaseDeDatos"

Flujo de integración continua desde la plataforma de Jenkins



Se definió el siguiente ciclo de integración continua para los proyectos sql:

- **Obtener código fuente:** Jenkins realiza la descarga del código fuente almacenado en el repositorio del proyecto en Gitlab.

- **Ejecución de análisis estático de código:** El código es analizado con base a las reglas definidas en la plataforma de Sonarqube para cada lenguaje; para los

proyectos de base de datos se emplea el plugin para sql instalado en Sonarqube. Dependiendo del resultado del análisis, Sonarqube retorna una respuesta a Jenkins para continuar o detener el proceso.

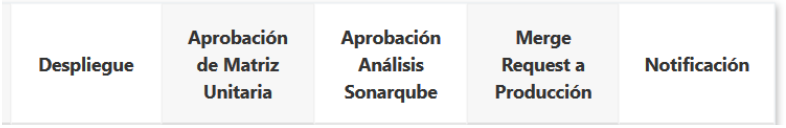
El resultado del análisis y el reporte de los hallazgos encontrados queda almacenado en la plataforma de Sonarqube.

Nota: Cada proyecto que se ejecuta desde la plataforma de Jenkins crea su propio proyecto, no es necesario crear el proyecto manualmente en la plataforma de Sonarqube.

Para los proyectos se construye de manera automática el archivo sonar-project.properties con los siguientes datos:

sonar.projectName = <nombre proyecto>
sonar.projectKey = <nombre proyecto>
sonar.projectBaseDir = ./

Flujo de despliegue



- **Aprobación de Matriz Unitaria:** Notificación y aprobación de la matriz unitaria del proyecto por parte del equipo de administración y control.

- **Aprobación análisis SonarQube:** Notificación y aprobación de los resultados del análisis estático del código (se adjunta url del reporte en Sonarqube) por parte del equipo correspondiente.

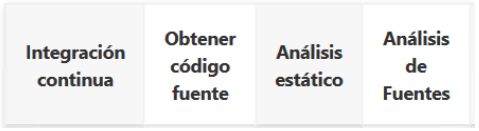
- **Merge request a producción:** Se envía una petición a Gitlab de merge a la rama main del repositorio del proyecto. Está solicitud es asignada a Iván Vasquez Galarza para su aprobación.

- **Notificación:** Este paso envía un correo electrónico informando el estado final de la ejecución e información asociada al despliegue, como los aprobadores de cada paso, la tarea de Jira y su descripción.

Nota: Si el pipeline presenta alguna falla en cualquier parte del proceso, se enviará una notificación vía correo electrónico a los equipos correspondientes a cada proyecto, con información del pipeline donde se presentó el error.

Proyectos ubicados en los subgrupos con sufijo "Frontend"

Flujo de integración continua desde la plataforma de Jenkins



Se definió el siguiente ciclo de integración continua para los proyectos frontend:

- **Obtener código fuente:** Jenkins realiza la descarga del código fuente almacenado en el repositorio del proyecto en Gitlab.

- **Ejecución de análisis estático de código:** El código es analizado con base a las reglas definidas en la plataforma de Sonarqube para cada lenguaje; para los proyectos de base de datos se emplea el plugin para sql instalado en Sonarqube. Dependiendo del resultado del análisis, Sonarqube retorna una respuesta a Jenkins para continuar o detener el proceso.

  El resultado del análisis y el reporte de los hallazgos encontrados queda almacenado en la plataforma de Sonarqube.

  Nota: Cada proyecto que se ejecuta desde la plataforma de Jenkins crea su propio proyecto, no es necesario crear el proyecto manualmente en la plataforma de Sonarqube.

  Para los proyectos se construye de manera automática el archivo sonar-project.properties con los siguientes datos:

  sonar.projectName = <nombre proyecto>
  sonar.projectKey = <nombre proyecto>
  sonar.projectBaseDir = ./

- **Análisis de fuentes:** Se ejecuta el siguiente comando para realizar el análisis con Veracode de las fuentes del proyecto (las fuentes se comprimen en un archivo .zip):

  java -Dhttps.proxyHost=172.17.1.205 -Dhttps.proxyPort=9091 -jar /root/Jenkins/pipeline-scan.jar --file <archivo .zip con las fuentes del proyecto> --fail_on_severity='Very High, High'

  Las opciones -Dhttps.proxyHost y -Dhttps.proxyPort se utilizan para especificar el proxy empleado para realizar la conexión con la plataforma de Veracode.

Flujo de despliegue

- **Análisis en Veracode:** El proceso ejecuta el pipeline ubicado en la siguiente url: https://jenkinsdes.bolivariano.fin.ec/job/tests/job/veracode/, con información relacionada al proyecto, como el nombre del grupo en Veracode, el nombre del proyecto y la ruta de las fuentes a analizar con base a las reglas de seguridad definidas en la plataforma de Veracode. Dependiendo del resultado del análisis, Veracode retorna una respuesta a Jenkins para continuar o detener el proceso.

- **Aprobación análisis Veracode:** Notificación y aprobación de los resultados del análisis del proyecto en Veracode por parte del equipo de ciberseguridad.

- **Aprobación de Matriz Unitaria:** Notificación y aprobación de la matriz unitaria del proyecto por parte del equipo de administración y control.

- **Aprobación análisis SonarQube:** Notificación y aprobación de los resultados del análisis estático del código (se adjunta url del reporte en Sonarqube) por parte del equipo correspondiente.

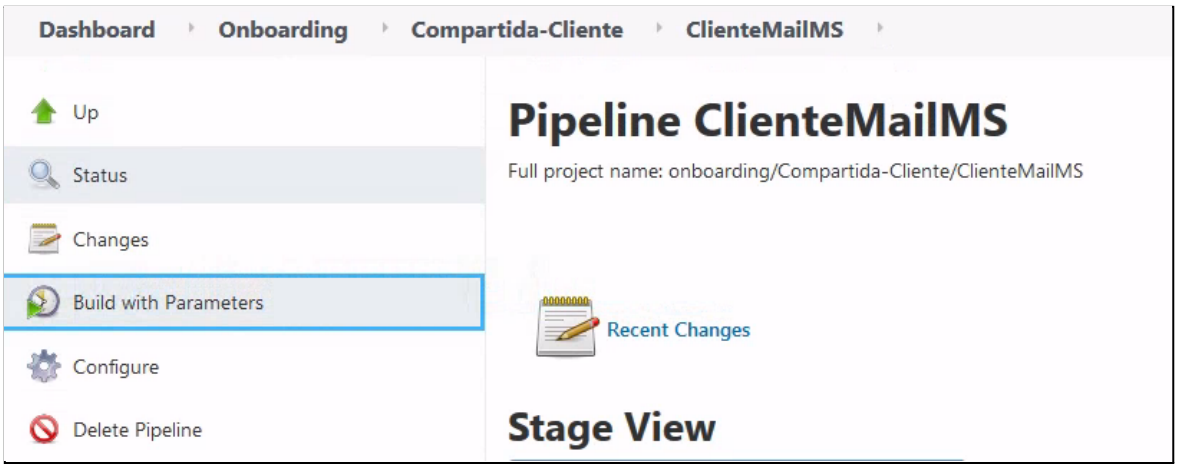
- **Merge request a producción:** Se envía una petición a Gitlab de merge a la rama main del repositorio del proyecto. Está solicitud es asignada a Iván Vasquez Galarza para su aprobación.

- **Notificación:** Este paso envía un correo electrónico informando el estado final de la ejecución e información asociada al despliegue, como los aprobadores de cada paso, la tarea de Jira y su descripción.

Nota: Si el pipeline presenta alguna falla en cualquier parte del proceso, se enviará una notificación vía correo electrónico a los equipos correspondientes a cada proyecto, con información del pipeline donde se presentó el error.

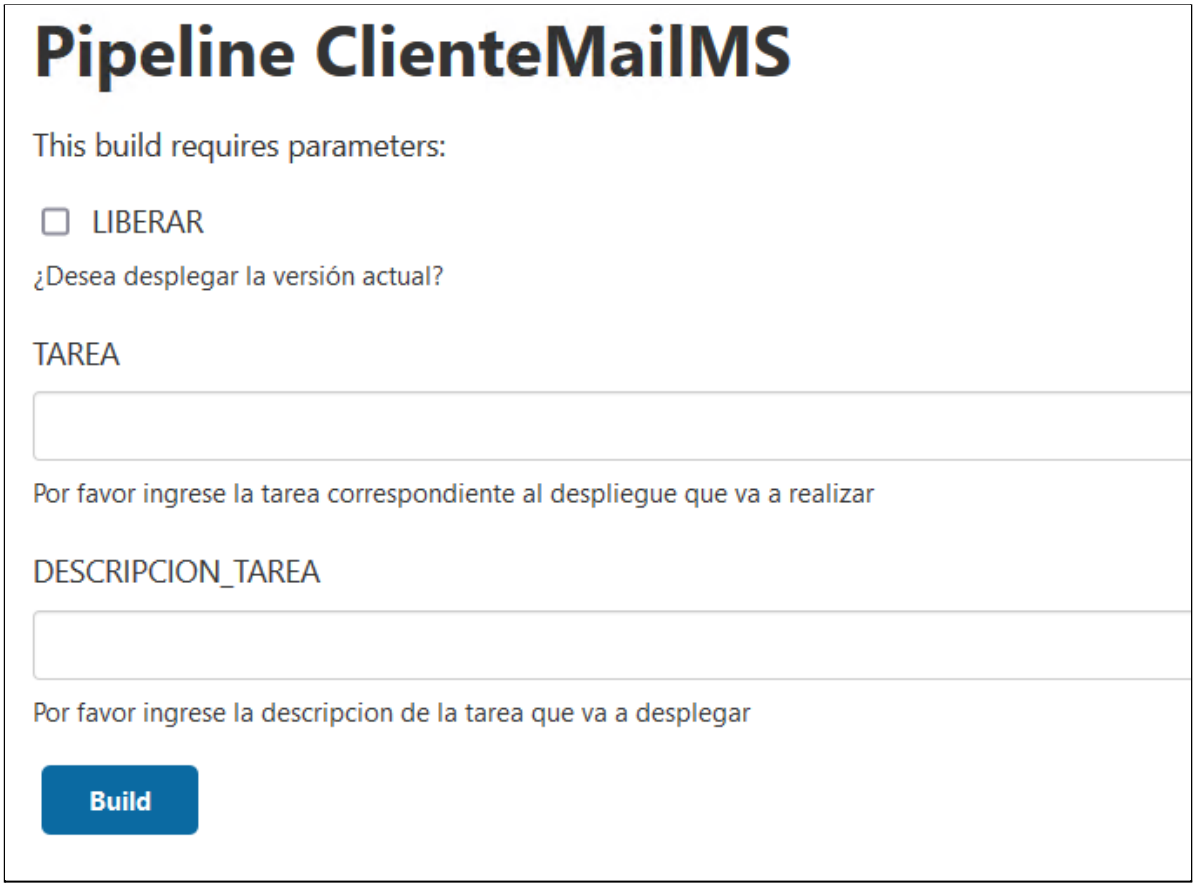
1.2. Ejecución Pipeline

Para ejecutar un pipeline seleccione la opción **Build with Parameters,** ubicada al lado izquierdo.



Los pipelines de los proyectos son parametrizados, es decir, que cuentan con un parámetro que se emplea para la ejecución del pipeline. El parámetro en este caso es **LIBERAR,** dependiendo si el parámetro está seleccionado o no se ejecuta únicamente el ciclo de integración continua o todo el proceso de despliegue.

El parámetro **TAREA,** permite agregar la tarea de Jira relacionada con el despliegue que se va a llevar a cabo y el parámetro **DESCRIPCION_TAREA**, permite agregar una descripción de la tarea solicitada en el parámetro **TAREA**.



Por defecto el parámetro **LIBERAR** se encuentra desactivado, por lo que el pipeline solo ejecutará el ciclo de integración definido para cada proyecto al dar clic en **Build**. Si se activa, se ejecutará el ciclo de producción, donde se solicita aprobación para continuar el proceso.

Ejecución automática

Los pipelines se encuentran integrados a Gitlab, para que al momento de realizar un push sobre el repositorio del proyecto en la rama **desarrollo**, el pipeline asociado al repositorio se ejecute automáticamente.