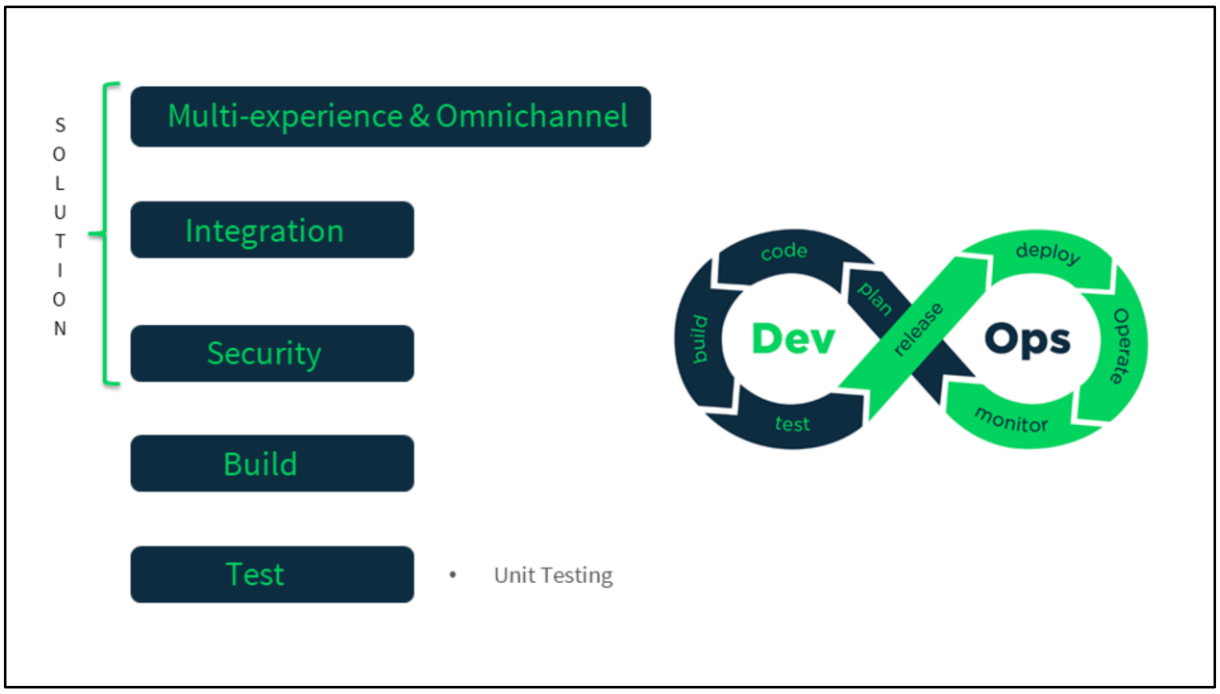
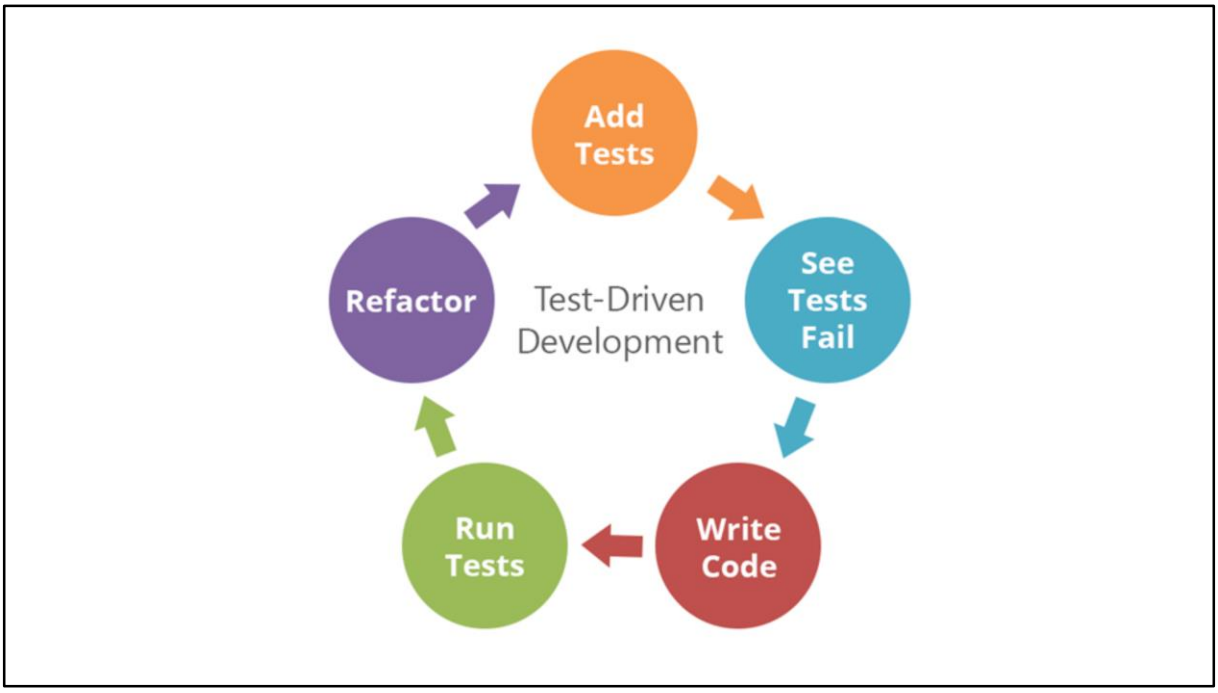
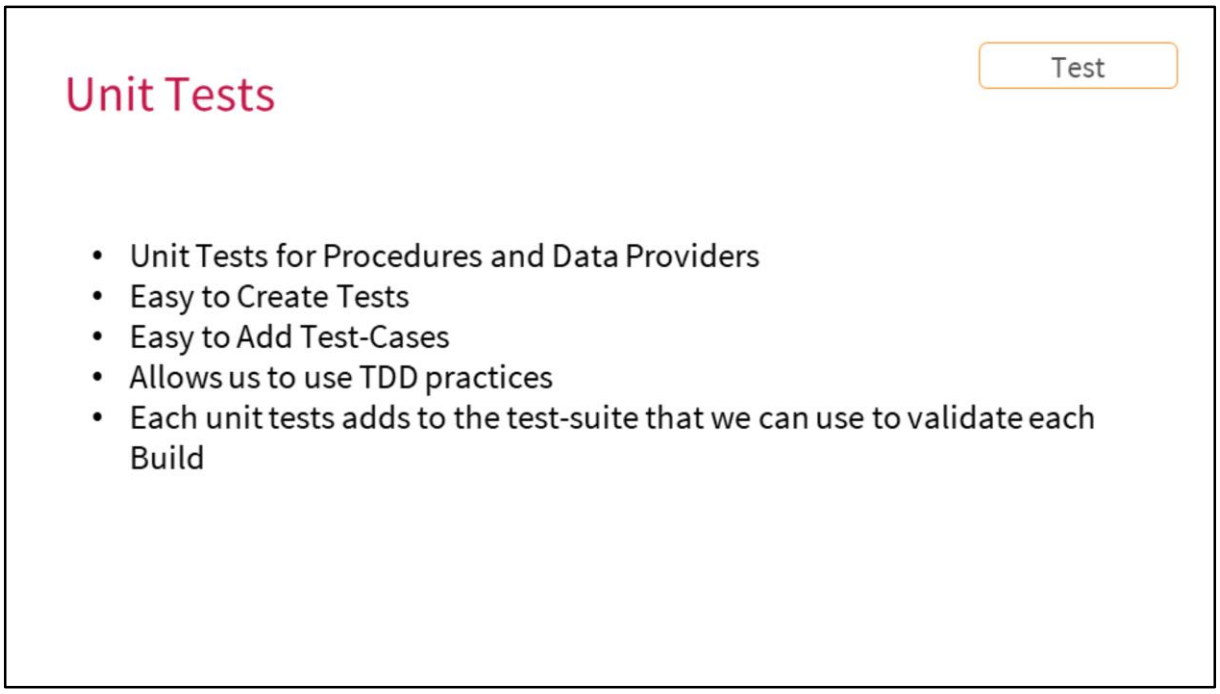

GeneXus™
The power of doing



Hasta aquí hemos atravesado las primeras partes del diagrama de DevOps, aspectos que hacen a la aplicación que estamos desarrollando y cómo hacerlo con GeneXus. Sobre Build ya hablamos en la introducción y se dirá algo más después.

Ahora veremos lo nuevo en Testing, para luego pasar a lo que hace a las Operaciones, la parte verde del diagrama.


Add Tests
See Tests Fail
Write Code
Run Tests
Refactor
Test-Driven Development

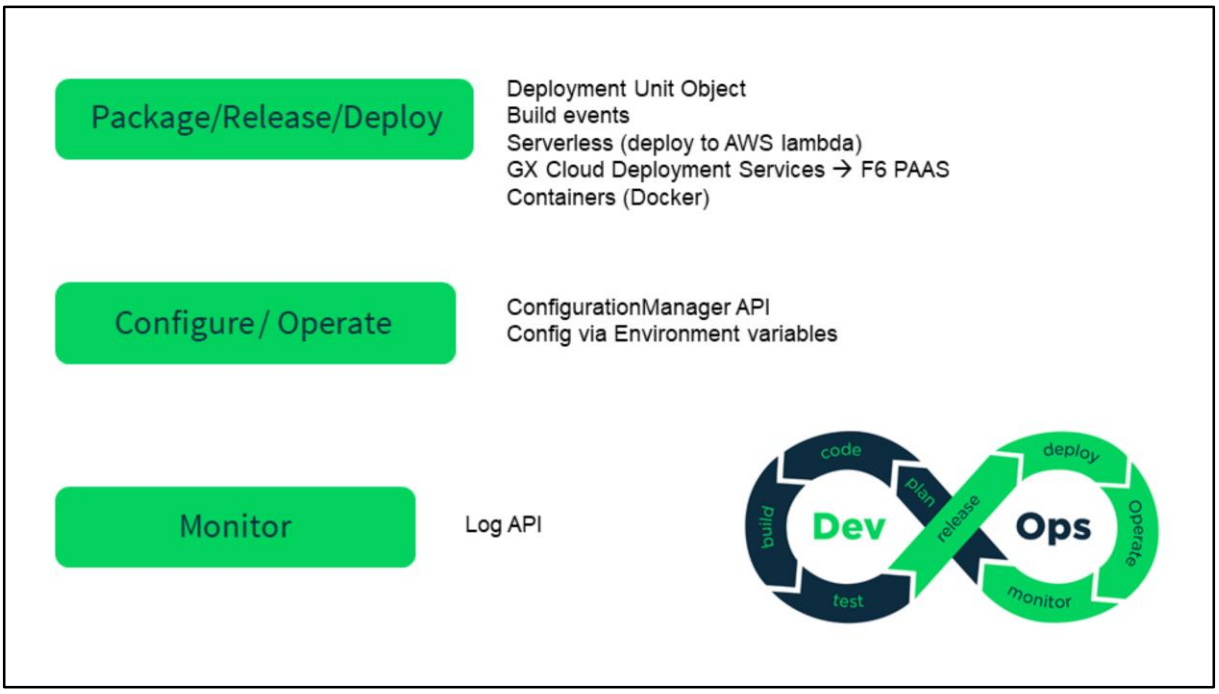


Respecto a Jenkins, ya tenemos en beta un plugin que se integra con GeneXus Server.
Así que pueden realizar procesos de CI CD con Jenkins a partir de allí.

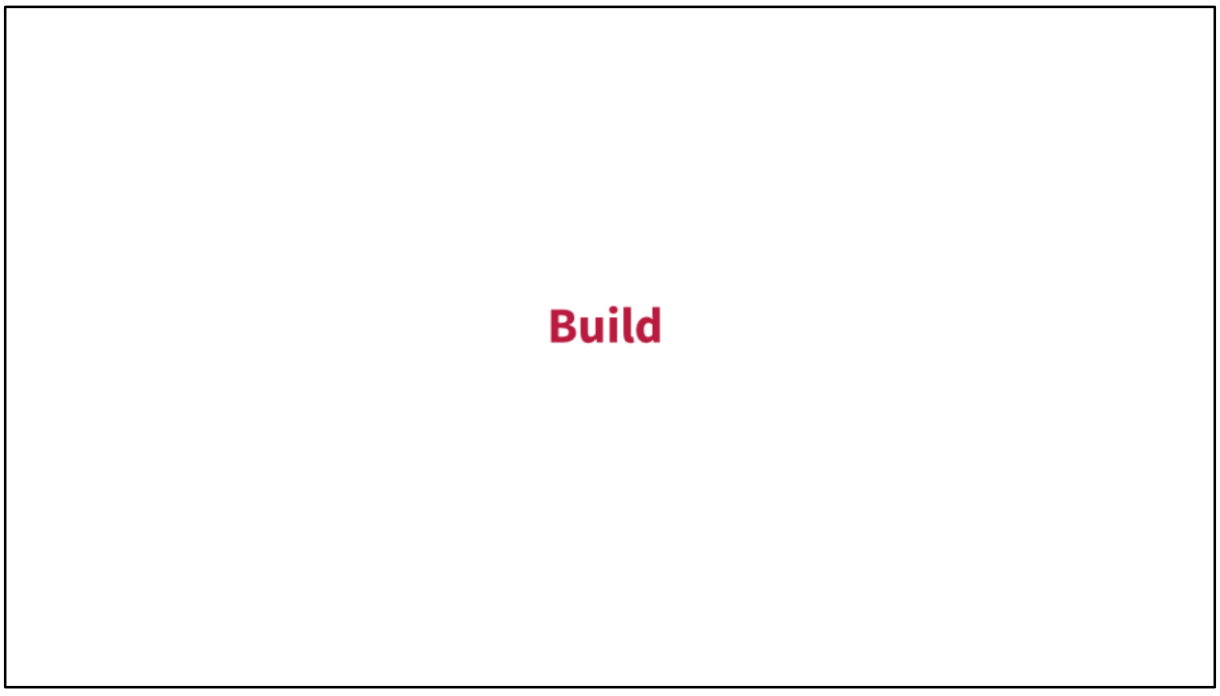

DEMO

Continuous development / Continuous integration

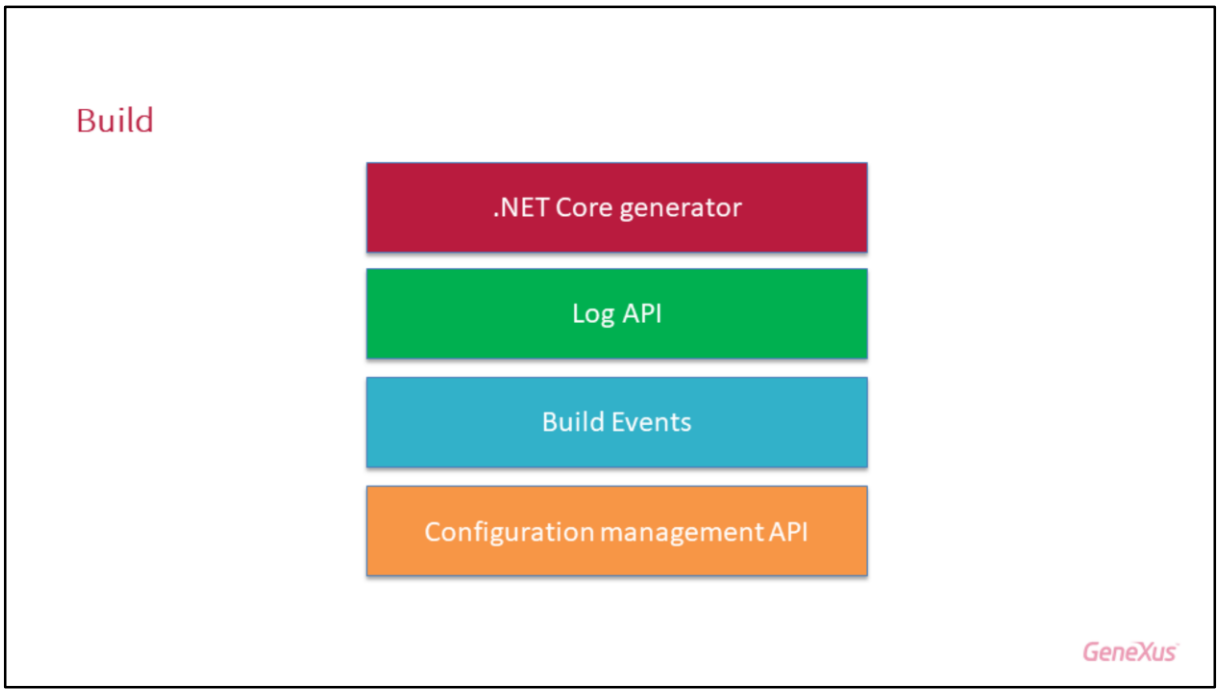
GeneXus 16



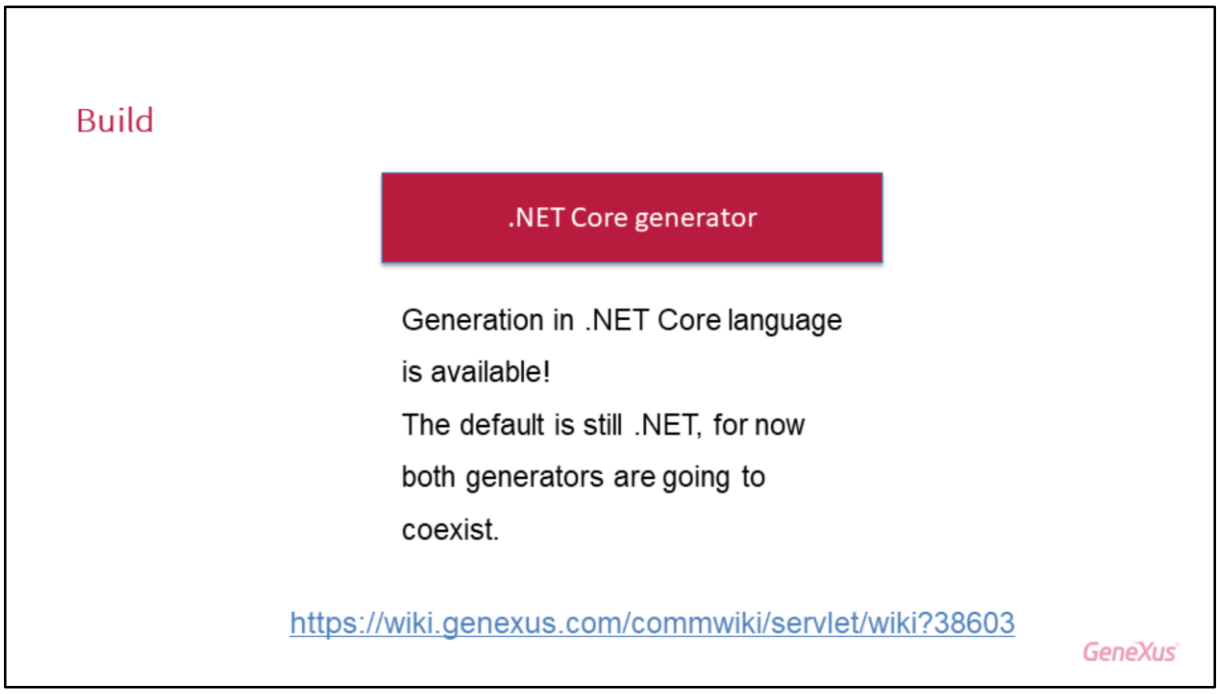
En la introducción separamos las funcionalidades que hacen a las Operaciones. Ahora veremos los aspectos de Package, Release, Deploy.



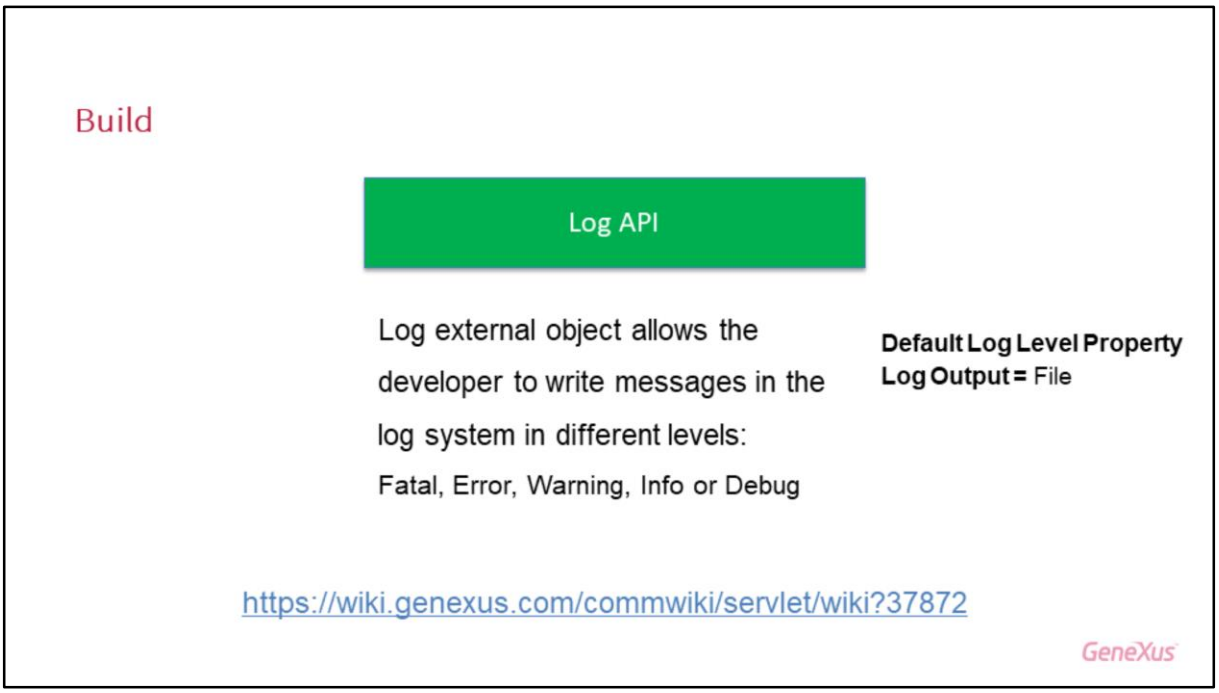
Trabajamos en varias funcionalidades relacionadas con el proceso de Build, que iremos detallando a continuación.



GeneXus 16 incluye varias funcionalidades interesantes con respecto al proceso de Build, como la generación en .NET Core, una API para crear logs del usuario, la posibilidad de disparar eventos asociados al Build, para ejecutar cosas antes o después del mismo y una API para leer la configuración que está utilizando una aplicación en ejecución.



GeneXus 16 incluye un generador .NET Core.
.NET Core es el nuevo framework de Microsoft de código abierto multiplataforma, compatible con Windows, Mac y Linux.
Este framework está optimizado para trabajar en la nube y permite desarrollo de aplicaciones web, aplicaciones de consola y aplicaciones de Inteligencia Artificial, Machine Learning e IoT.
Si bien se assume que es el sucesor del .NET, van a pasar unos años antes de que se convierta un estándar del mercado, por lo cual en GeneXus va a seguir existiendo el generador .NET y ambos generadores estarán disponibles. El código generado sigue siendo C#.
Más información en: https://wiki.genexus.com/commwiki/servlet/wiki?38603

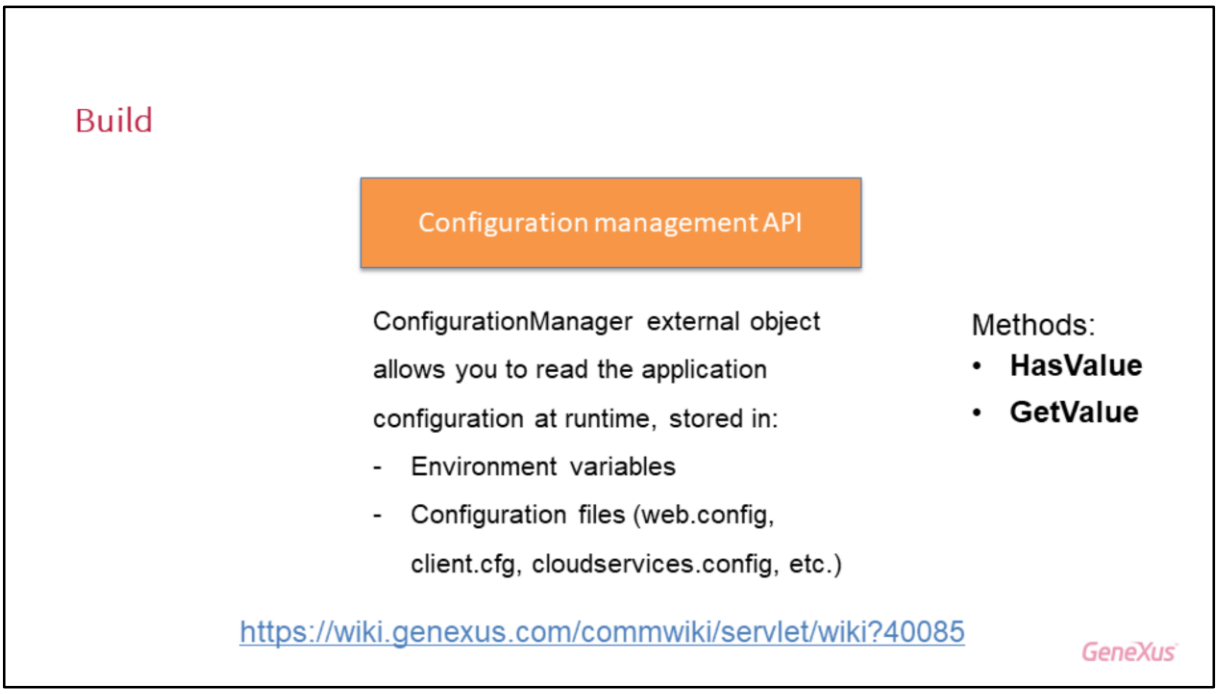


GeneXus 16 ofrece una API que permite que los usuarios puedan agregar sus propios mensajes de log desde la aplicación, en el log del sistema.
El objeto externo Log está disponible tanto para plataforma Web, como para SD.

Si el log está habilitado (es decir que la propiedad **Default Log Level Property** tiene un valor diferente de "Off" y la propiedad **Log Output** tiene el valor "File"), en .NET el archivo de log que se llama **client.log** y en Java se envía al **log del Tomcat**, quien divide los mensajes en varios archivos de log según el tipo de error. Para ver los logs en plataforma Smart Devices, en Android se utiliza la herramienta **LogCat** que puede invocarse por línea de comando o mediante el **Android Device Monitor** y en iOS se pueden ver los archivos de log con la **consola de debug de Xcode**, o buscando los archivos .log en el directorio de datos de la aplicación.

La API tiene varios métodos que permiten generar distintos tipos de entradas en el log, según el orden de importancia, como Fatal, Error, Warning, Info o Debug. Más información en: https://wiki.genexus.com/commwiki/servlet/wiki?37872

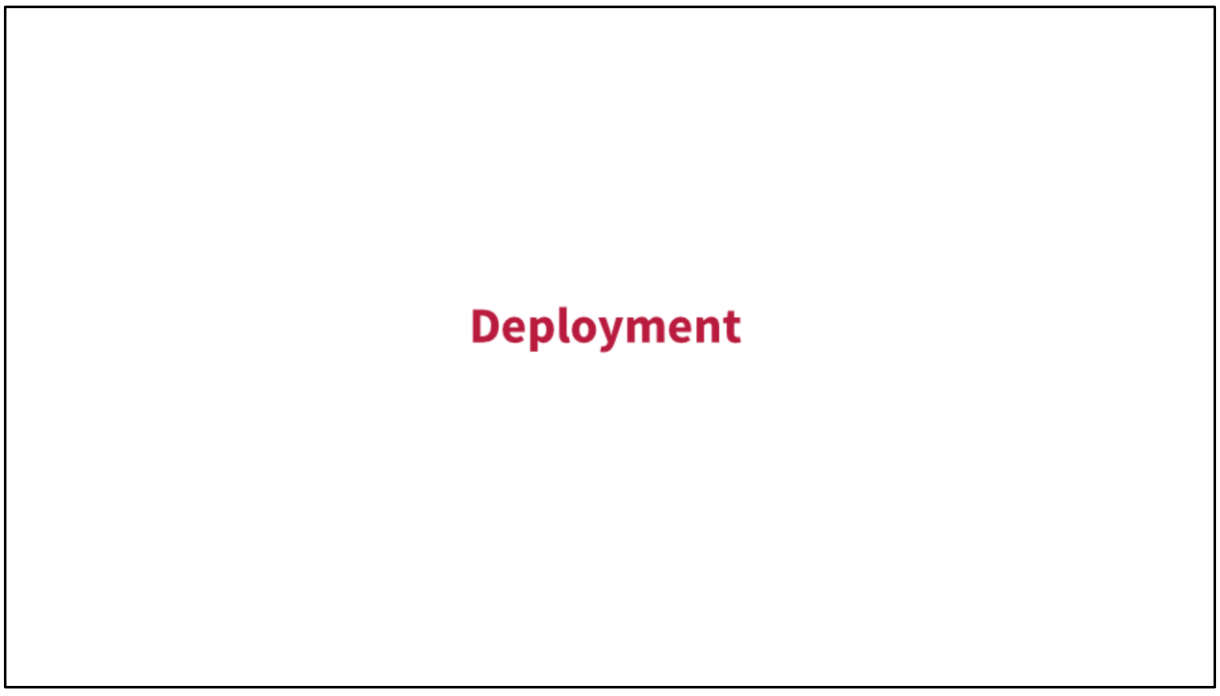
GeneXus 16 incorpora eventos que pueden dispararse antes o después del proceso de Build. Esto permite ejecutar tareas sencillas como vaciar un directorio, o copiar archivos.
Para crear esos comandos ya están disponibles una serie de macros que acceden al path del modelo o al directorio de instalación de GeneXus, si necesidad de hardcodear esos paths.
Los commandos utilizados se salvan en el Environment, o sea que si se comitea las propiedades del Environment en GeneXus Server, estarán siempre disponibles. Cuando se ejecutan estos commandos, puede verse en la ventana de Output.
Más información en: https://wiki.genexus.com/commwiki/servlet/wiki?39474



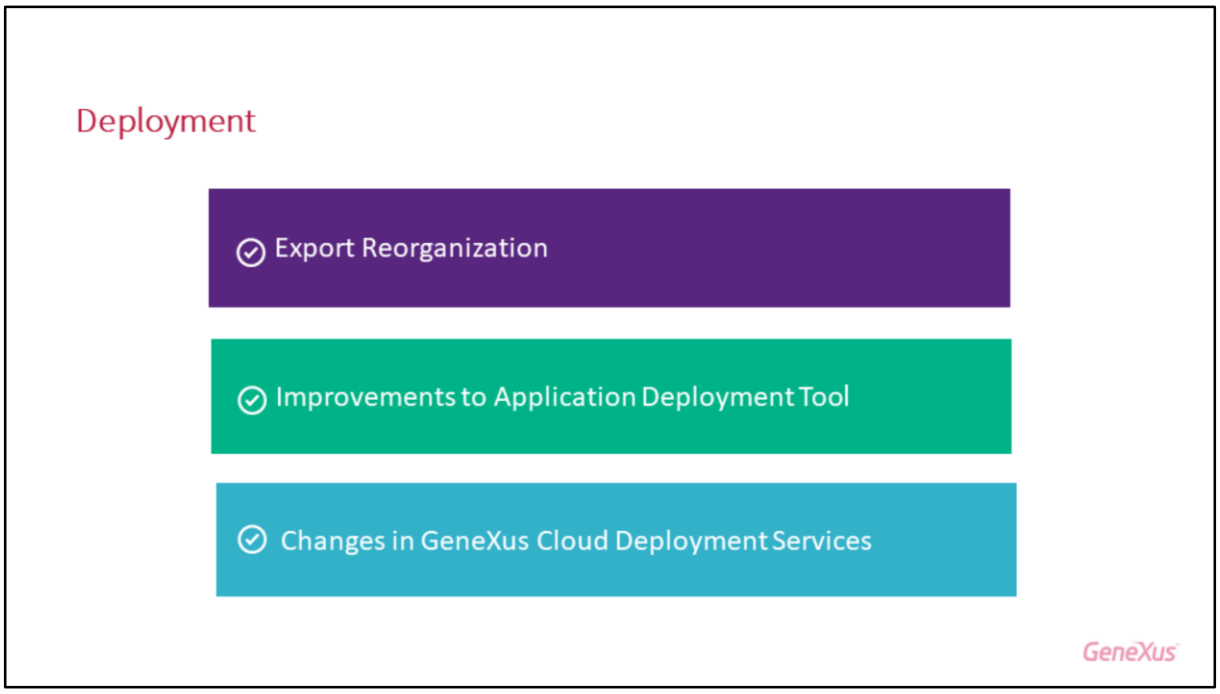
El objeto externo ConfigurationManager permite leer la configuración que está siendo utilizada en tiempo de ejecución, almacenada en una variable de ambiente o en archivos de configuración como el web.config, client.cfg o cloudservices.config.

Dispone de los métodos **HasValue** par saber si una propiedad fue seteada y **GetValue** para obtener el valor de una propiedad. En ambos métodos se le puede pasar el nombre de un archive, el cual es opcional, si no se pone assume unos de los archivos de configuración estandar.
Los nombres de las propiedades son también los mismo usados en estos archivos estandar o en las variables de ambiente, pero es posible crear nuestras propias propiedades y nuestros propios archivos de configuración.

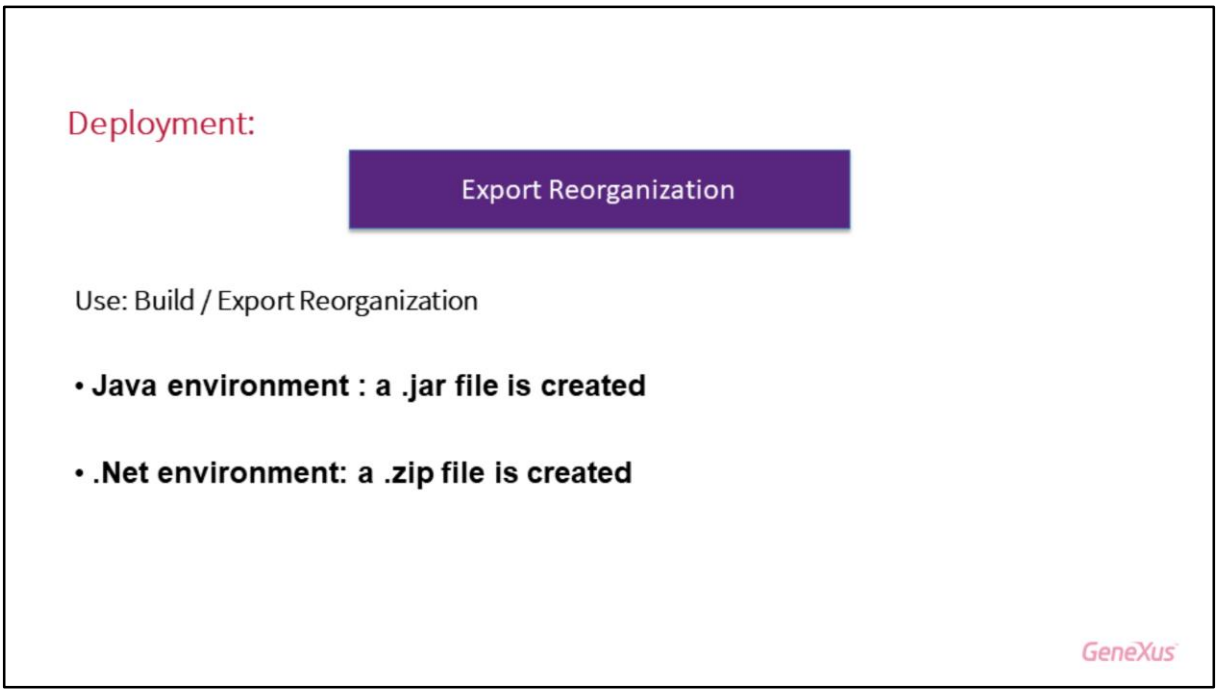
Más información en: https://wiki.genexus.com/commwiki/servlet/wiki?40085



GeneXus 16 agrega funcionalidades al proceso de deployment



GeneXus 16 presenta ventajas en varias partes del proceso de Deployment, como por ejemplo ahora tenemos la posibilidad de hacer deploy de reorganizaciones.
También hay mejoras en la herramienta de Application Deploy Tool, como la posibilidad de poder agrupar desplegables en una unidad de despliegue o realizar el despliegue a un Docker Container o a un servidor Serverless.
También hay novedades en los servicios GeneXus Cloud Deployment, tanto en el deployment automático con F6, como en mejoras en el Cloud Manager.



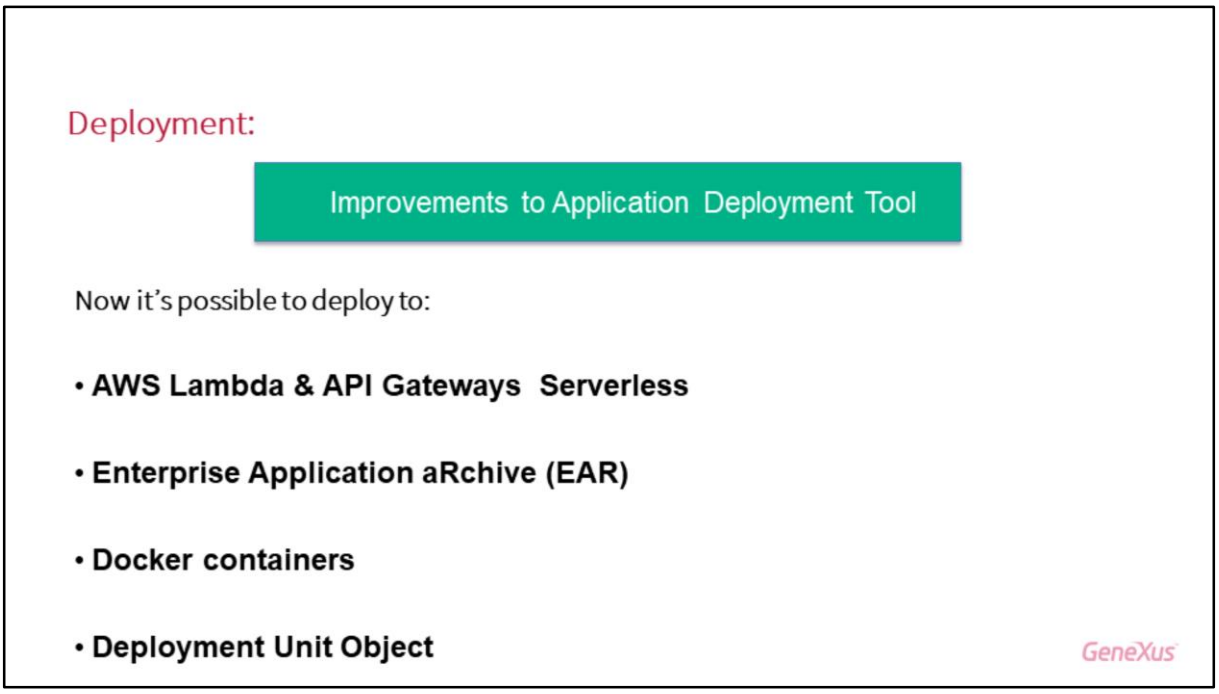
Ahora es posible exportar la reorganización, tanto en Java como en .Net. Usamos Build / Export reorganization.

Esto desplegará los archivos necesarios al servidor para ejecutar la reorganización allí, de modo de actualizar la base de datos con los cambios desde la última reorganización.

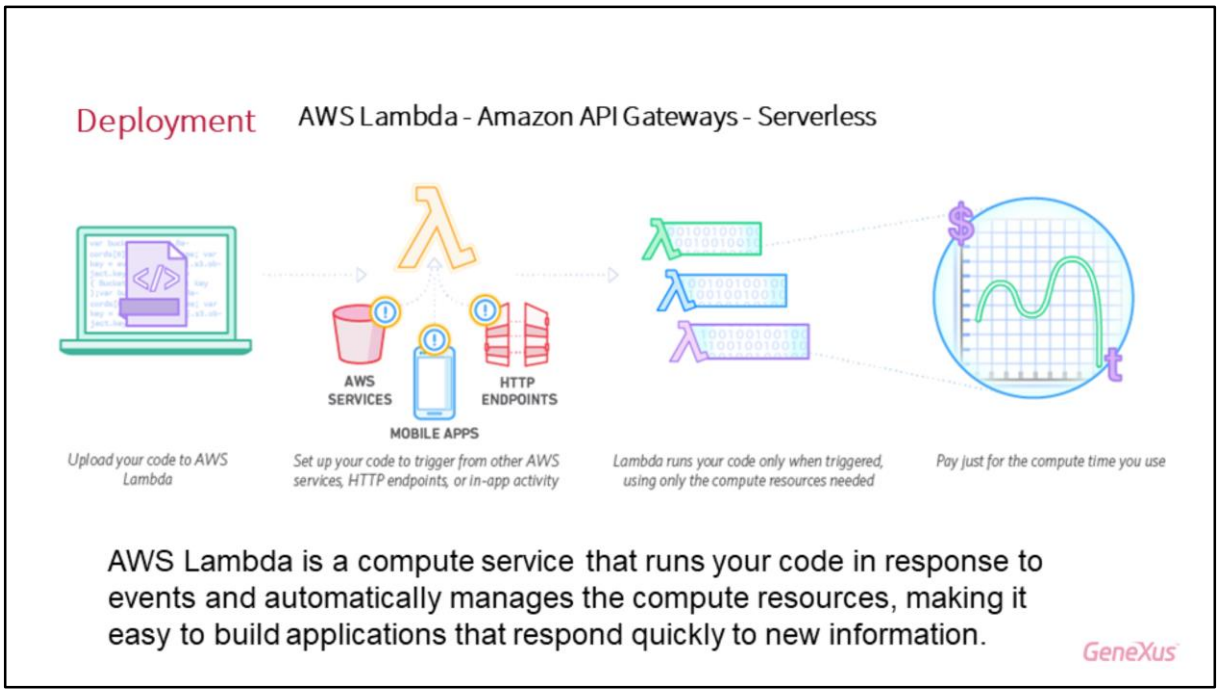
Exportar la reorganización y ejecutarla es parte del proceso de deployment de cualquier aplicación. GeneXus crea (y por defecto ejecuta) los programas de la reorganización, durante el Build All/Rebuild All y específicamente cuando se usa Build / Create Database Tables, o Build / Impact Database Tables.

La opción Export Reorganization que se encuentra en Build / Export Reorganization, permite crear un paquete con los archivos requeridos para ejecutar los programas de reorganización que fueron creados en el ultimo proceso de build. Este paquete puede ser enviado a otro ambiente de ejecución (p.ej. un environment de Producción) para ser ejecutado, a fin de crear o reorganizar los correspondientes esquemas de la Base de datos, desde la última reoganización.
Si es la primera vez, se va a crear la base de datos, las tablas y todas las estructuras, pero si es un reorganización que se hace después porque la aplicación va desarrollándose y es necesario hacer una nueva reorganización, solamente se van a ejecutar en la base de datos los cambios desde la última reorganización.

- En un ambiente Java, la opción Export Reorganization crea un archivo .jar con los programas desde la última reorganización.
- En un ambiente .Net, la opción Export Reorganization crea un archivo .zip con los programas desde la última reorganización.



A continuación detallaremos algunas mejoras realizadas a la herramienta de despliegue.



Cuando subimos una aplicación a un web server, la aplicación de back-end debe responder a **eventos**. Estos eventos pueden ser del cliente, como subir una imagen, o ingresar un pago, o un sensor que envía información, o pueden ser eventos del server, como un timer que corre en el server para ejecutar algo cada cierto tiempo, o cuando se hace un insert a la base de datos.

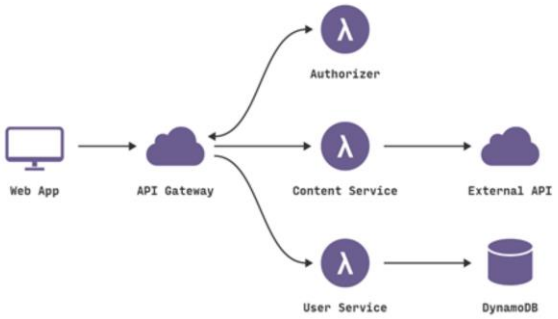
Como la cantidad de estos eventos pueden escalar, se hace necesario monitorear la carga y performance del servidor y además manejar las actualizaciones del sistema operativo, patches de seguridad, etc.

AWS Lambda es un servicio que ejecuta el código que tengamos en el servidor en respuesta a esos eventos, sin tener que realizar tareas de administración, manejando automáticamente los recursos, balanceando carga y publicando métricas de performance.

Además solo se paga por el tiempo informático que se consume en la ejecución. No se cobra nada cuando el código no se está ejecutando.

Deployment

AWS Lambda - Amazon API Gateways - Serverless



Benefits:

- Integrates with AWS Lambda to allow you to create completely serverless APIs
- You create REST APIs that let you call publicly available AWS services through your code running in AWS Lambda
- You pay only for calls made to your APIs and data transfer out.

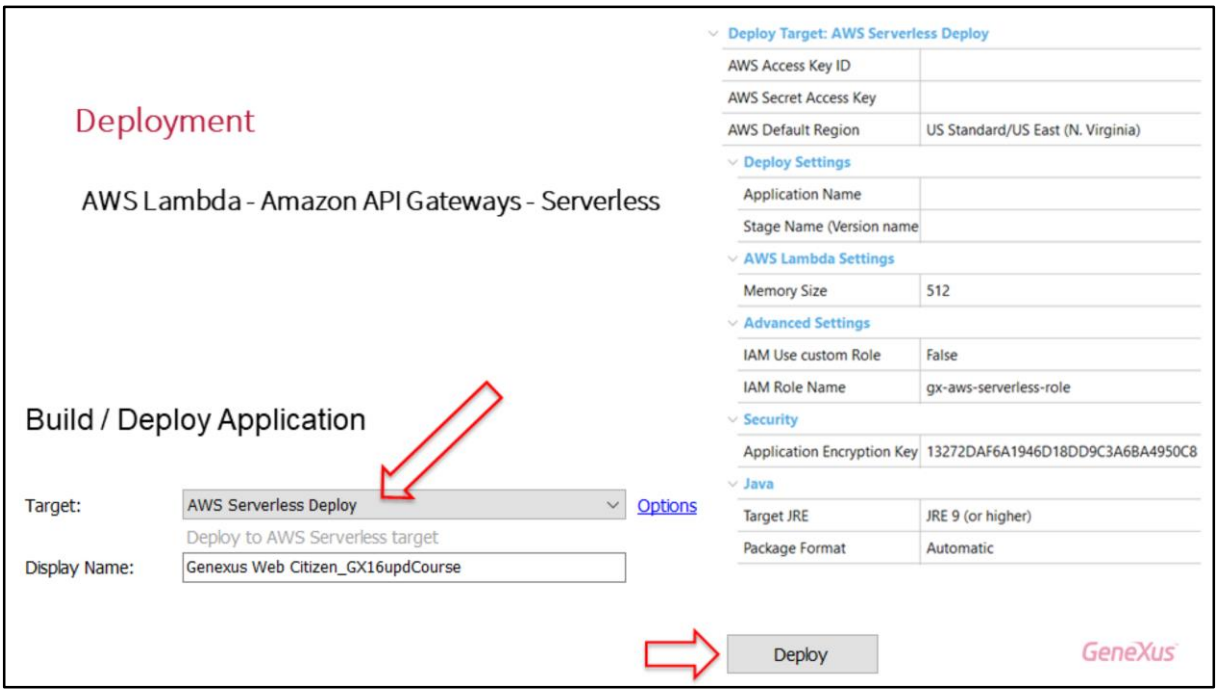
GeneXus

Cuando el código que subimos a AWS Lambda debe responder a eventos que son del cliente, por ejemplo una invocación HTTP, o una invocación desde una app mobile, o un Gateway que direcciona datos de sensores, es necesaria una interfase REST que funcione como “puerta de entrada” para estos requests.

Esta interface REST la provee la API Gateway, quien recibe los request de afuera del server (de los clientes) e invoca internamente a las funciones Lambda, en respuesta a esos eventos del cliente.

Amazon API Gateway se integra a la perfección con AWS Lambda para que pueda crear API sin servidores. Amazon API Gateway le permite crear API REST que sus aplicaciones móviles y web pueden usar para llamar a los servicios de AWS disponibles públicamente mediante el código ejecutado en AWS Lambda.

Amazon API Gateway no requiere pagos mínimos ni costos iniciales. Solo deberá pagar por las llamadas a la API que reciba y el volumen de datos de salida transferido.



Para desplegar en un servidor AWS Lambda, debemos ir a Build / Deploy Application y elegir la opción: AWS Serverless Deploy

Luego debemos completar la AWS Access Key Id, AWS Secret Access Key y AWS Default Region. Y también la Application Name y Stage Name (version).

Y presionar Deploy.

Al finalizer, GeneXus mostrará en la ventana de Output la URL final.

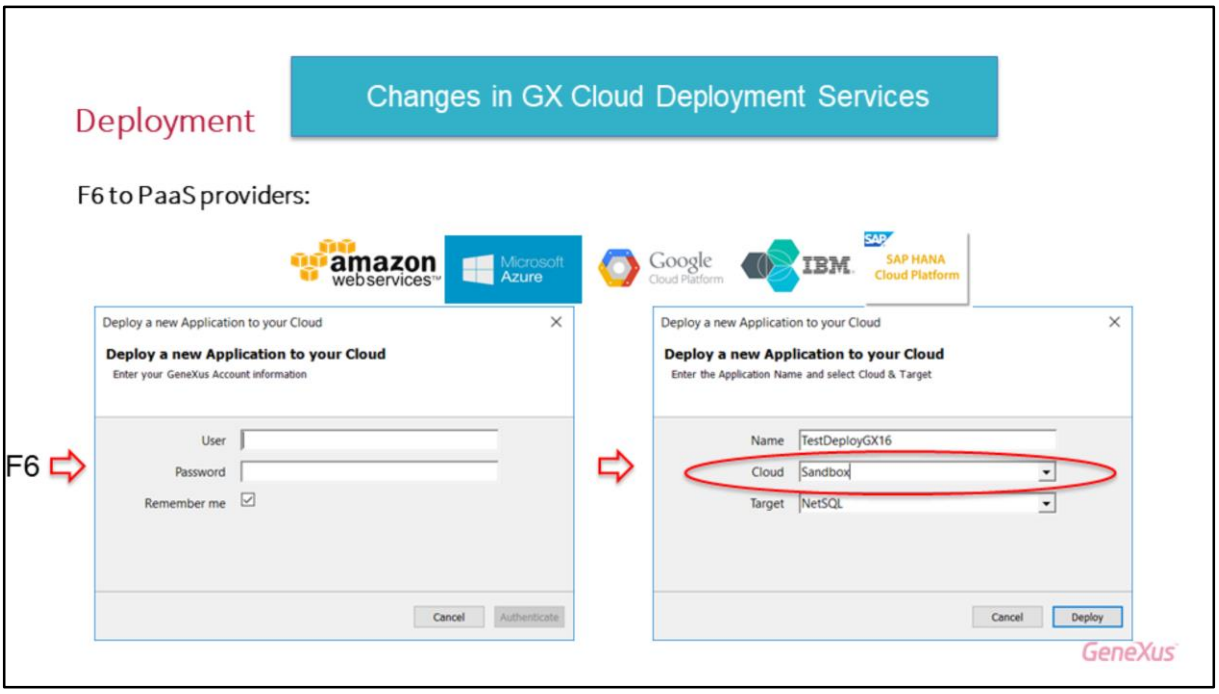
Deployment

Enterprise Application aRchive (EAR)

- In previous version there was a EAR Deployment Wizard (part of Deployment Wizard)
- Now is generated by the Application Deployment Tool if some selected object (or a called one) is an Enterprise Java Beans
- The .EAR file created will have:
  - One web application with all the servlets and static contents
  - One EJB application with all the EJB defined
  - All the JAR files required by the application

GeneXus

Un archivo Enterprise Java Beans es un componente del servidor, en una aplicación Java 2 Enterprise Edition (J2EE).



Cuando se presiona F6 para hacer el deploy a la nube, en primer lugar se deben ingresar las credenciales de GeneXus .

Si se contrata el servicio de GeneXus, el personal técnico arma el ambiente y asocia las credenciales del usuario GeneXus Account a dicho ambiente. Cuando esa persona se loguea, se abre el cuadro de diálogo mostrando el nombre del ambiente preparado.

Este ambiente hasta ahora podia ser armado únicamente en un servidor IaaS, pero ahora también puede ser en un servidor Paas.

Si no se contrató el servicio de GeneXus, el único ambiente disponible es el Sandbox.

Deployment to a Docker Container

Package/Release/Deploy

Deployment Unit Object
Build events
Serverless (deploy to AWS lambda)
GX Cloud Deployment Services → F6 PAAS
Containers (Docker)

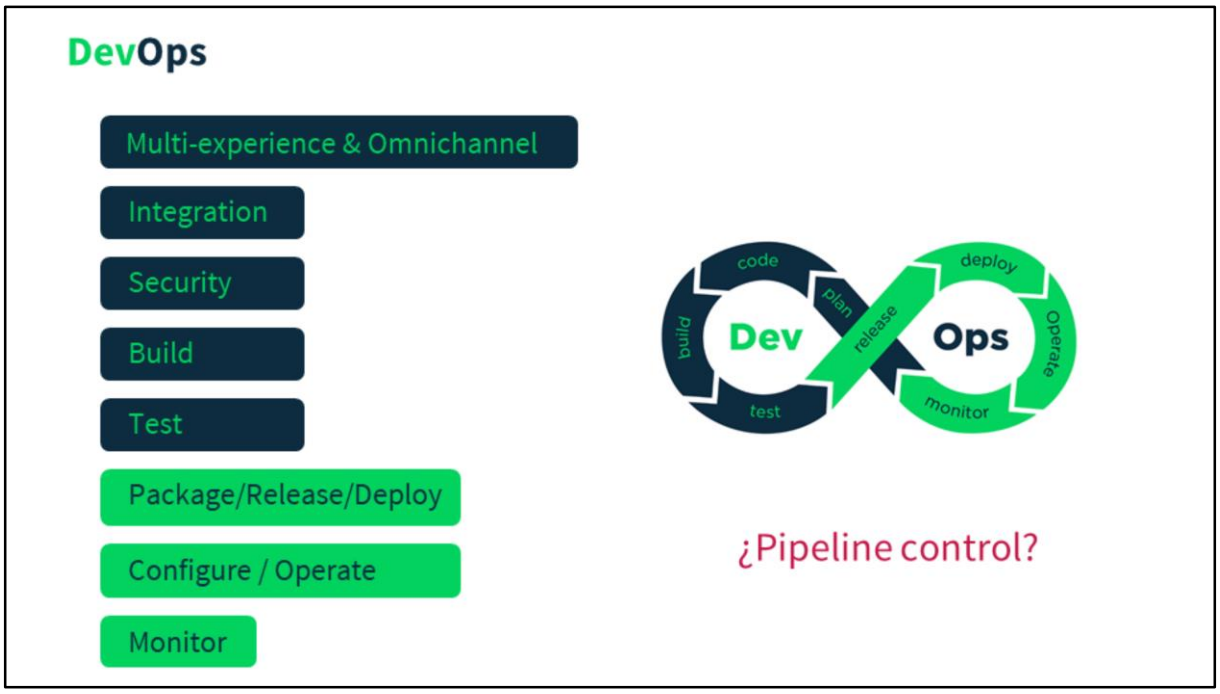
Configure / Operate

ConfigurationManager API
Config via Environment variables

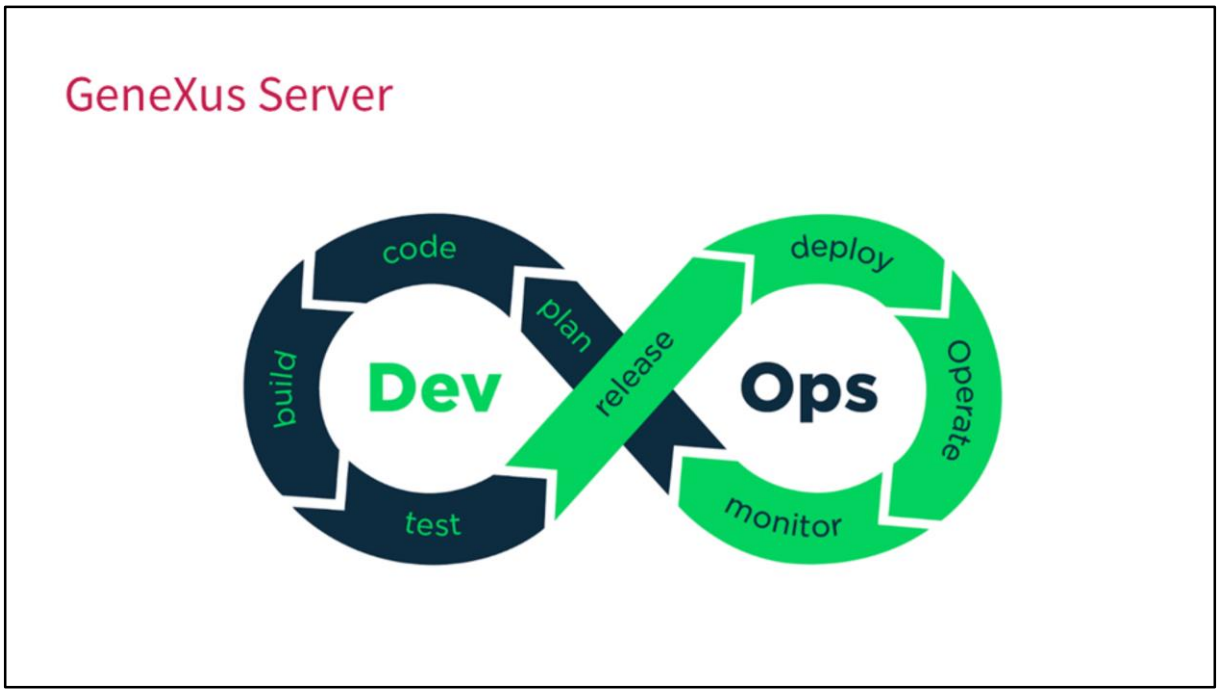
Monitor

Log API

Sobre las apis de configuración y la Api Log no tenemos más para agregar que lo dicho en la introducción. Pero...



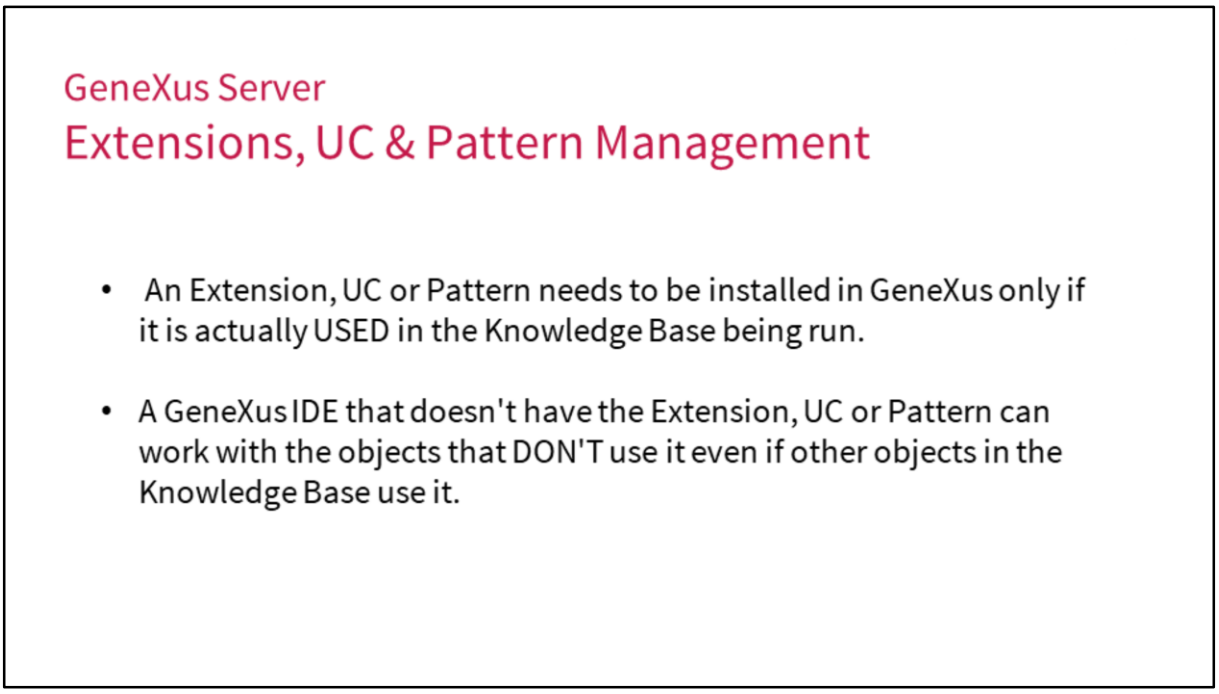
Hemos visto cómo GeneXus nos ayuda en cada una de las etapas del ciclo de DevOps,
Ahora … ¿cómo hacemos para facilitar el flujo de este ciclo?



Primero que nada vamos a ver ahora algunas novedades en GXServer que es una pieza clave – cuando trabajamos en equipo - para poder integrar el código de forma sencilla

En esta version hay mejoras que se orientan a simplificar el tema del manejo y mantenimiento de las instalaciones del server.

Desde el U2 de la Gx15 es possible actualizer una instalacion de una instancia de GeneXus Server diferente a la instancia x defecto



Otra mejora tiene que ver con el manejo de extensiones, UC y patterns PARA SIMPLIFICAR EL TEMA DE INSTALACION DE LOS MISMOS EN EL PROPIO SERVER Y TODOS LOS GENEXUS IDE QUE INTERACTUAN CON EL SERVER:

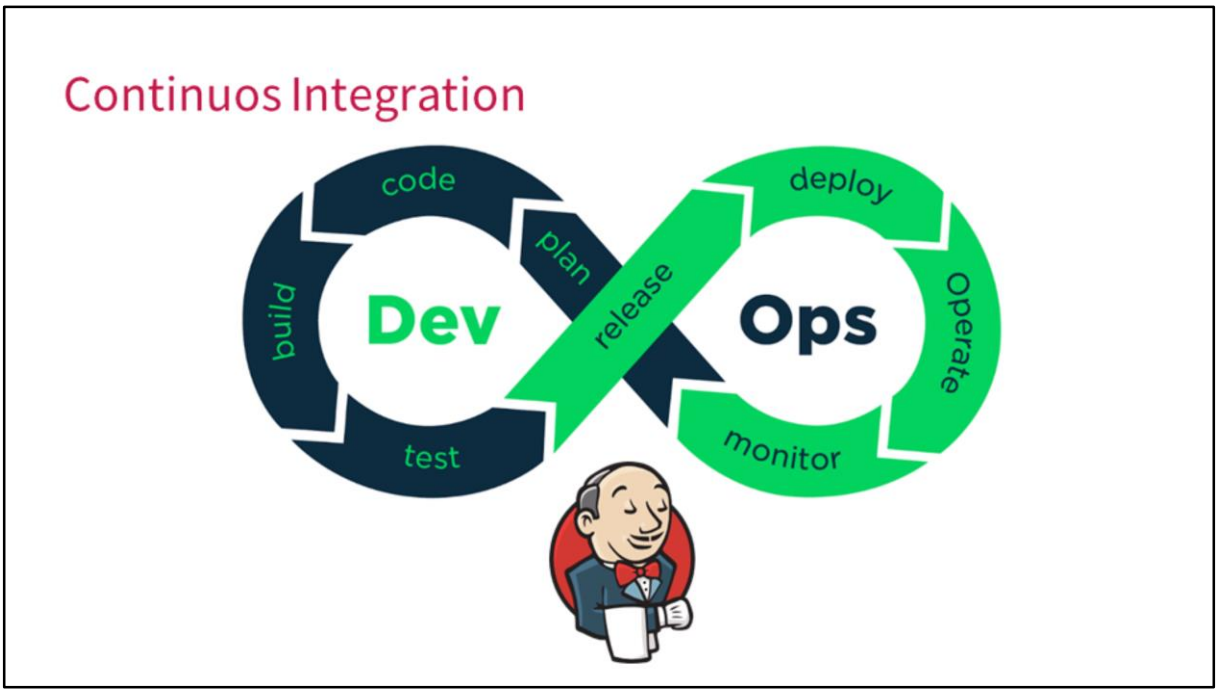
Cuando trabajamos en una KB que usa alguna Extensión, al subir esa KB al server necesitamos que esa Extensión este disponible en el Server – eso es xq las extensiones
Definen una parte nueva para objetos existentes y para que el objeto se pueda subir al server, el mismo necesita conocer esas partes nuevas.

Ahora bien ANTES, si yo tenia en mi instalación de Genexus una extensión que NO usaba en ningún objeto de la KB, y subia esa KB al server igual el server necesitaba
Tener la extensión. Y si el Server tenia la extensión, cualquier instancia de Genexus que bajara una KB de ese server también necesitaba tener la extensión. Por tanto, si en una KB se usaba la extensión, era necesario que todos los GeneXus que usaban alguna KB hosteada en ese server, tuvieran la extensión instalada también.

Desde la 15 u3 estas partes definidas por las extensiones se marcan como opcionales (automáticamente) y son enviadas a GeneXus Server solamente cuando es estrictamente necesario, esto implica que:
• Solo se necesita tener una Extensión instalada en GeneXus si efectivamente la misma es UTILIZADA en la Base de conocimiento en la que se está trabajando.
• Si una Extensión está instalada en GeneXus Server porque una de las Bases de conocimiento almacenadas la utiliza, se pueden tener otras Bases de conocimiento que no utilicen la Extensión y que no se generen referencias a la misma, permitiendo que la extensión no tenga que estar instalada en un IDE GeneXus de desarrollo que no necesita la misma.
• Un IDE GeneXus que no tenga la Extensión, puede trabajar con los objetos que NO utilizan la misma por más que otros objetos de la Base de conocimiento la usen.

Este es el caso por ejemplo de K2B, cuya extensión K2B Object Designer define una parte nueva en los objetos Web Panels. Si en una instancia de GeneXus Server se tiene instalada esta Extensión, entonces ya no es necesario que todos los IDE GeneXus que trabajen con Bases de conocimiento que no utilizan la Extensión, tengan la misma instalada y además, un IDE GeneXus que no tenga instalada la Extensión puede trabajar con aquellos objetos de la Base de

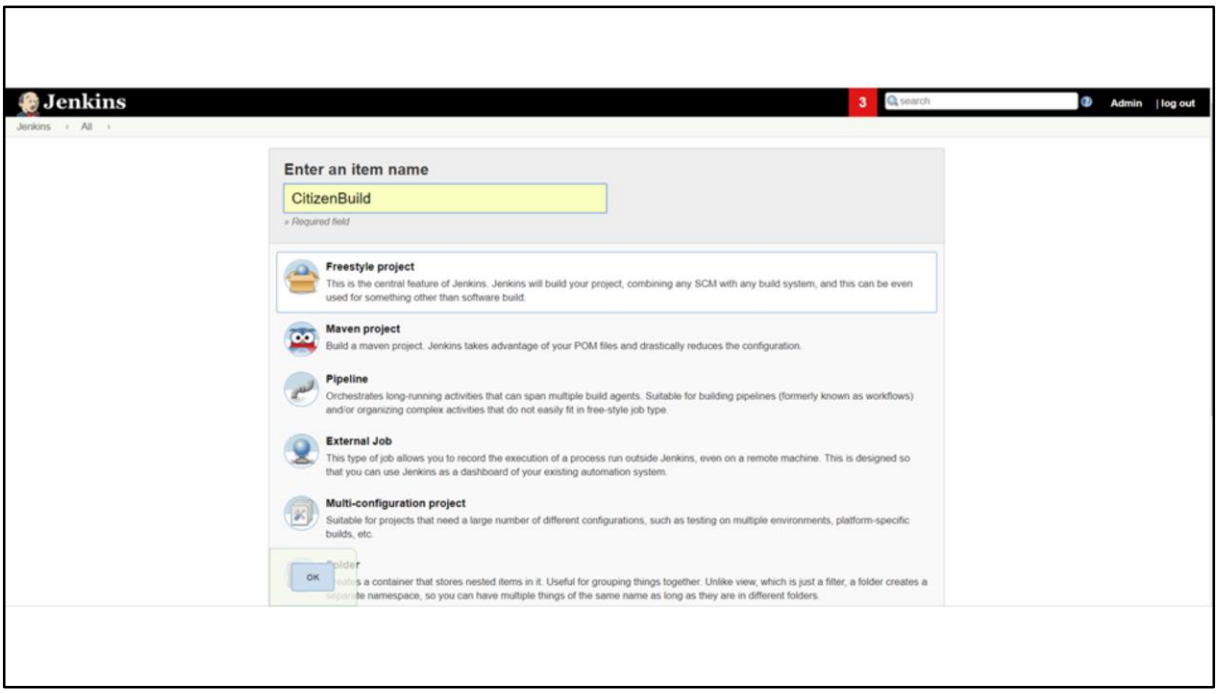
conocimiento que no utilicen la misma.

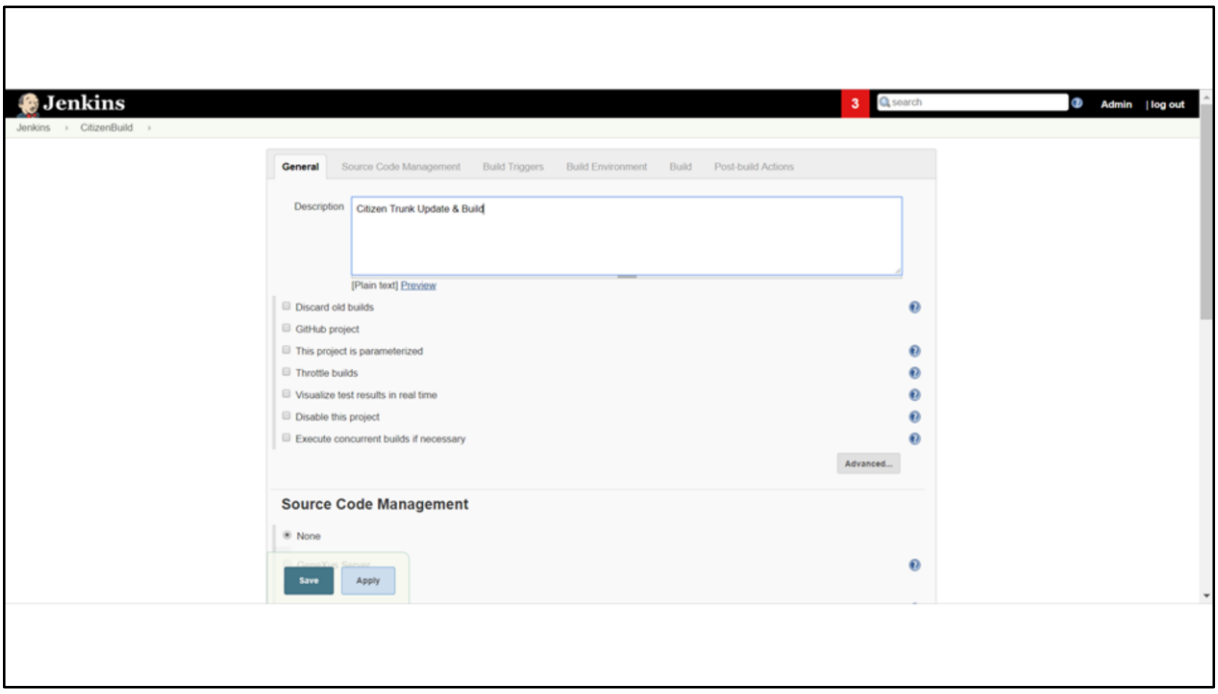


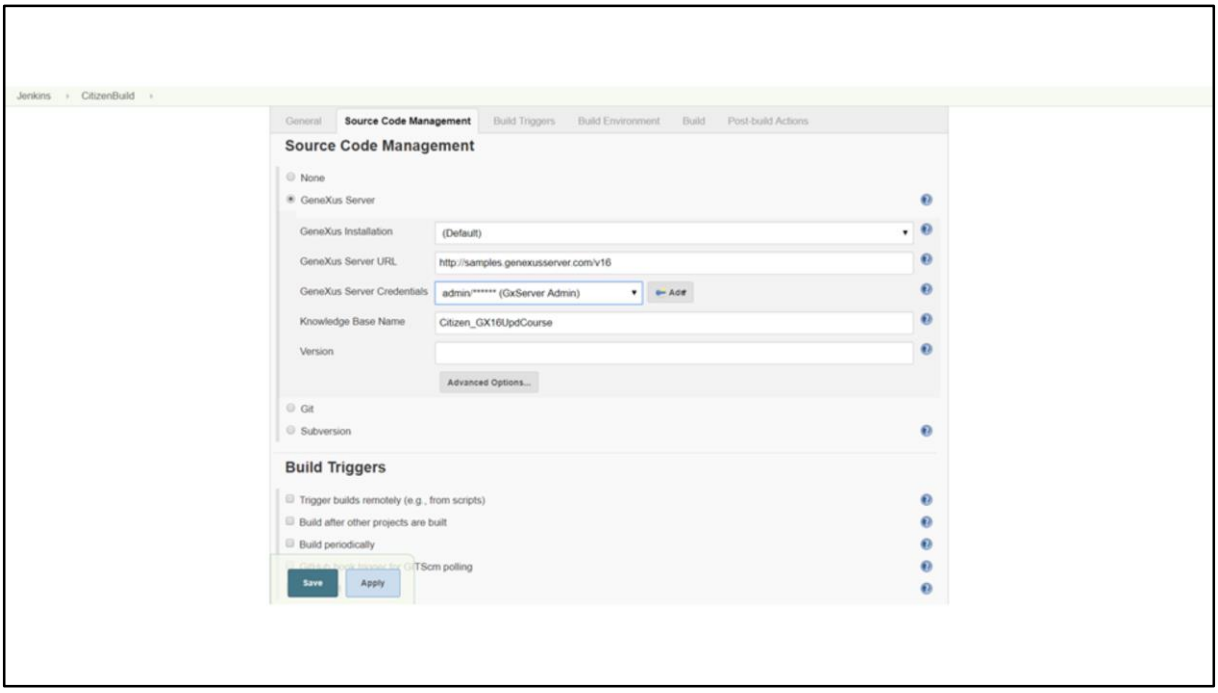
Hablamos del Server que es una herramienta fundamental en simplificar la integracion del codigo, pero para realmente poder alcanzar un flujo de DevOps necesitamos
Comenzar implementando una integracion continua. Una novedad en esta version es que ademas de Cruise-Control ahora GeneXus se integra Tambien con Jenkins como motor de Integracion continua.

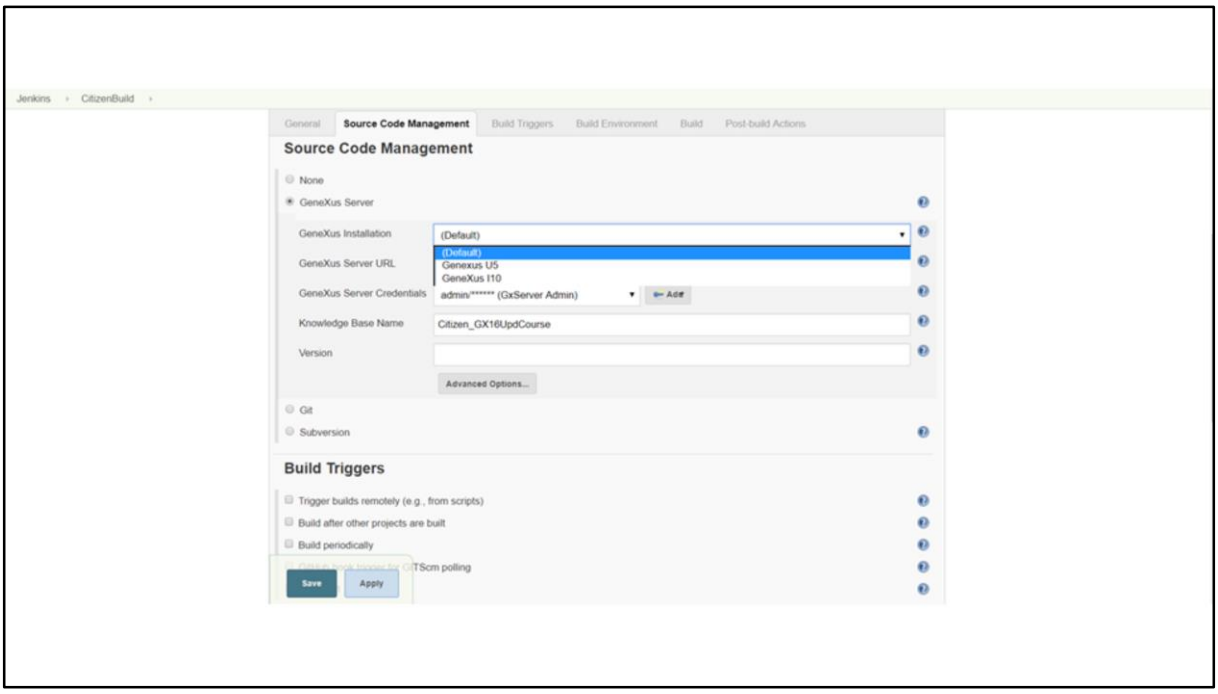
De esta manera se puede automatizar el caso ‘basico’ que seria que cuando hay un cambio en el codigo, se dispare automaticamente un build

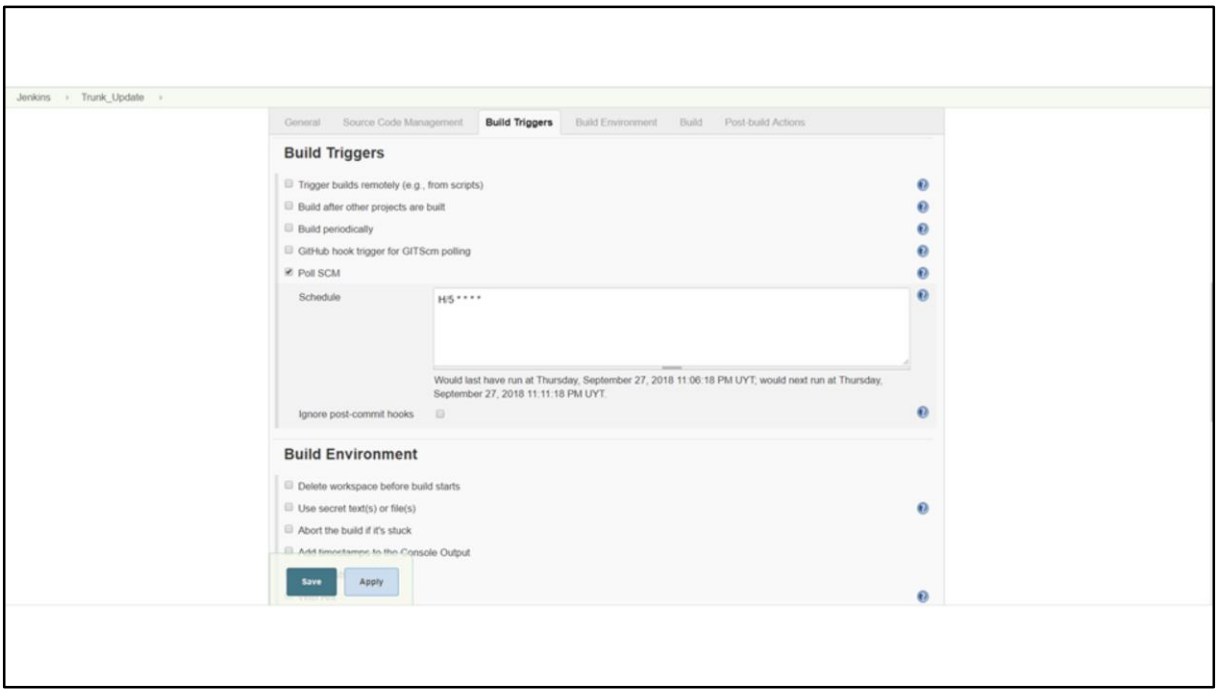

Jenkins
Jenkins
New Item
People
Build History
Manage Jenkins
My Views
Credentials
New View
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle

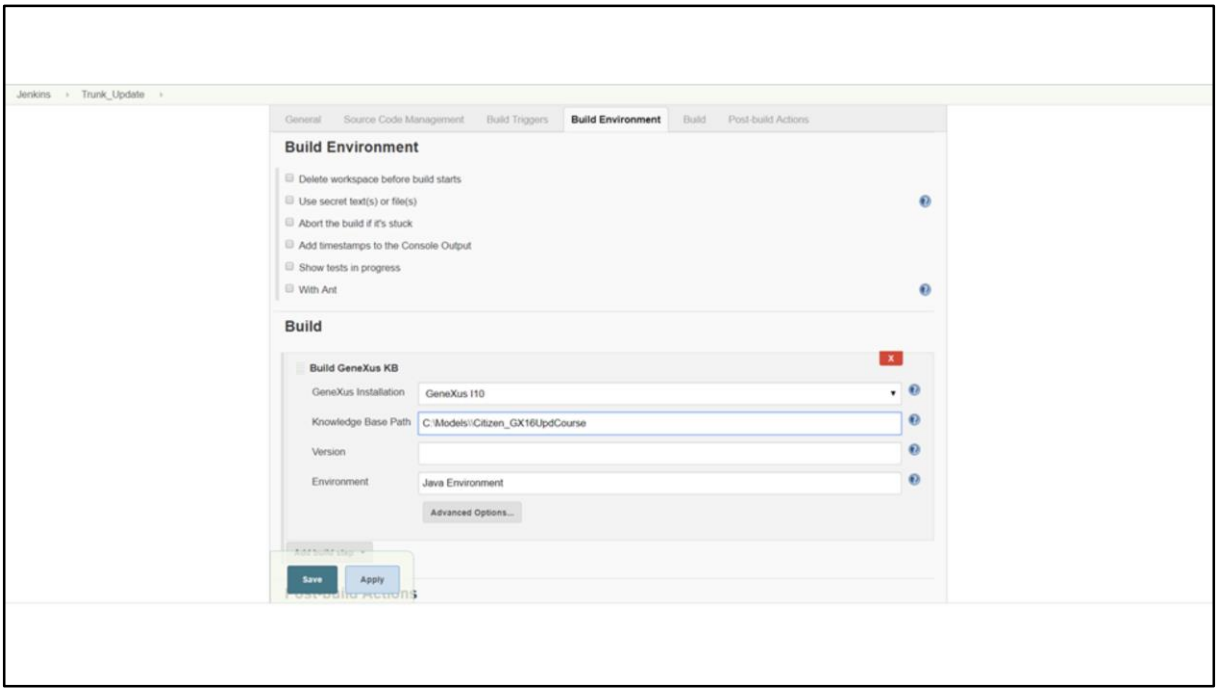

Jenkins
3
search
Admin
| log out
Jenkins › All ›
Enter an item name
CitizenBuild
» Required field
Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
External Job
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
Folder
a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a
e namespace, so you can have multiple things of the same name as long as they are in different folders.
OK

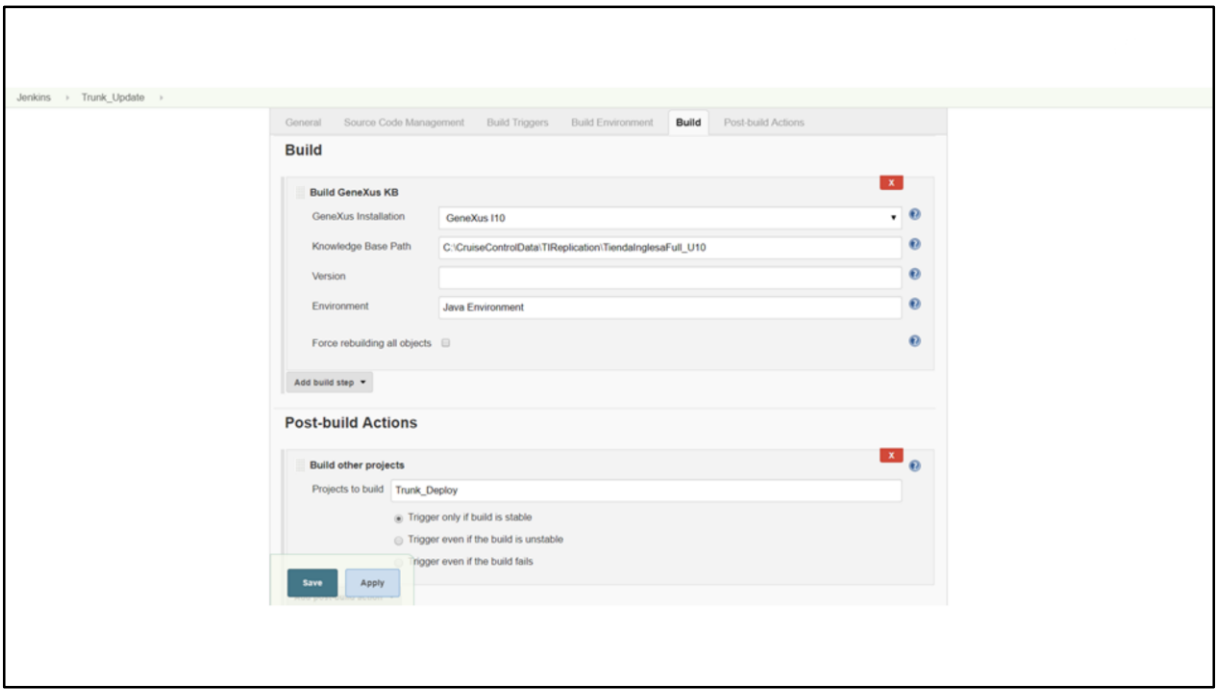

Jenkins
3
search
Admin
| log out
Jenkins › CitizenBuild ›
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Description
Citizen Trunk Update & Build
[Plain text] Preview
Discard old builds
GitHub project
This project is parameterized
Throttle builds
Visualize test results in real time
Disable this project
Execute concurrent builds if necessary
Advanced...
Source Code Management
None
Save
Apply

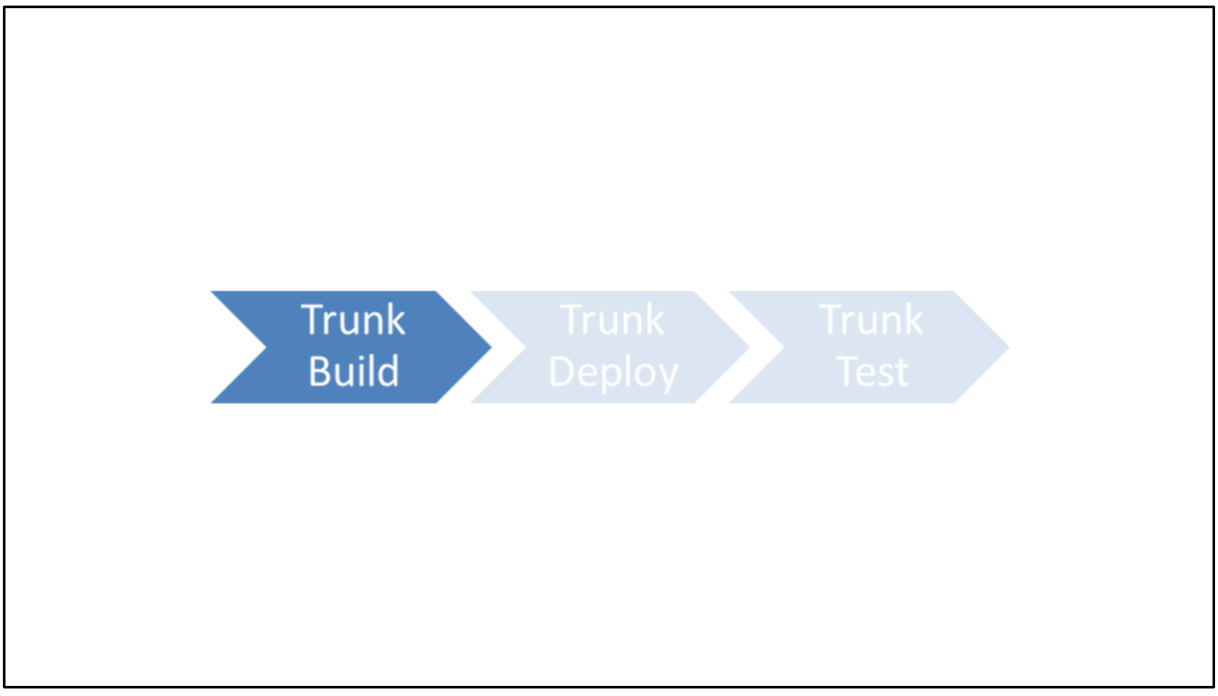

Jenkins › CitizenBuild ›
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Source Code Management
None
GeneXus Server
GeneXus Installation
(Default)
GeneXus Server URL
http://samples.genexusserver.com/v16
GeneXus Server Credentials
admin/****** (GxServer Admin)
Add
Knowledge Base Name
Citizen_GX16UpdCourse
Version
Advanced Options...
Git
Subversion
Build Triggers
Trigger builds remotely (e.g., from scripts)
Build after other projects are built
Build periodically
Save
Apply

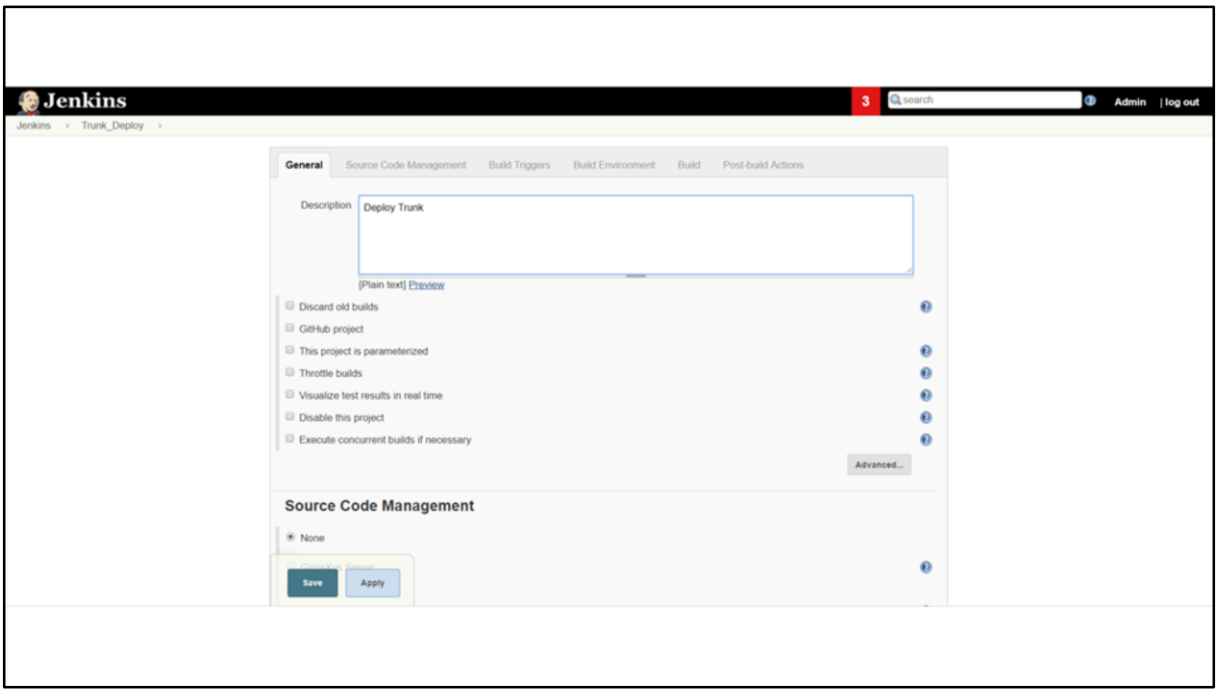

Jenkins › CitizenBuild ›
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Source Code Management
None
GeneXus Server
GeneXus Installation
(Default)
(Default)
Genexus U5
GeneXus I10
GeneXus Server URL
GeneXus Server Credentials
admin/****** (GxServer Admin)
Add
Knowledge Base Name
Citizen_GX16UpdCourse
Version
Advanced Options...
Git
Subversion
Build Triggers
Trigger builds remotely (e.g., from scripts)
Build after other projects are built
Build periodically
SCM polling
Save
Apply

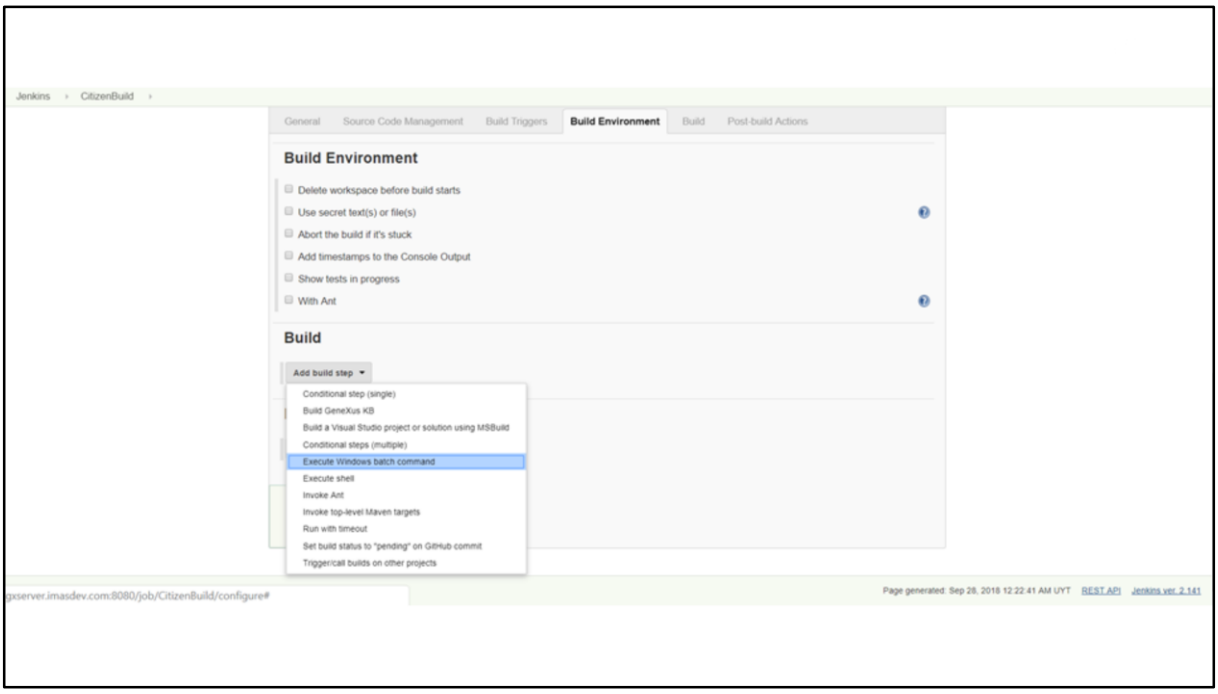

Jenkins › Trunk_Update ›
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Build Triggers
Trigger builds remotely (e.g., from scripts)
Build after other projects are built
Build periodically
GitHub hook trigger for GITScm polling
Poll SCM
Schedule
H/5 * * * *
Would last have run at Thursday, September 27, 2018 11:06:18 PM UYT; would next run at Thursday, September 27, 2018 11:11:18 PM UYT.
Ignore post-commit hooks
Build Environment
Delete workspace before build starts
Use secret text(s) or file(s)
Abort the build if it's stuck
Add timestamps to the Console Output
Save
Apply

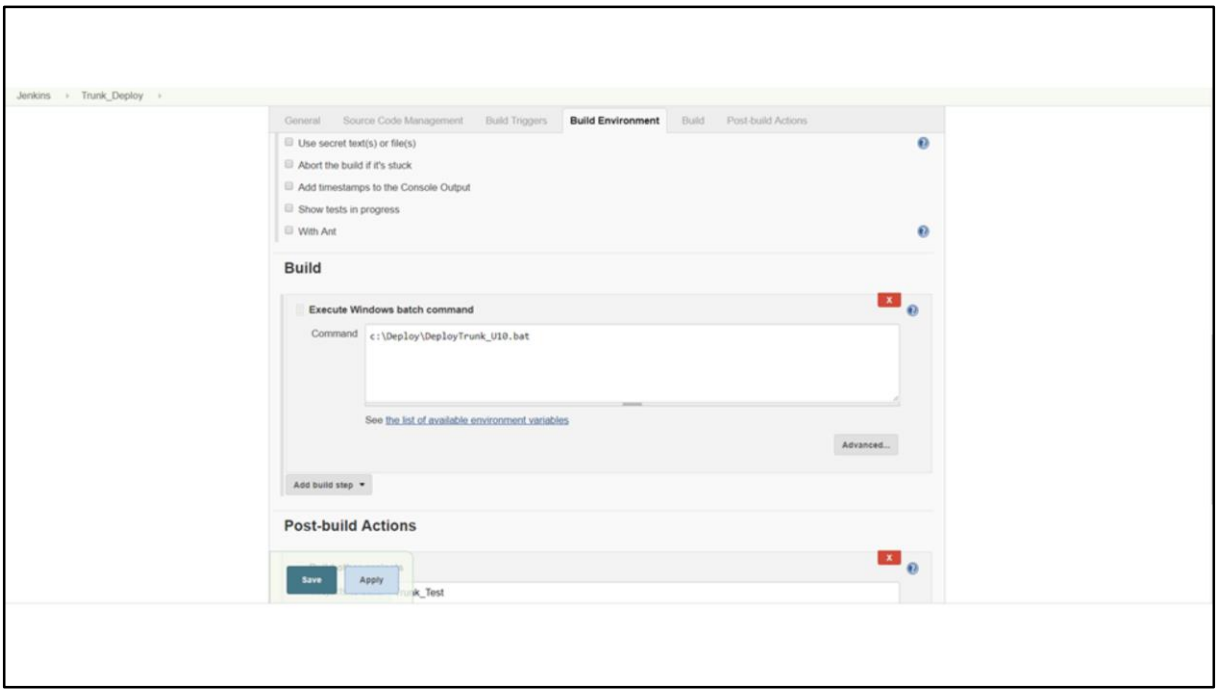

Jenkins › Trunk_Update ›
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Build Environment
Delete workspace before build starts
Use secret text(s) or file(s)
Abort the build if it's stuck
Add timestamps to the Console Output
Show tests in progress
With Ant
Build
Build GeneXus KB
GeneXus Installation
GeneXus I10
Knowledge Base Path
C:\Models\Citizen_GX16UpdCourse
Version
Environment
Java Environment
Advanced Options...
Save
Apply

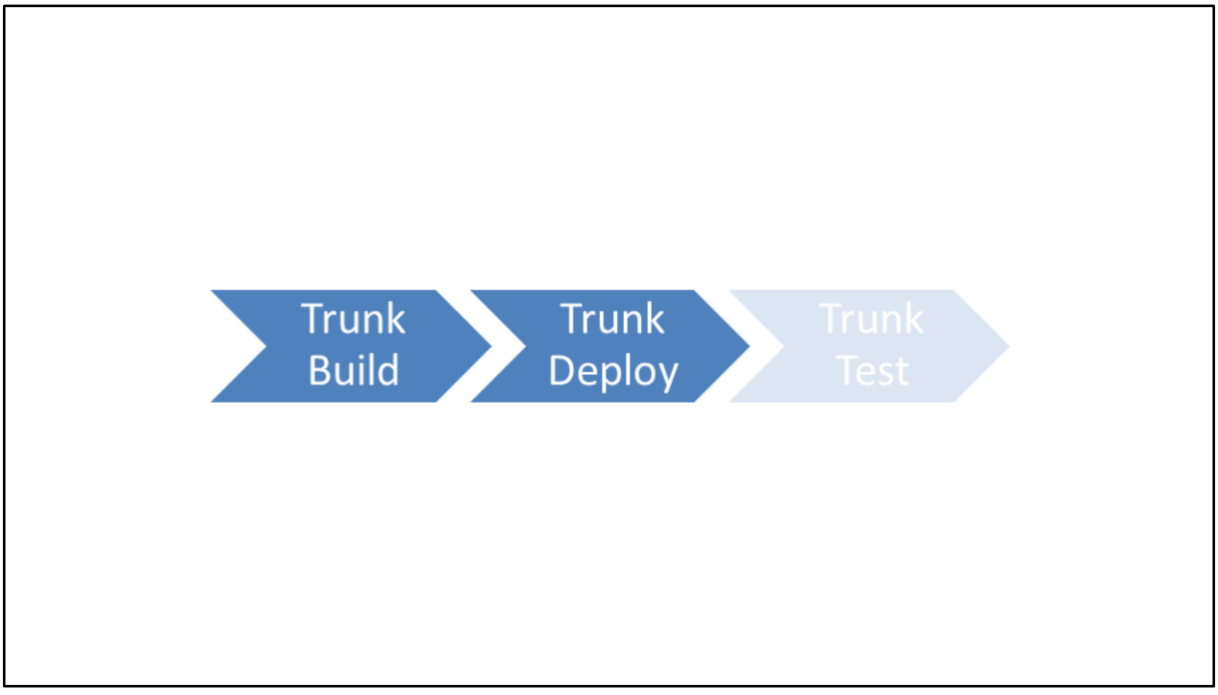

Jenkins › Trunk_Update ›
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Build
Build GeneXus KB
GeneXus Installation
GeneXus I10
Knowledge Base Path
C:\CruiseControlData\TIReplication\TiendaInglesaFull_U10
Version
Environment
Java Environment
Force rebuilding all objects
Add build step
Post-build Actions
Build other projects
Projects to build
Trunk_Deploy
Trigger only if build is stable
Trigger even if the build is unstable
Save
Apply

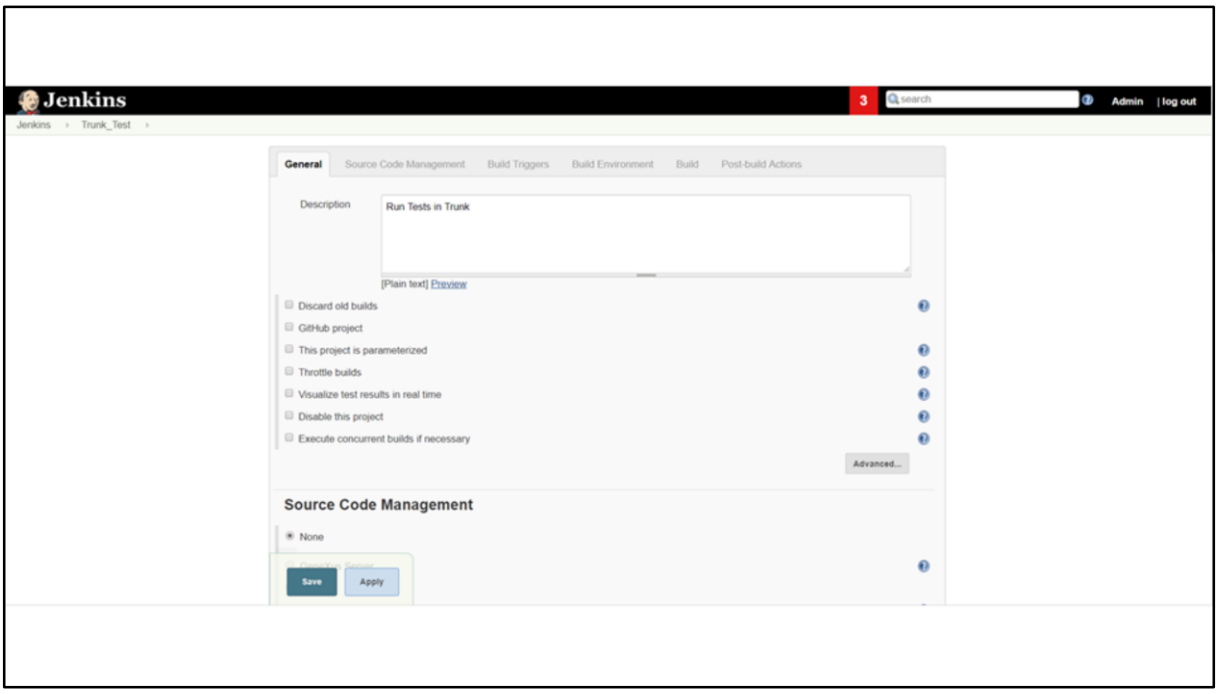

Trunk
Build
Trunk
Deploy
Trunk
Test

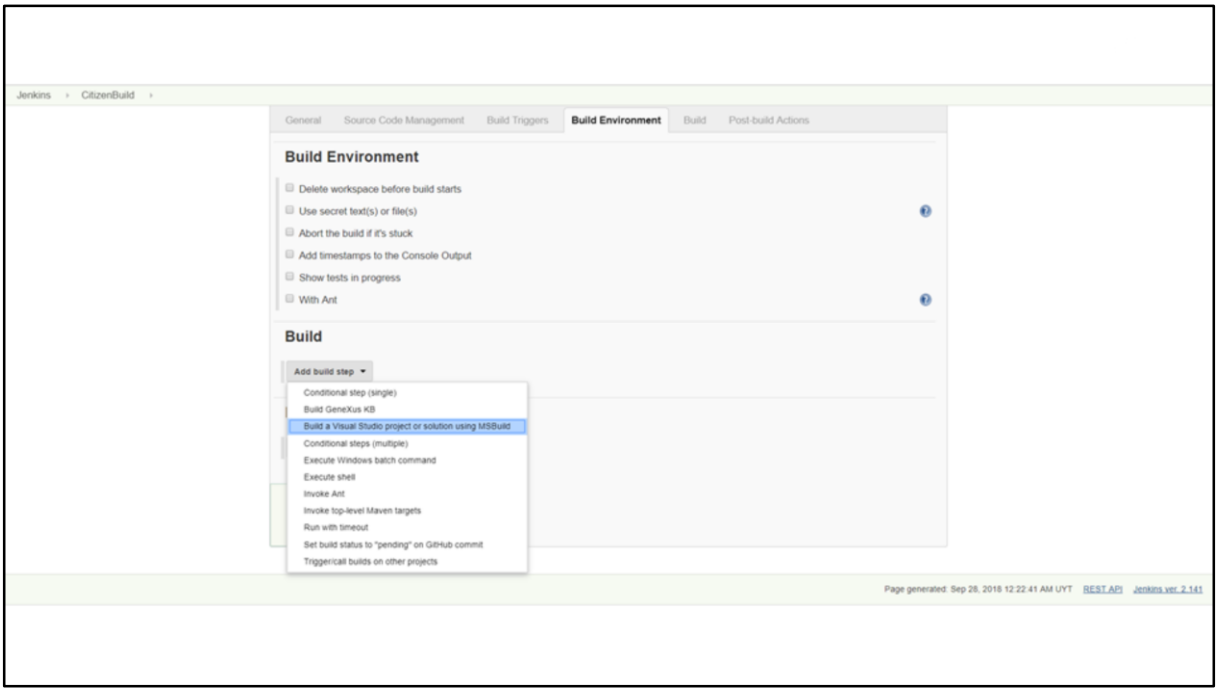

Jenkins
3
search
Admin
| log out
Jenkins › Trunk_Deploy ›
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Description
Deploy Trunk
[Plain text] Preview
Discard old builds
GitHub project
This project is parameterized
Throttle builds
Visualize test results in real time
Disable this project
Execute concurrent builds if necessary
Advanced...
Source Code Management
None
Save
Apply

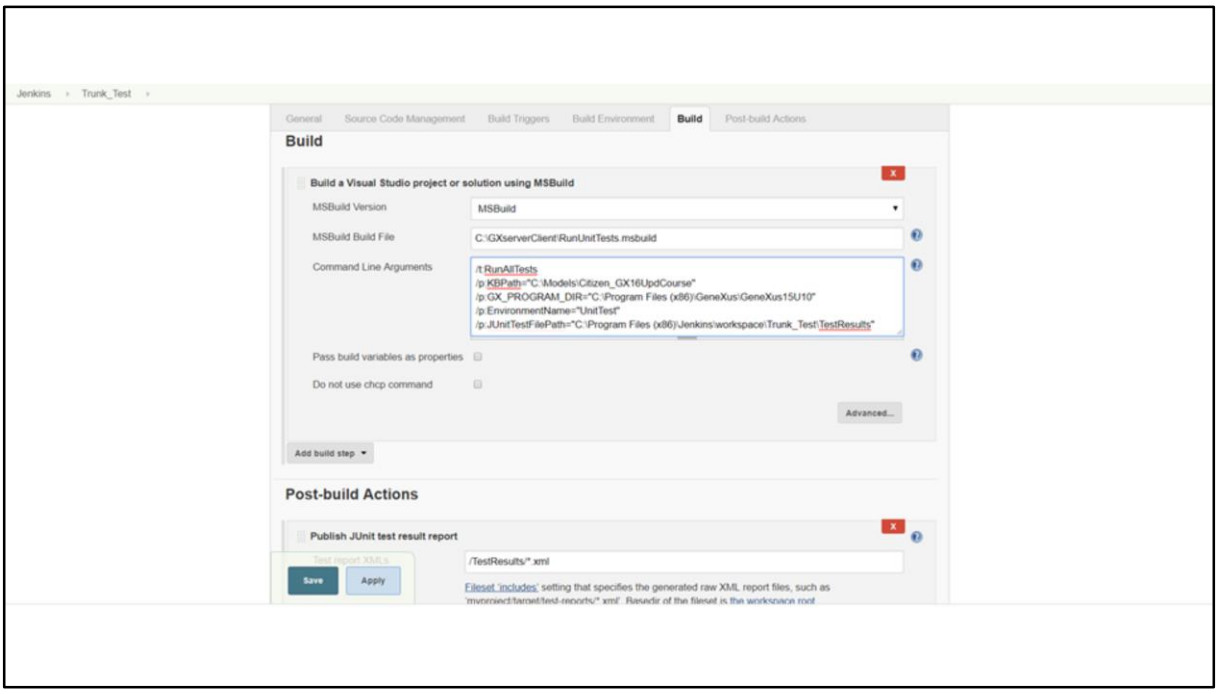

Jenkins › CitizenBuild ›
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Build Environment
Delete workspace before build starts
Use secret text(s) or file(s)
Abort the build if it's stuck
Add timestamps to the Console Output
Show tests in progress
With Ant
Build
Add build step
Conditional step (single)
Build GeneXus KB
Build a Visual Studio project or solution using MSBuild
Conditional steps (multiple)
Execute Windows batch command
Execute shell
Invoke Ant
Invoke top-level Maven targets
Run with timeout
Set build status to "pending" on GitHub commit
Trigger/call builds on other projects
gxserver.imasdev.com:8080/job/CitizenBuild/configure#
Page generated: Sep 28, 2018 12:22:41 AM UYT
REST API
Jenkins ver. 2.141

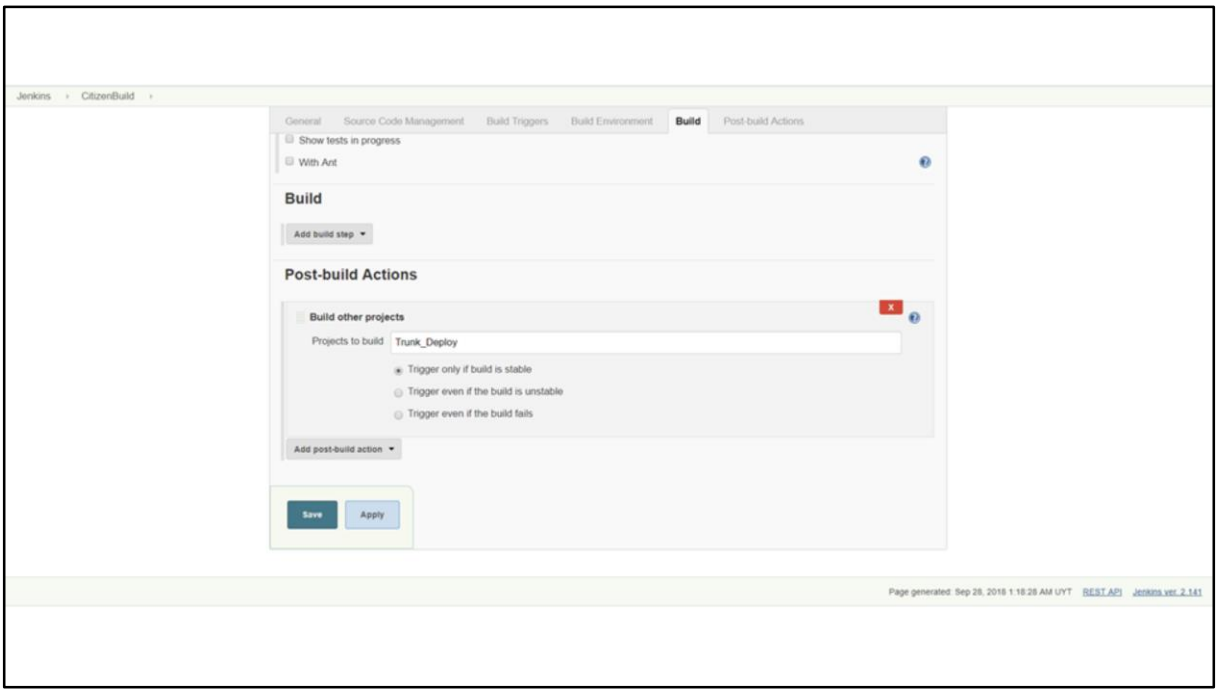

Jenkins › Trunk_Deploy ›
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Use secret text(s) or file(s)
Abort the build if it's stuck
Add timestamps to the Console Output
Show tests in progress
With Ant
Build
Execute Windows batch command
Command
c:\Deploy\DeployTrunk_U10.bat
See the list of available environment variables
Advanced...
Add build step
Post-build Actions
Save
Apply

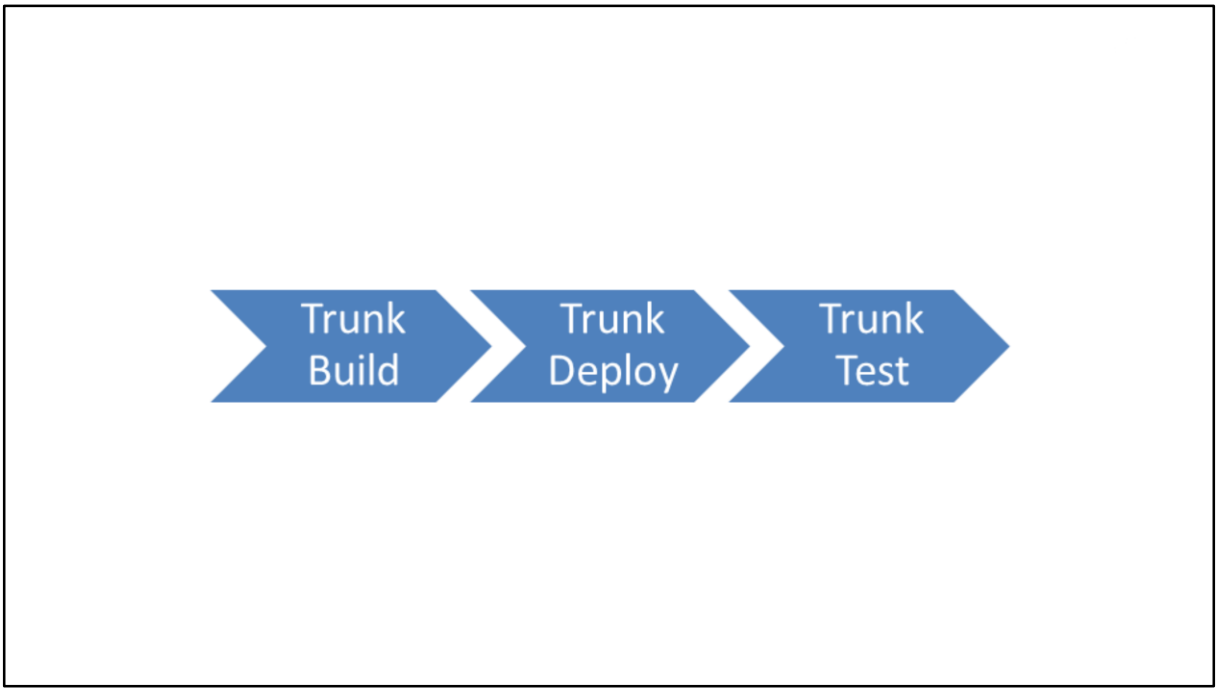

Trunk
Build
Trunk
Deploy
Trunk
Test

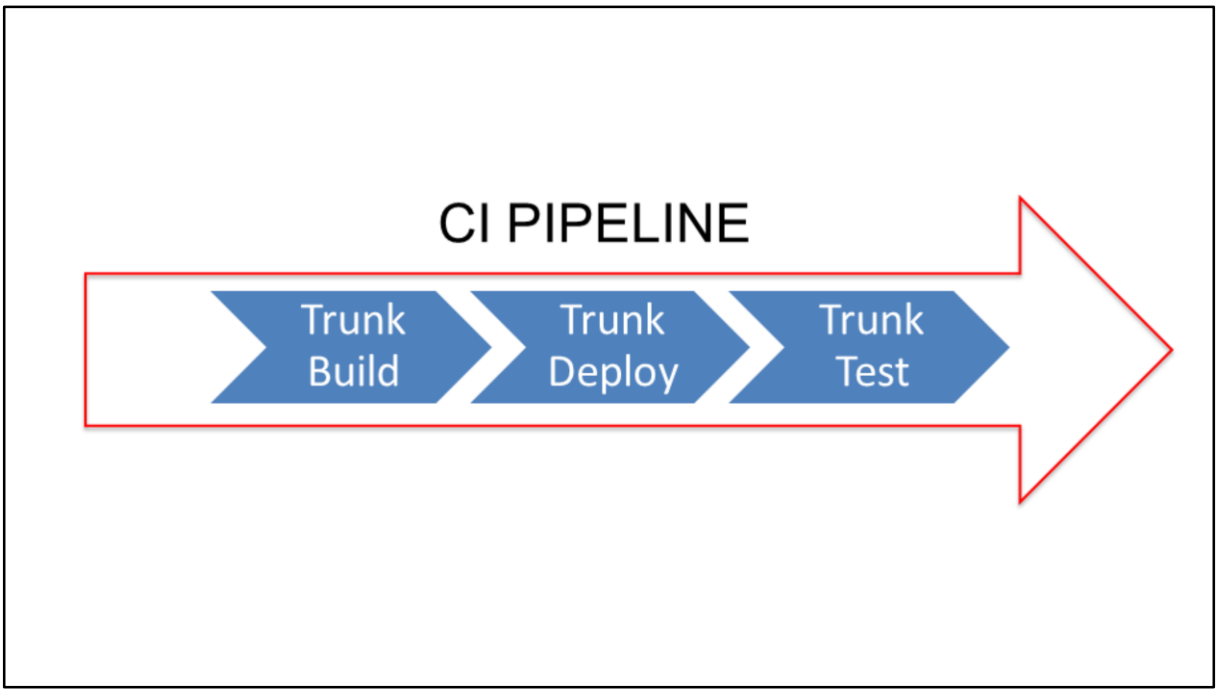

Jenkins
3
search
Admin
| log out
Jenkins › Trunk_Test ›
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Description
Run Tests in Trunk
[Plain text] Preview
Discard old builds
GitHub project
This project is parameterized
Throttle builds
Visualize test results in real time
Disable this project
Execute concurrent builds if necessary
Advanced...
Source Code Management
None
Save
Apply

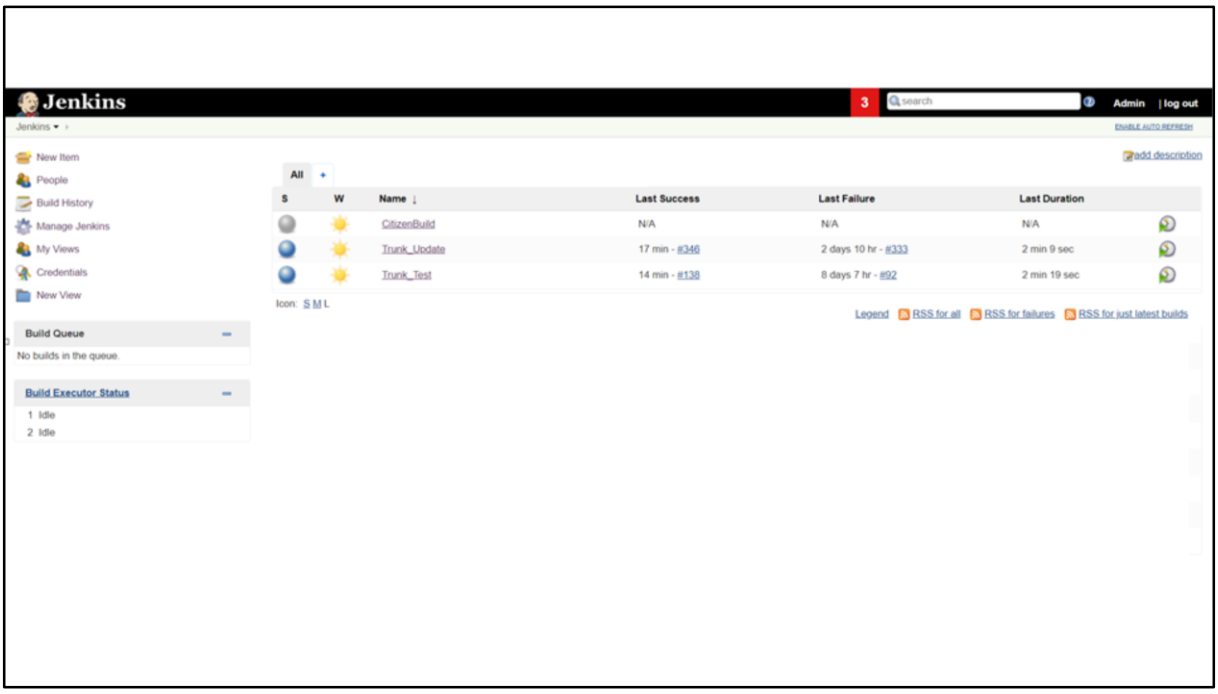

Jenkins › CitizenBuild ›
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Build Environment
Delete workspace before build starts
Use secret text(s) or file(s)
Abort the build if it's stuck
Add timestamps to the Console Output
Show tests in progress
With Ant
Build
Add build step
Conditional step (single)
Build GeneXus KB
Build a Visual Studio project or solution using MSBuild
Conditional steps (multiple)
Execute Windows batch command
Execute shell
Invoke Ant
Invoke top-level Maven targets
Run with timeout
Set build status to "pending" on GitHub commit
Trigger/call builds on other projects
Page generated: Sep 28, 2018 12:22:41 AM UYT
REST API
Jenkins ver. 2.141


Jenkins › Trunk_Test ›
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Build
Build a Visual Studio project or solution using MSBuild
MSBuild Version
MSBuild
MSBuild Build File
C:\GXserverClient\RunUnitTests.msbuild
Command Line Arguments
/t:RunAllTests
/p:KBPath="C:\Models\Citizen_GX16UpdCourse"
/p:GX_PROGRAM_DIR="C:\Program Files (x86)\GeneXus\GeneXus15U10"
/p:EnvironmentName="UnitTest"
/p:JUnitTestFilePath="C:\Program Files (x86)\Jenkins\workspace\Trunk_Test\TestResults"
Pass build variables as properties
Do not use chcp command
Advanced...
Add build step
Post-build Actions
Publish JUnit test result report
Test report XMLs
/TestResults/*.xml
Fileset 'includes' setting that specifies the generated raw XML report files, such as
Save
Apply

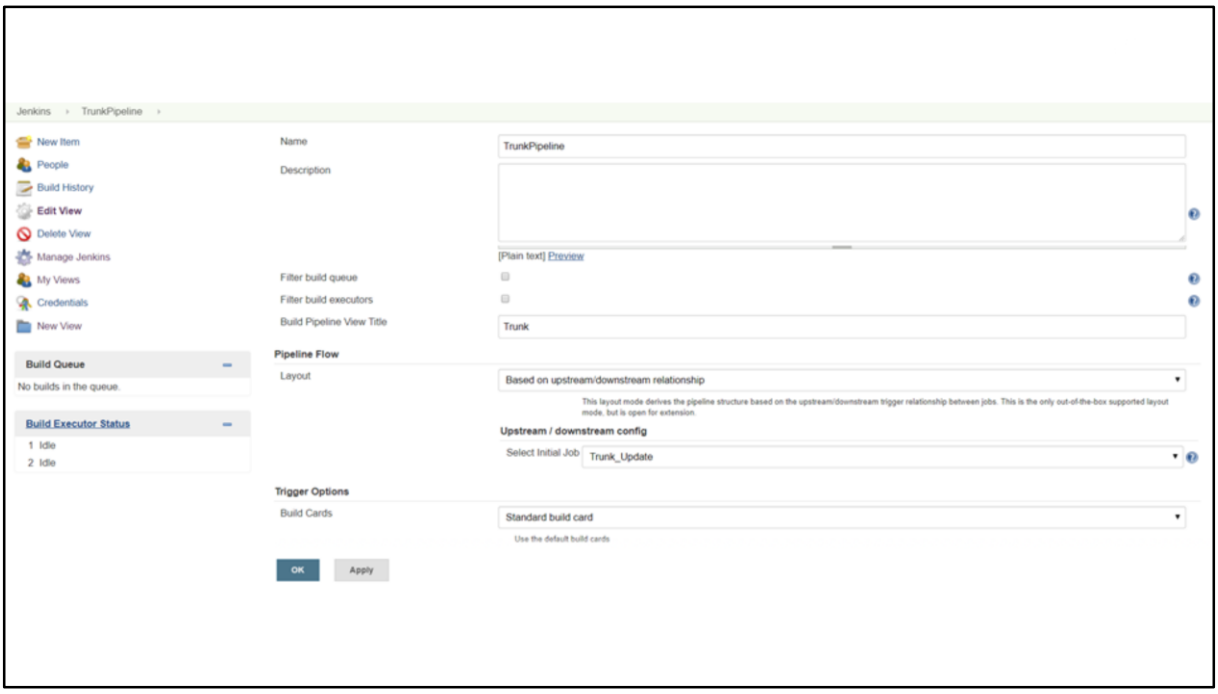

Jenkins › CitizenBuild ›
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Show tests in progress
With Ant
Build
Add build step
Post-build Actions
Build other projects
Projects to build
Trunk_Deploy
Trigger only if build is stable
Trigger even if the build is unstable
Trigger even if the build fails
Add post-build action
Save
Apply
Page generated: Sep 28, 2018 1:18:28 AM UYT
REST API
Jenkins ver. 2.141

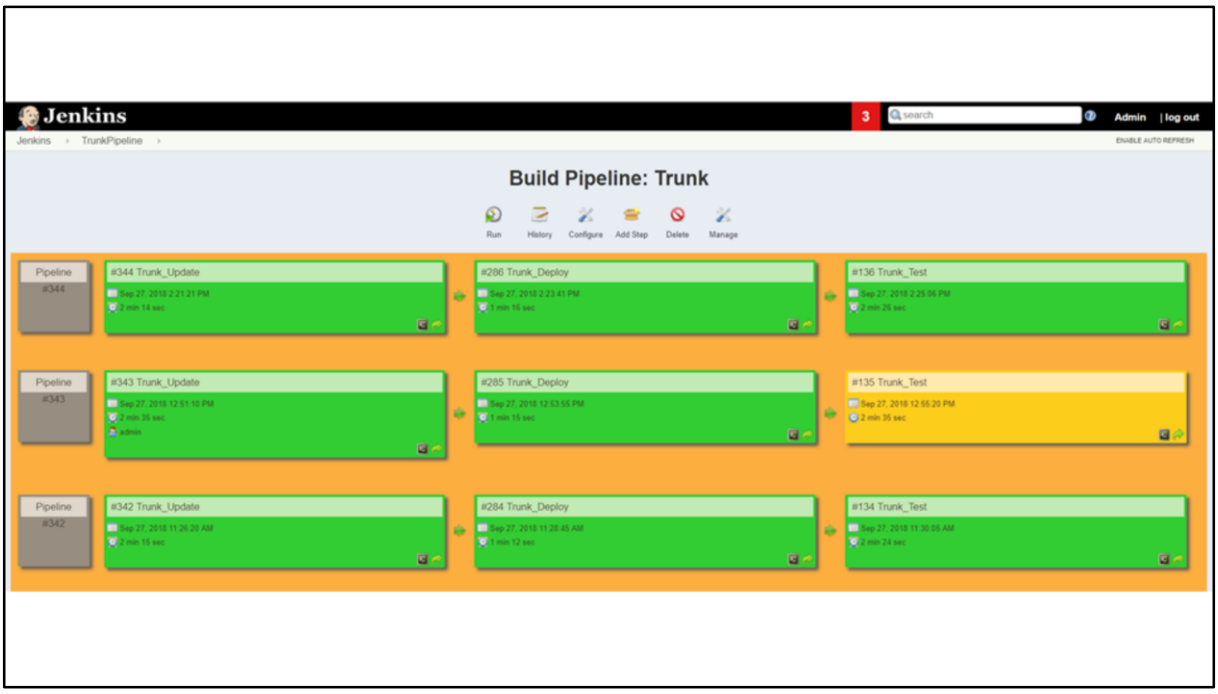

Trunk Build
Trunk Deploy
Trunk Test

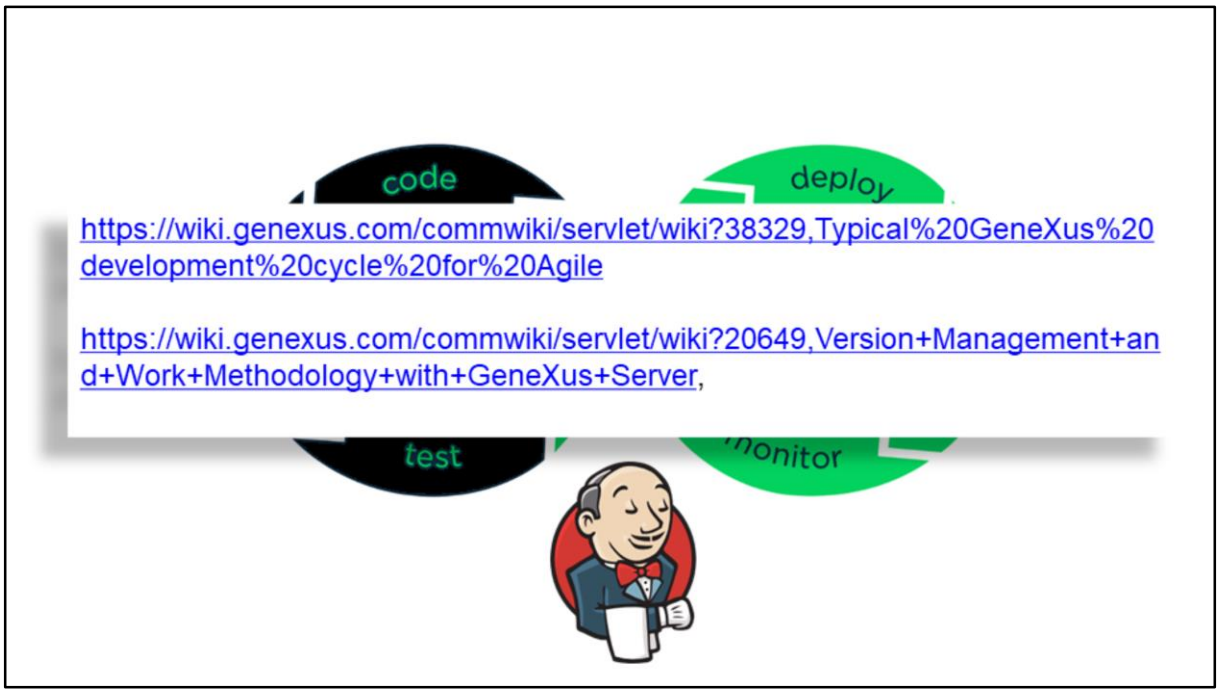

CI PIPELINE
Trunk Build
Trunk Deploy
Trunk Test


Jenkins
3
search
Admin
log out
Jenkins
ENABLE AUTO REFRESH
New Item
People
Build History
Manage Jenkins
My Views
Credentials
New View
add description
All
S
W
Name
Last Success
Last Failure
Last Duration
CitizenBuild
N/A
N/A
N/A
Trunk_Update
17 min - #346
2 days 10 hr - #333
2 min 9 sec
Trunk_Test
14 min - #138
8 days 7 hr - #92
2 min 19 sec
Icon: S M L
Legend
RSS for all
RSS for failures
RSS for just latest builds
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle


Jenkins
3
search
Admin
| log out
Jenkins ›
New Item
People
Build History
Manage Jenkins
My Views
Credentials
New View
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
View name
Trunk Pipeline
Build Pipeline View
Shows the jobs in a build pipeline view. The complete pipeline of jobs that a version propagates through are shown as a row in the view.
List View
Shows items in a simple list format. You can choose which jobs are to be displayed in which view.
My View
This view automatically displays all the jobs that the current user has an access to.
OK
Page generated: Sep 27, 2018 11:14:22 PM UYT
REST API
Jenkins ver. 2.141

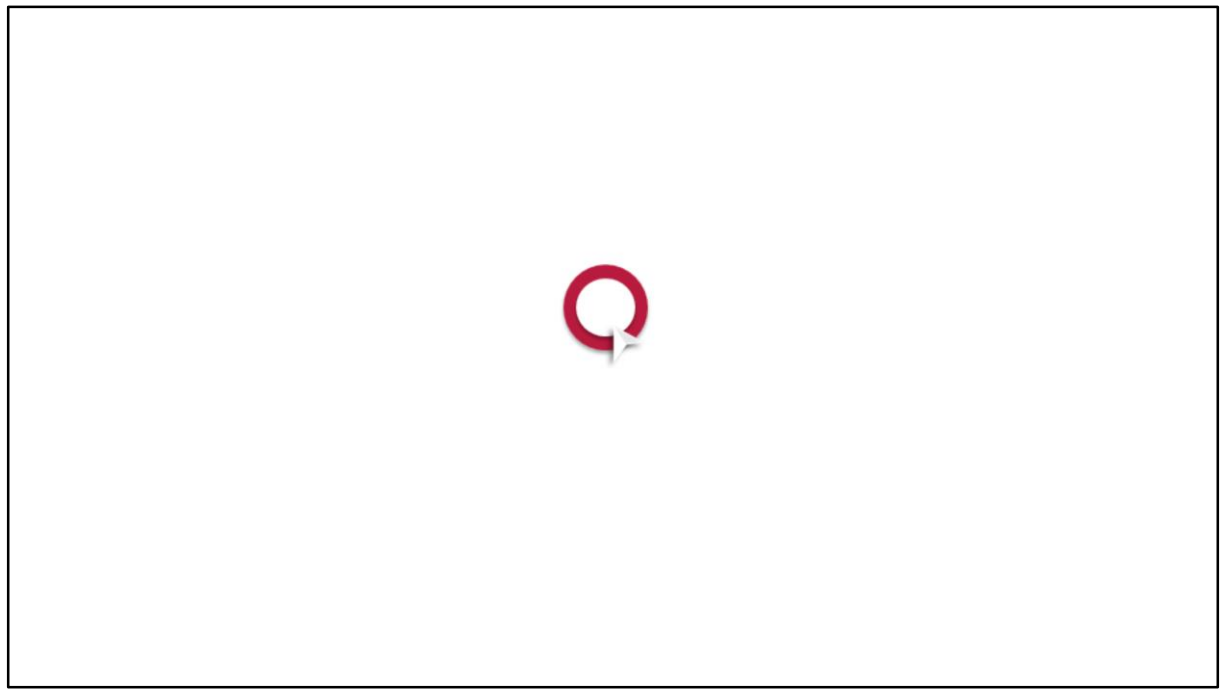

Jenkins › TrunkPipeline ›
New Item
People
Build History
Edit View
Delete View
Manage Jenkins
My Views
Credentials
New View
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
Name
TrunkPipeline
Description
[Plain text] Preview
Filter build queue
Filter build executors
Build Pipeline View Title
Trunk
Pipeline Flow
Layout
Based on upstream/downstream relationship
This layout mode derives the pipeline structure based on the upstream/downstream trigger relationship between jobs. This is the only out-of-the-box supported layout mode, but is open for extension.
Upstream / downstream config
Select Initial Job
Trunk_Update
Trigger Options
Build Cards
Standard build card
Use the default build cards
OK
Apply

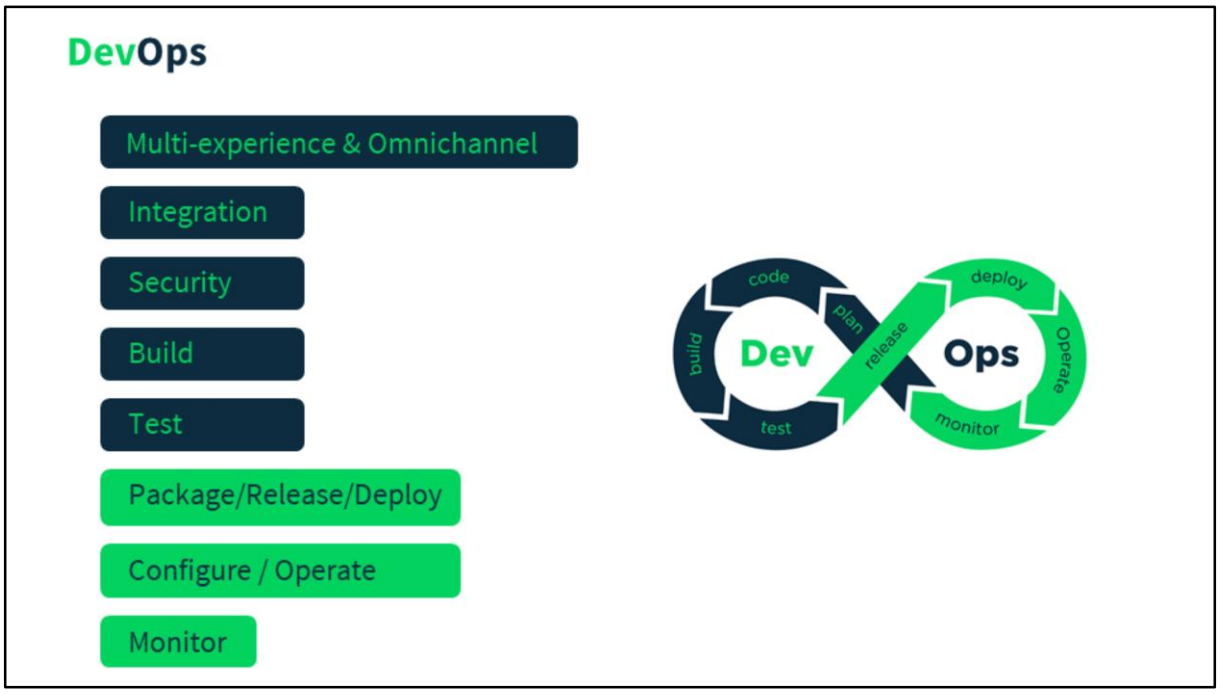

Jenkins
3
search
Admin
| log out
Jenkins › TrunkPipeline ›
ENABLE AUTO REFRESH
Build Pipeline: Trunk
Run
History
Configure
Add Step
Delete
Manage
Pipeline
#344
#344 Trunk_Update
Sep 27, 2018 2:21:21 PM
2 min 14 sec
#286 Trunk_Deploy
Sep 27, 2018 2:23:41 PM
1 min 16 sec
#136 Trunk_Test
Sep 27, 2018 2:25:06 PM
2 min 26 sec
Pipeline
#343
#343 Trunk_Update
Sep 27, 2018 12:51:10 PM
2 min 35 sec
admin
#285 Trunk_Deploy
Sep 27, 2018 12:53:55 PM
1 min 15 sec
#135 Trunk_Test
Sep 27, 2018 12:55:20 PM
2 min 35 sec
Pipeline
#342
#342 Trunk_Update
Sep 27, 2018 11:26:20 AM
2 min 15 sec
#284 Trunk_Deploy
Sep 27, 2018 11:28:45 AM
1 min 12 sec
#134 Trunk_Test
Sep 27, 2018 11:30:05 AM
2 min 24 sec



Hablamos del Server que es una herramienta fundamental en simplificar la integracion del codigo, pero para realmente poder alcanzar un flujo de DevOps necesitamos
Comenzar implementando una integracion continua. Una novedad en esta version es que ademas de Cruise-Control ahora GeneXus se integra Tambien con Jenkins como motor de Integracion continua.

De esta manera se puede automatizar el caso ‘basico’ que seria que cuando hay un cambio en el codigo, se dispare automaticamente un build





Con esto hemos visto los cambios más importantes entre la versión GeneXus 15 liberada y la versión GeneXus 16.


GeneXus™
The power of doing