CI/CD Pipelines for IBM App Connect Enterprise

A Practical Strategy for "Two Speed" DevOps Integrations

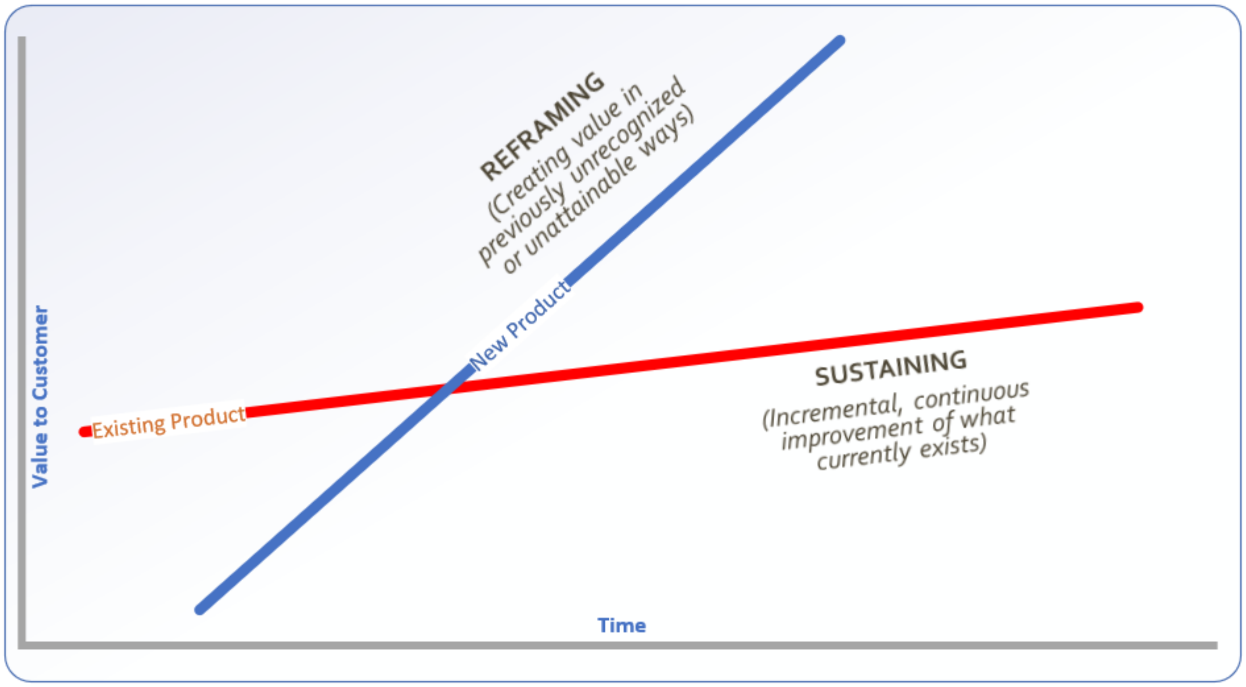
John Carr

Integration Services Practice Manager

October, 2024

Table of Contents

1. Overview



This document represents an abridgement of discussions, customer engagements and independent research we at TxMQ, Inc. conducted over the past few years helping our clients modernize their IBM Integration Bus (IIB) hub-and-spoke environments into more container-based NextGen integrations using IBM App Connect Enterprise (ACE).

Many organizations have existing, traditional ESB and SDLC practices in place serving its customers. Over time, sustaining this product through incremental, continuous improvements provides limited value.

As an example: the ESB satisfies traditional on-prem hub-and-spoke integrations, but needs for more container-based deployments are not possible; this hampers new business initiatives.

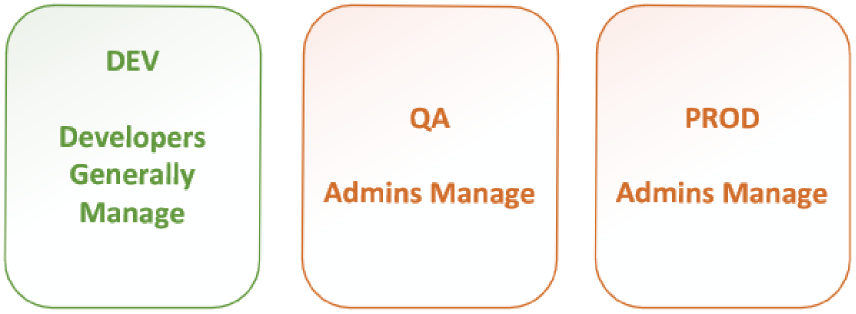
One of larger challenges integration teams face is the need to maintain their core ESB along with other established systems as part of scheduled delivery cycles (Classic in nature). At the same time, there is a growing need for next-gen integrations which has faster, more fluid delivery cycles (DevOps in nature).

That said, reframing the product (ESB and SDLC Practices) provides a way to create value in previously unrecognized ways to satisfy both needs.

2. Case Study: An On-Prem ESB and SDLC Practices

This section provides a *summary* and *key points* of SDLC activities supporting current ESB environments we've encountered from various client engagements. Its purpose is more as a case study and reference when, in **Section 4** (Pipeline Explorer), **Section 5** (Pipeline Strategy for "Two Speed" Operations), **Section 6** (Pipeline for ESB Integrations) and **Section 7** (Pipeline for Next-Gen Integrations), changes are introduced.

2.1 Responsibilities by Environment

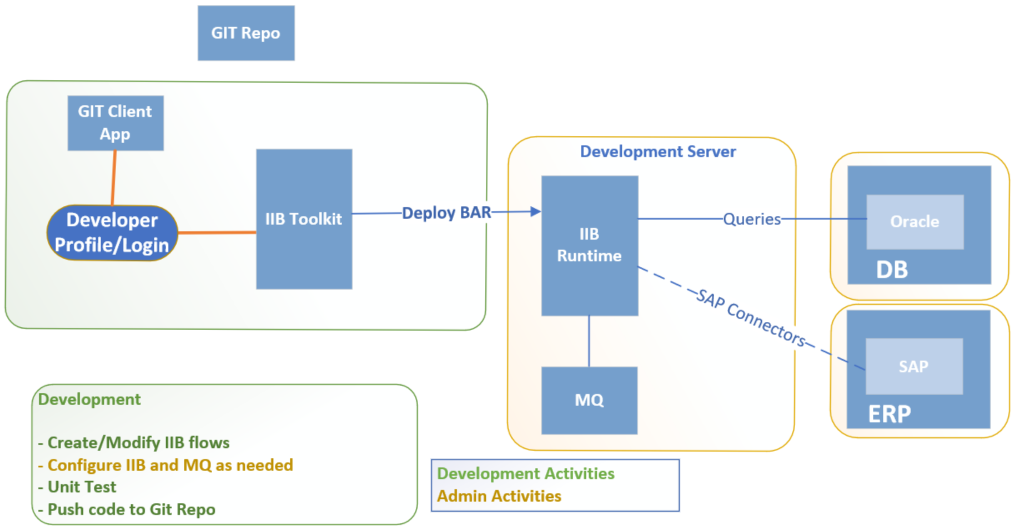


In this case study, a traditional three environment setup (DEV, QA, PROD) is operational on-prem using IBM MQ LTS 9.x and IBM Integration Bus 10.x on Red Hat Enterprise Linux. High Availability for MQ/IIB is configured through multi-instance mechanisms.

Developers have dominion over their environment to create MQ queues, new IIB Servers, deploy flows, etc. The exception is the admin team manages this development environment to apply Fixpacks or MQ/IIB config changes either through a schedule of maintenance activities.

Admins have dominion over the QA and PROD environments with incoming change requests through ServiceNOW. Developers are not allowed to make any configuration changes to these environments.

2.2 Development Practices

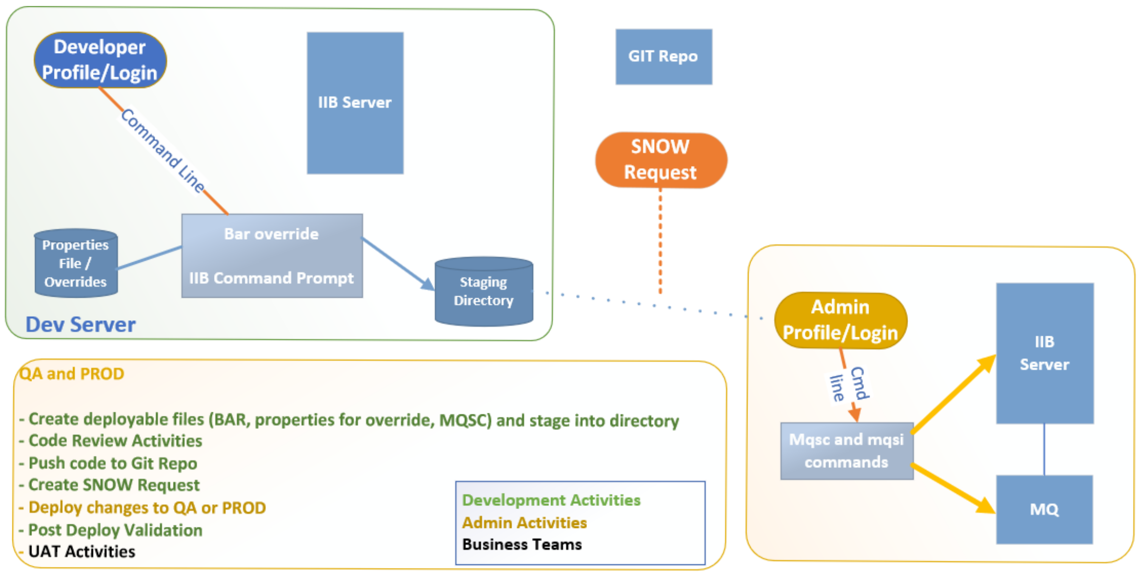


Development practices in this case study follow a traditional, structured approach for integration development with IIB. This approach is not unlike what other IIB developers perform at other shops. In general, a Developer will logon to a Windows workstation to perform:

- Use a GIT Client App to check out source from a GIT Repository onto their drive
- With IIB Toolkit, import source into a workspace to create/modify flows

- Build/deploy BAR file to an IIB Runtime
- Perform simple unit testing within the Toolkit
- Debug flows with the IIB Toolkit Debugging Perspective as needed
- Push code back to GIT Repository when ready for QA Staging

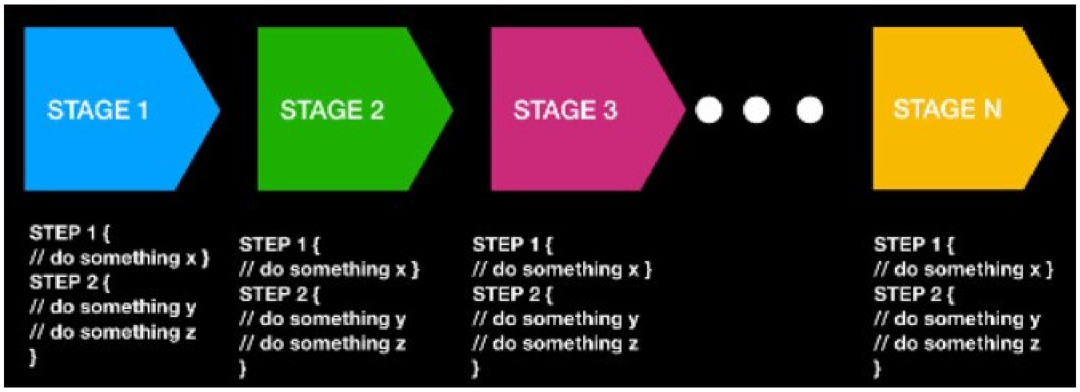
2.3 Promote to QA and PROD Practices



Development and admin practices follow a traditional, structured approach for deploying to QA/PROD. This approach has similarities to other shops with IIB. In general, when a developer is ready to promote to QA:

- Login to the DEV Server to create an environment specific BAR file with overrides.
- Place associated BAR files and MQSC scripts into the appropriate staging directory
- Send email to supervisor/leads requesting a code review. This code review includes inspection of unit test reports and other documents on the GIT repository, in addition to spot checks of the integration code to verify compliance to established best practices
- Checklist is completed by supervisor/leads and developer and pushed to GIT repository
- Developer creates a ServiceNOW request for Admins to deploy changes
- Admins review ServiceNOW request containing deployment steps, create commands to satisfy each step, SSH into QA/PROD and manually execute from the command prompt. When complete, the Admin notifies the Developer and closes the ServiceNow request.
- Developer performs post-deploy validation
- Business or other teams perform acceptance testing

3. The Purpose of CI/CD Pipelines



3.1 CI/CD Pipelines in a Nutshell

A CI/CD pipelines (or “Pipeline” for short) represents a series of steps or jobs to be performed in order to deliver a new version of software. Pipelines are a practice focused on improving software delivery using either a DevOps or site reliability engineering (SRE) approach. Although it is possible to manually execute each of the steps of a CI/CD pipeline, the true value of CI/CD pipelines is realized through automation.

3.2 Elements of a Pipeline

The steps that form a CI/CD pipeline are distinct subsets of tasks grouped into what is known as a pipeline stage. Automation here can save both time and effort. Typical pipeline stages include:

- **Build**: When the application is compiled.
- **Test**: When code is tested.
- **Release**: When the application is delivered to the repository.
- **Deploy**: When the code is deployed to production.

This represents more of the milestones related to pipeline stages and not considered a comprehensive list. That said, pipelines are unique to the requirements of your organization and products it serves.

3.3 The Role of Jenkins

It is important to realize Jenkins itself is not a pipeline. Just creating a new Jenkins job does not construct a pipeline. Jenkins is an engine used to house a choreography of steps to perform a series of actions/interactions with other tools. It offers a way for other application APIs, software libraries, build tools, etc. to plug into Jenkins, and it executes and automates the tasks. On its own,

Jenkins does not perform any functionality but gets more and more powerful as other tools are plugged into it.
Jenkins itself has a way to execute a job sequentially in a defined way by codifying it and structuring it inside multiple blocks that can include multiple steps containing tasks. This is called a Jenkinsfile.

3.4 The Role of Artifactory

Artifactory is a product by JFrog that serves as a binary repository manager. This repository is a natural extension to the source code repository, in that it will store the outcome of your build process, often denoted as artifacts. Most of the times one would not use the binary repository directly but through a package manager that comes with the chosen technology.

In most cases these will store individual application components that can later be assembled into a full product - thus allowing a build to be broken in smaller chunks, making more efficient use of resources, reducing build times, etc.

Binary repositories are as vital a part of a well-designed DevOps setup as the source code repository or continuous integration.

Data retention policies are needed and dependent on its users to determine what needs to be cleaned up. Artifactory will not delete binaries automatically. In general, there are three kinds of techniques that are used to manage artifact storage in Artifactory:

- Clearing oversized caches
- Deleting unused artifacts
- Limiting the number of build snapshots that are retained

For clearing oversized caches: Artifactory's remote repositories store downloaded files in a cache. It’s generally beneficial to retain the entire cache as it speeds up downloads. However, if the artifacts being used for a given project change, you might find it worthwhile to periodically clear out the cache.

For deleting unused artifacts: Artifactory will not delete binaries automatically unless you make use of defining the rules. Of these, one of the most popular is the artifactCleanup plugin, which runs on a cron job, automatically deleting any artifact that has not been downloaded for "x" number of days.

For limiting the number of build snapshots retained: When this setting is enabled, during any given build run resulting in uploads reaching the Max Unique Snapshots number you entered, older releases will automatically be deleted. The highest number will always be your latest release.

3.5 The Role of a Test Harness

A Test Harness are a collection of stubs, drivers and other supporting tools required to automate test execution. The harness executes tests by using a test library and generates test reports. Test harness contains all the information needed to compile and run a test like test cases, target deployment port (TDP), source file under test, stubs, etc.

3.6 The Role of Code Quality

The static code analysis can be used to expose the areas of code that can be improved in terms of quality, and even higher, we can integrate this static analysis into the development workflow, and thus, tackle the code quality issues in the early stages of the development even before they reach the production.

SonarQube may be in use within your organization for code quality of HTML, Java, Python and TypeScript. The product itself provides code quality with a set of free or 3rd party plugins for the rules. BetterCodingTools (BCT) is an example of a 3rd party provider specific to inspecting IIB/ACE flows and ESQL modules. Therefore, investing in a SonarQube plugin for IIB/ACE rules/inspection is recommended.

Reference: http://bettercodingtools.com/mb-rules/

4. Pipeline Explorer

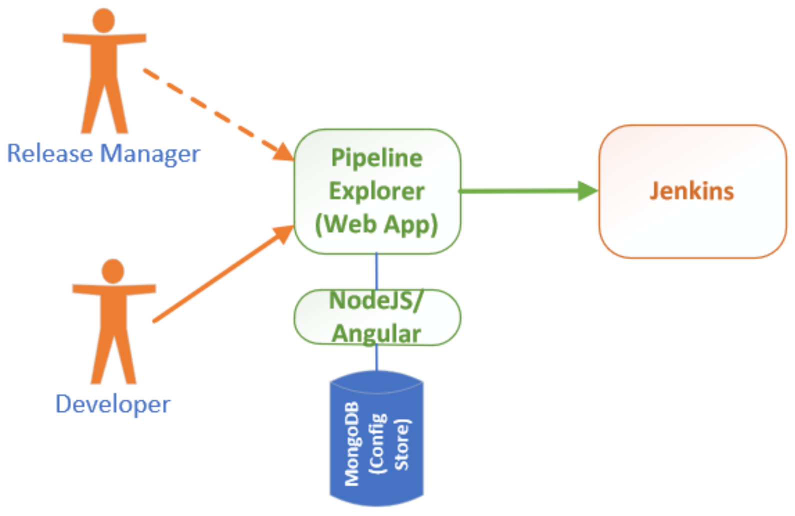
This section provides justifications for a Pipeline Explorer as a web-based frontend to Jenkins. Further details for the composition and purpose of Pipeline Explorer are provided, too.

4.1 The Challenges with Jenkins as the Main UI

Jenkins is a powerful and flexible tool to choregraph pipelines, but one of the larger challenges for a newcomer is to learn how to operate it. For example, organizing Jenkins Jobs and filling out the web forms can often lead to confusion as to what job to submit.

Though customizations are possible within Jenkins, a better approach would be to abstract the complexities of Jenkins and other DevOps tools to allow IIB/ACE developers to focus on their messages flow life cycle. This is where ***Pipeline Explorer*** comes in.

4.2 Pipeline Explorer



Pipeline Explorer is should be considered a ***strategic asset and DevOps accelerator*** for an organization's Integration Team. Its purpose is to be a web-based app for orchestrating builds, tests, and deployment jobs supporting integration needs. This means integration developers could focus on their development flows rather than the complexity of Jenkins and other DevOps tools.

Pipeline Explorer could be written in a number of languages and platforms, but a recommendation is to write it as a MEAN (MongoDB, Express JS, Angular JS and Node JS) application. Through this web application, integration developers would be able to build, deploy and test their IIB/ACE work and also provide role-based promotion options.

Through Pipeline Explorer, as an accelerator for DevOps, the benefits would be:

- An automated solution for building, testing, promoting and deploying IIB/ACE Integrations

- A config database containing details about what flows are deployed in the topology, with a reference back to the ServiceNOW requests, Git Repository, Testing results, and deployed artifacts

- A dashboard providing visibility into the IIB/ACE infrastructure and SDLC metrics

- Ability to identify who authorized each release and when

- Automated Library and Dependency Management to identify fail-fast scenarios

Learn more about the MEAN stack here: https://www.ibm.com/cloud/learn/mean-stack-explained

5. Pipeline Strategy for "Two-Speed" Operations

One of larger challenges integration teams face is the need to maintain their core ESB interactions with SAP or other established systems with scheduled delivery cycles (Classic in nature), and at the same time support a growing need for next-gen integrations which has faster, more fluid delivery cycles (DevOps in nature).

An organization's ESB may satisfy on-prem hub-and-spoke integrations, but needs for more container-based deployments are not possible; this hampers new business initiatives. Leveraging containers can drastically increase the agility of NextGen Pipelines. Containers allow integrations to be deployed reliably and migrate quickly between various environments by packaging code, configuration settings, and dependencies into a single object (application image).

A recommendation is to construct and Pipeline supporting both needs in a "Two Speed" strategy.

Two-speed is the concept that strategic planning for an IT department should include a fast track that allows some projects to be implemented quickly. The strategy proposes that agile, innovative initiatives should be allowed to move forward quickly without being hampered by the checks and balances that are needed to maintain business-critical IT operations.

Key points from the 2014 McKinsey article entitled, "A two-speed IT architecture for the digital enterprise" support this recommendation:

- Delivering an enriched customer experience requires a new digital architecture running alongside legacy systems

- A two-speed IT architecture will help companies develop their customer-facing capabilities at high speed while decoupling legacy systems for which release cycles of new functionality stay at a slower pace

- Unlike enterprises that are born digital, traditional companies don't have the luxury of starting with a clean slate; they must build an architecture designed for the digital enterprise on a legacy foundation

Reference: https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/a-two-speed-it-architecture-for-the-digital-enterprise

Following a "Two-Speed" approach to the Pipeline, **Sections 6** (Pipeline for ESB Integrations) and **Section 7** (Pipeline for Next-Gen Integrations) provide further descriptions of the strategy to maintain stability to transactional core systems such as the ESB, while providing adaptability for next-gen initiatives.

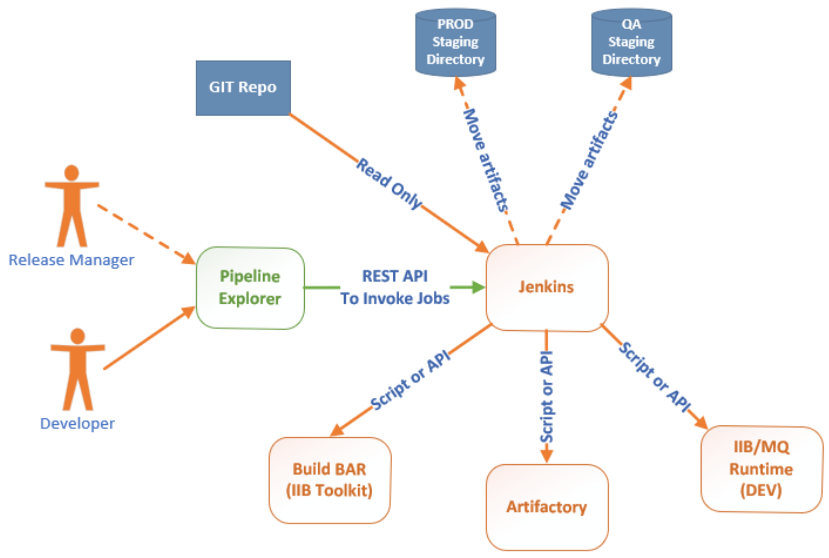
6. Pipeline for ESB Integrations

This section is closely related to **Section 2** (Current Environment and SDLC Practices) as the Pipeline introduces new functionality while maintaining the ESB operations in place. As each new version of the Pipeline is described, additional functionality and automation is included as an accelerator for the current ESB.

Though this Pipeline is for the current ESB supporting IIB v10, its approach is the same when upgrading to ACE v12 or v13 to continue the hub-and-spoke ESB with minimal changes to the Pipeline; this would be to maintain stability to transactional core systems such as SAP and not introduce containers.

For clarity, the diagrams represent Pipeline tasks performed at a high level. Refer to **Section 3.2** (Elements of a Pipeline) to learn more about tasks.

6.1 ESB - Version 1

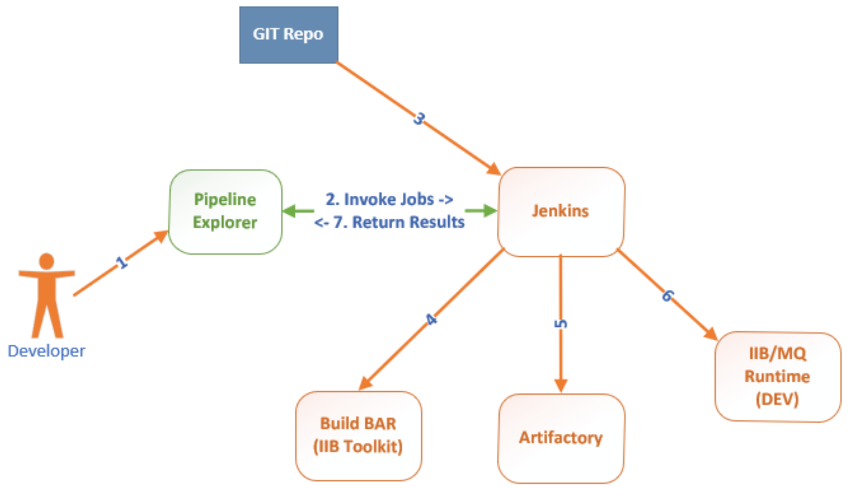


Overall, Pipeline v1 for ESB Integrations is an introductory iteration for the organization. Activities performed in **Section 2.2** (Development Practices) continue to take place outside of the Pipeline.

As an introduction version, Pipeline will be used for basic “seeding” of the Pipeline Explorer Configuration Database as its role is to be the one source to know what integrations/flows are deployed within the ESB Topology.

This is important so the new Pipeline begins to play a role in promoting to QA and PROD environments. Each time the Pipeline is used, it contains more information about the ESB in use. Learn more about Pipeline Explorer in **Section 4**.

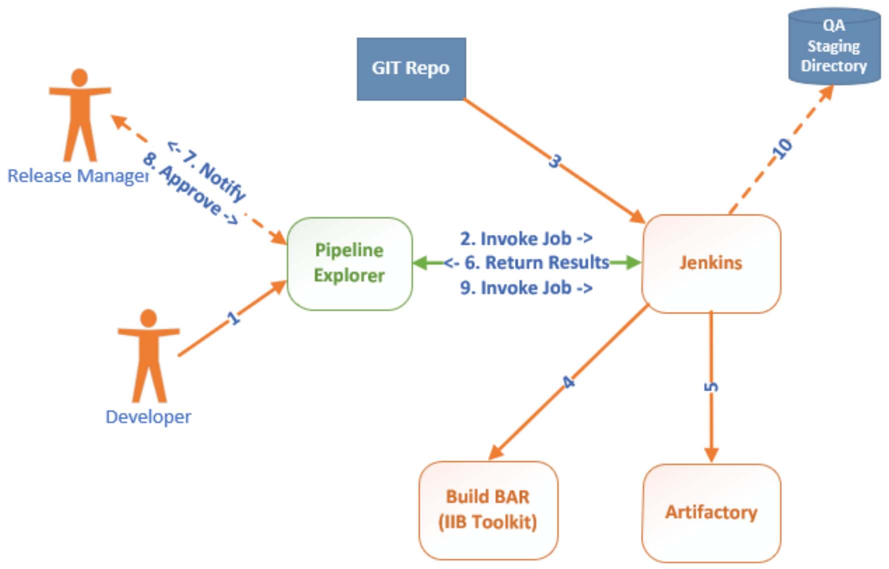
Activities for DEV



Step Number	Description	Notes
1	Create a “Change Package” indicating source location (GIT Repo), Config File location on GIT to BAR File Creation/Overrides for DEV, and Target (IIB Exec Group	The GIT Repo will include additional config file details and other artifacts to support the Pipeline. The intentions are these config files contain details (in JSON or YAML) on how to build and override BAR files.
2	Pipeline Explorer invokes Jenkins via REST API with information provided in the “Change Package” and other details contained within its internal Config Database	The “Change Package” has an ID for tracking
3	Jenkins, though custom scripting, checks out the necessary source files from GIT, performs some dependency checks to determine build parameters are satisfied	Can be a combination of Jenkins plugins and custom scripts within the jenkinsfiles
4	Jenkins performs activities to build the BAR with appropriate overrides	Build BAR activities can be performed as an ANT or bash script.
5	Check BAR file(s) for DEV into Artifactory	

6	Deploy to Target IIB Runtime and Execution Group	This would be the existing DEV IIB Environment. If the flow(s) already exists on the Target location, these deploy actions replace the existing flow(s) with this newly built BAR file
7	Results are returned to Pipeline Explorer to update its config database and for the Developer to determine next steps	A Developer next step outside of the pipeline would be to manually run the Unit Test app to ensure the deploy was successful

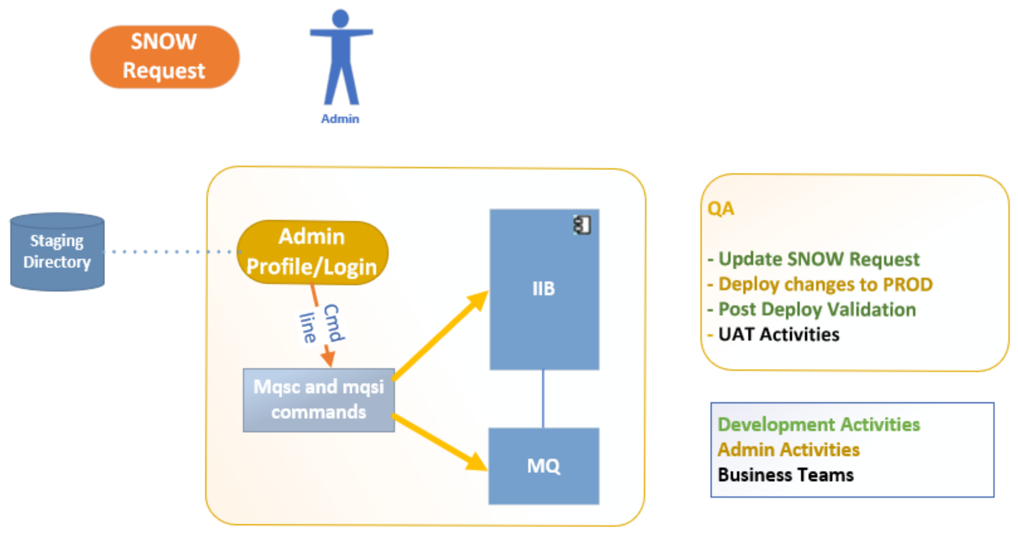
Activities for QA



Step Number	Description	Notes
1	Developer Updates the Change Package to include Config File location on GIT for BAR File Overrides for QA, and Target (IIB Exec Group	The GIT Repo will include additional config file details and other artifacts to support the Pipeline. The intentions are these config files contain details (in JSON or YAML) on how to build and override BAR files.
2	Pipeline Explorer invokes Jenkins via REST API with information provided in the "Change Package" and other details contained within its internal Config Database	The "Change Package" has an ID for tracking

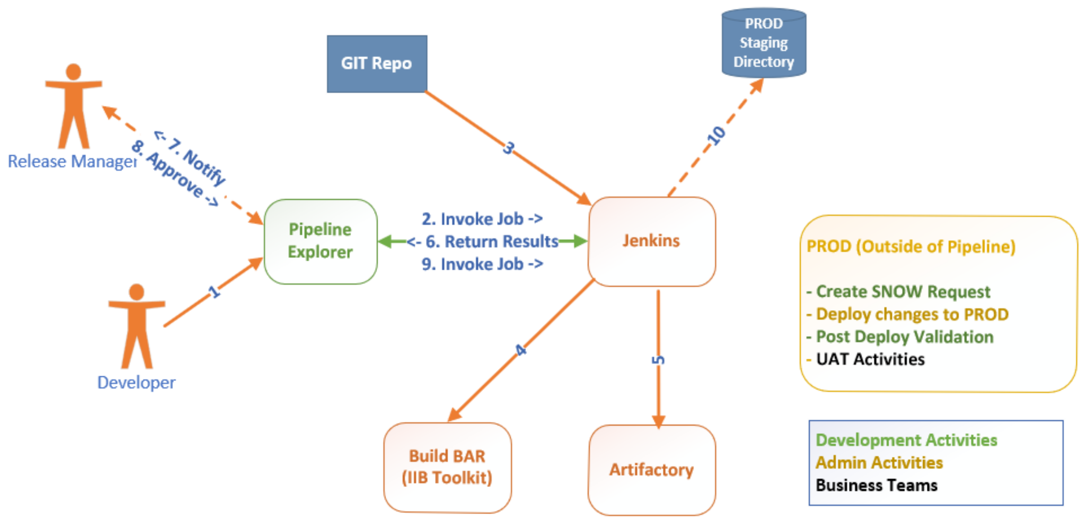
3	Jenkins, though custom scripting, checks out the necessary source files from GIT, performs some dependency checks to determine build parameters are satisfied. Also, it interacts with Artifactory to grab the BAR file already stored from the DEV process to use in Step 4	Can be a combination of Jenkins plugins and custom scripts within the jenkinsfiles
4	Jenkins performs activities to build the BAR with appropriate overrides for QA	Build BAR activities can be performed as an ANT or bash script.
5	Check BAR file(s) for QA into Artifactory	
6	Results are returned to Pipeline Explorer to update its config database.	If the build is successful, Step 7 is performed. If the build fails, the developer resolves issues to re-submit Change Package
7	If the build is successful from Step 6, then Package Explorer sends an email notification to the Release Manager	
8	The Release Manager, after performing the code review and other activities, logs into Package Explorer to approve/reject the Change Package	This action notifies the Developer their Change Package is approved or rejected
9	When the Change Package is approved, this action invokes another Jenkins job described in Step 10	
10	Jenkins interacts with Artifactory to get the QA BAR file and moves it to the QA Staging Directory	Pipeline Explorer Config Database is updated. Outside of the Pipeline, the Developer creates a ServiceNOW request for the IIB Admin to deploy to QA

Activities outside of Pipeline for QA



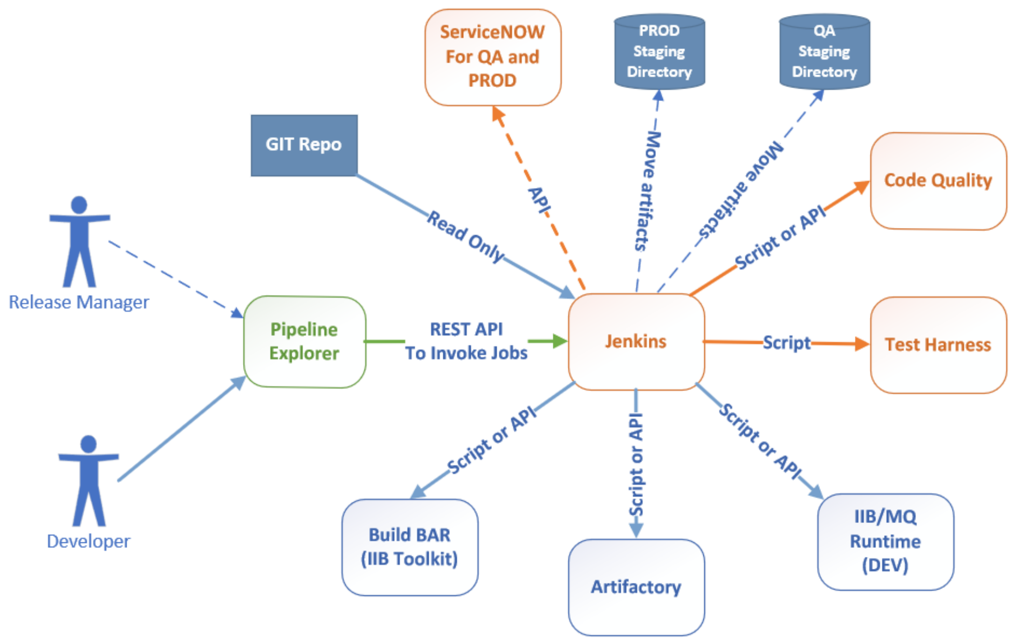
Outside of the Pipeline, the Developer creates a ServiceNOW request for the IIB Admin to deploy to QA. Post-deploy validation and UAT activities are unchanged.

Activities for PROD inside and outside of the Pipeline



Activities for PROD would be the same as the **Activities for QA** subsection, with exception to the BAR overrides and Staging Directory for PROD. Also, outside of the Pipeline, the Developer creates a ServiceNOW request for the IIB Admin to deploy to PROD. Post-deploy validation activities are unchanged.

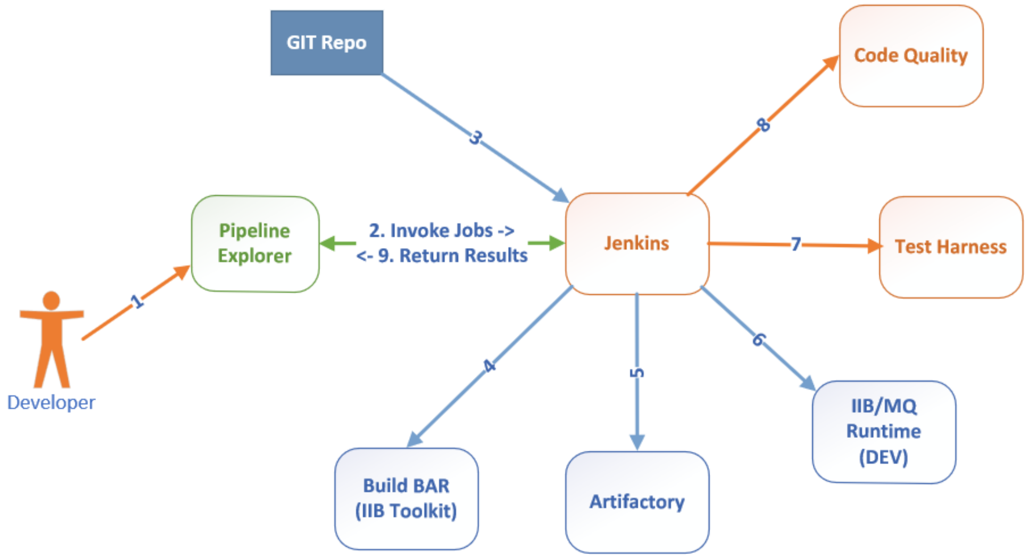
6.2 ESB - Version 2



Pipeline v2 for ESB Integrations builds upon its predecessor to include a Test Harness, Code Quality and basic integrations into ServiceNow. Some activities performed in **Section 2.2** (Development Practices) are incorporated into the Pipeline, such as a Test Harness for unit testing, while other activities such as the IIB Admins deploying to QA/PROD continue to take place outside of the Pipeline.

This iteration of Pipeline will be used more by the developers for their SDLC activities. The Pipeline Explorer Configuration Database continues to grow as its role is to be the one source to know what integrations/flows are deployed within the ESB Topology. This is important so the new Pipeline begins to play a role in promoting to QA and PROD environments. Each time the Pipeline is used, it contains more information about the ESB in use. Learn more about Pipeline Explorer in **Section 4**.

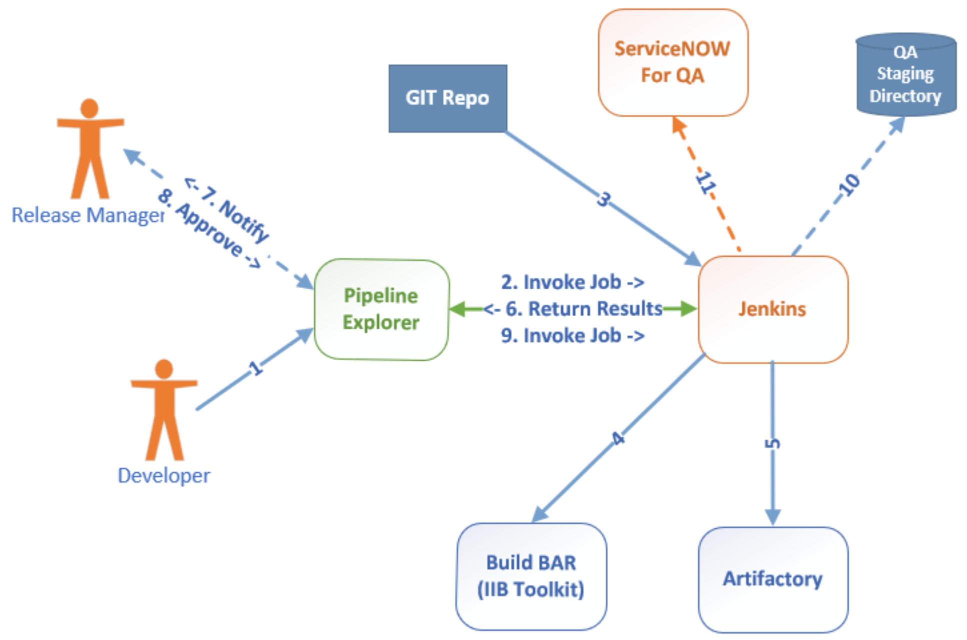
Activities for DEV



Step Number	Description	Notes
1	Create a "Change Package" indicating source location (GIT Repo), Config File location on GIT to BAR File Creation/Overrides for DEV, and Target (IIB Exec Group	The GIT Repo will include additional config file details and other artifacts to support the Pipeline. The intentions are these config files contain details (in JSON or YAML) on how to build and override BAR files.
2	Pipeline Explorer invokes Jenkins via REST API with information provided in the "Change Package" and other details contained within its internal Config Database	The "Change Package" has an ID for tracking
3	Jenkins, though custom scripting, checks out the necessary source files from GIT, performs some dependency checks to determine build parameters are satisfied	Can be a combination of Jenkins plugins and custom scripts within the jenkinsfiles
4	Jenkins performs activities to build the BAR with appropriate overrides	Build BAR activities can be performed as an ANT or bash script.
5	Check BAR file(s) for DEV into Artifactory	
6	Deploy to Target IIB Runtime and Execution Group	This would be the existing DEV IIB Environment. If the flow(s) already exists on the Target location, these deploy actions replace the existing flow(s) with this newly built BAR file

7	**Jenkins invokes the Test Harness as part of Unit Testing to verify the flow was successful. The output HTML report could be either put into GIT by Jenkins, or stored in the Pipeline Explorer Config Database**	**Unit testing with the custom Java App already in use can be used for this purpose. The Unit Test app acts as a test harness for message injection to the flow(s), and review the HTML output report of the results**
8	**Jenkins invokes a routine to do an automated code review of the flows and ESQL.**	**Examples of products to assist with the automated IIB/ACE code review are SonarCube with BCT Plugin (3rd party product).** **An alternative could be simple inspection of flow names to ensure they adhere to standards. This could be written even with bash scripts as part of a POC.** **This would be considered a basic code review and not a replacement for the code review process already in place performed by peers.**
9	Results are returned to Pipeline Explorer to update its config database and for the Developer to determine next steps	

Activities for QA



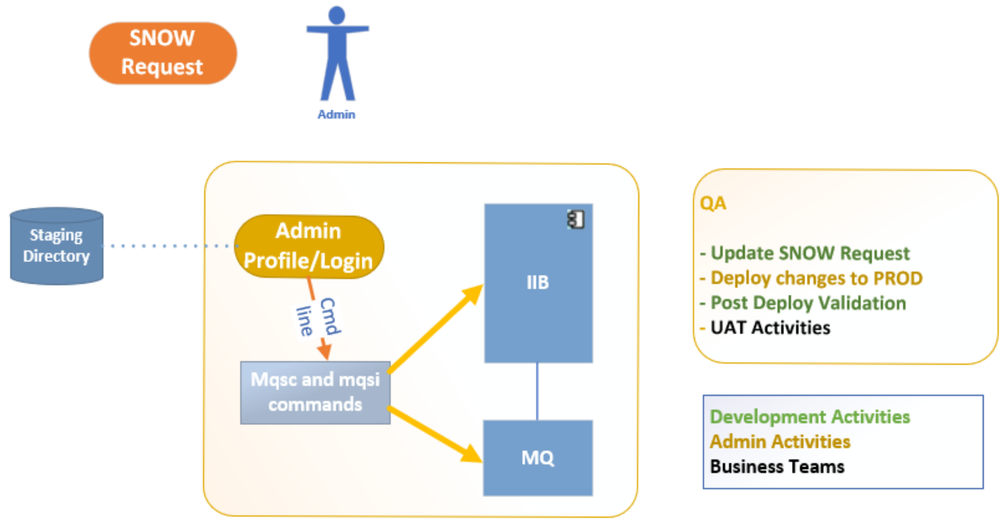
Pipeline v2 activities for QA are essentially the same as in V1. Refer the steps from **Section 6.1** to describe these steps in more detail. The only modifications made at this point would be:

- Additional checks made to ensure the Unit Test and the Code Quality as part of DEV deploy was complete. This could be stored in the config database as a Boolean (true/false the unit test was performed, etc.)

- Step 11 (ServiceNOW Request for QA Deploy) – This could be part of an iterative process to include ServiceNOW into the Pipeline to create a stub of a request that the Developer or Release Manager can use to make a formal request for Admin services.

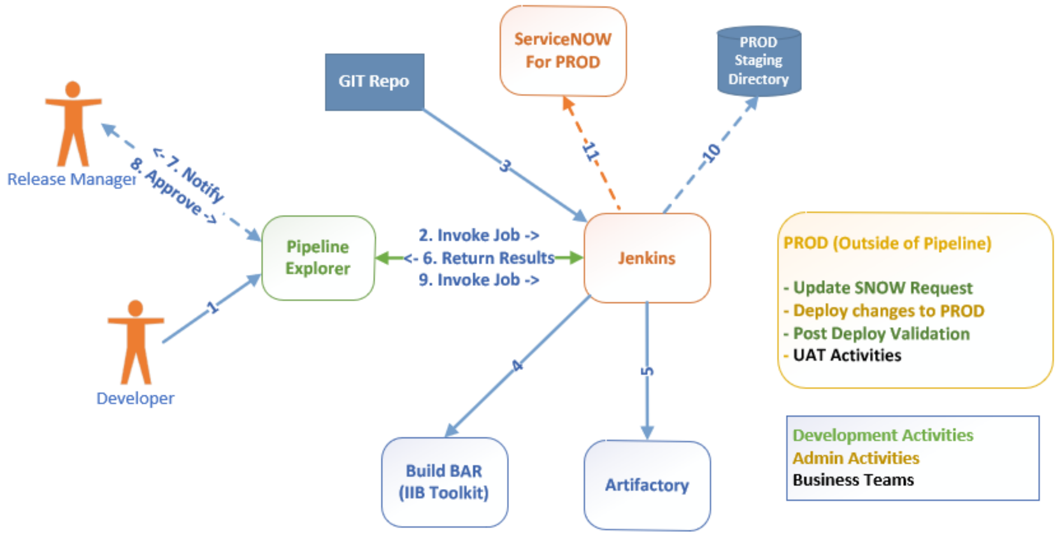
Outside of the Pipeline, the Release Manager performs a code review and manually checks the unit test reports for quality before Approval/Rejection.

Activities outside of Pipeline for QA



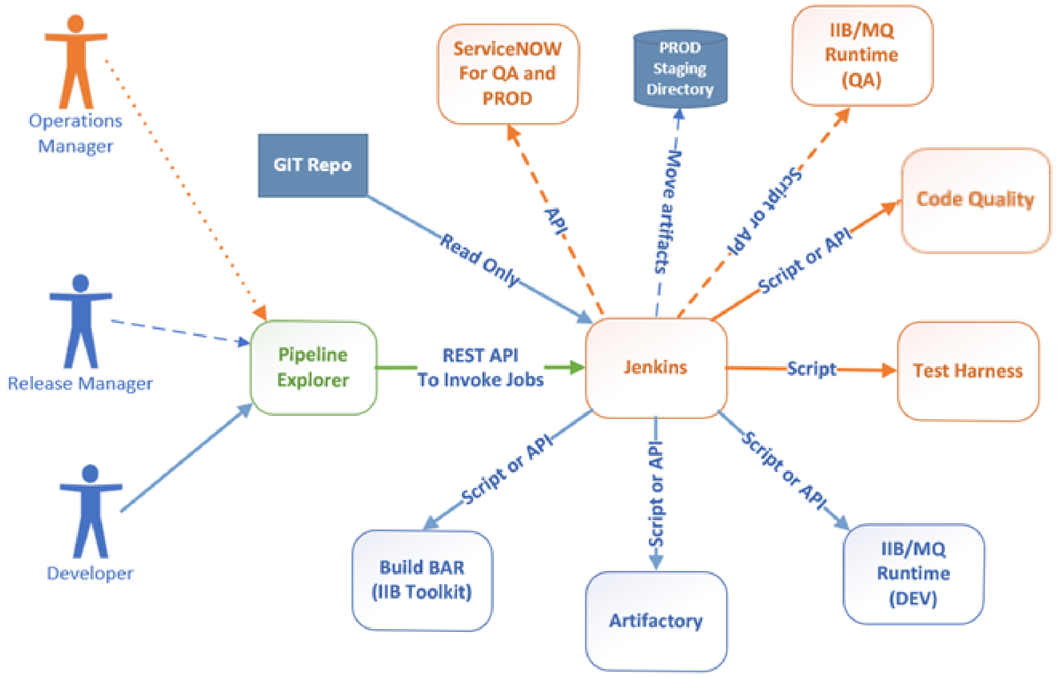
Outside of the Pipeline, the Developer updates the ServiceNOW generated request for the IIB Admin to deploy to QA. Post-deploy validation and UAT activities are unchanged.

Activities for PROD inside and outside of the Pipeline



Activities for PROD would be the same as the **Activities for QA** subsection, with exception to the BAR overrides and Staging Directory for PROD. Step 11 (ServiceNOW Request for PROD Deploy) creates a stub with basic information about the request. Outside of the Pipeline, the Developer updates this ServiceNOW request for the IIB Admin to deploy to PROD. Post-deploy validation activities are unchanged.

6.3 ESB - Version 3



Pipeline v3 for ESB Integrations builds upon its predecessor with enhancements to:

- Include more information as part of generating a ServiceNOW Request for QA/PROD
- Include Deploy actions to QA by Operations Manager (IIB Admin)
- The Test Harness to include Unit Test Validations for QA
- Code Quality (TBD depending on tools used, such as SonarCube)

More activities performed in **Section 2.2** (Development Practices) are incorporated into the Pipeline, such as automation for deploying/validation to QA, while other manual activities such as the IIB Admins deploying to PROD continue to take place outside of the Pipeline.

The Pipeline Explorer Configuration Database continues to grow as its role is to be the one source to know what integrations/flows are deployed within the ESB Topology. Learn more about Pipeline Explorer in **Section 4**.

Activities outside of Pipeline for QA

Outside of the Pipeline, the Developer updates the ServiceNOW generated request for the IIB Admin to deploy to QA. Through iterative Pipeline versions, more of the manual tasks performed are automated.

Activities for PROD outside of the Pipeline

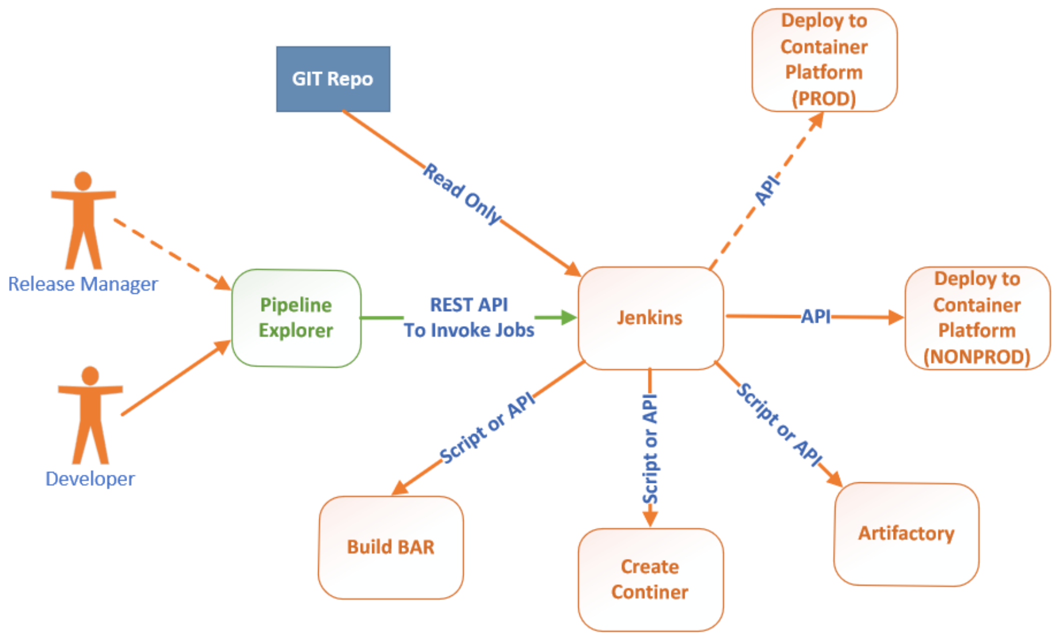
Outside of the Pipeline, the Developer updates this ServiceNOW request for the IIB Admin to deploy to PROD. Post-deploy validation activities are unchanged. Through iterative Pipeline versions, more of the manual tasks performed in PROD, such as Post Deploy Validation, could be automated.

7. Pipeline for NextGen Integrations

This section represents the Pipeline for NextGen Integrations that are container-based. As each new version of the Pipeline is described, additional functionality and automation is included as an accelerator for DevOps enablement using ACE v12 or v13.

For clarity, the diagrams represent Pipeline tasks performed at a high level. Refer to **Section 3.2** (Elements of a Pipeline) to learn more about tasks.

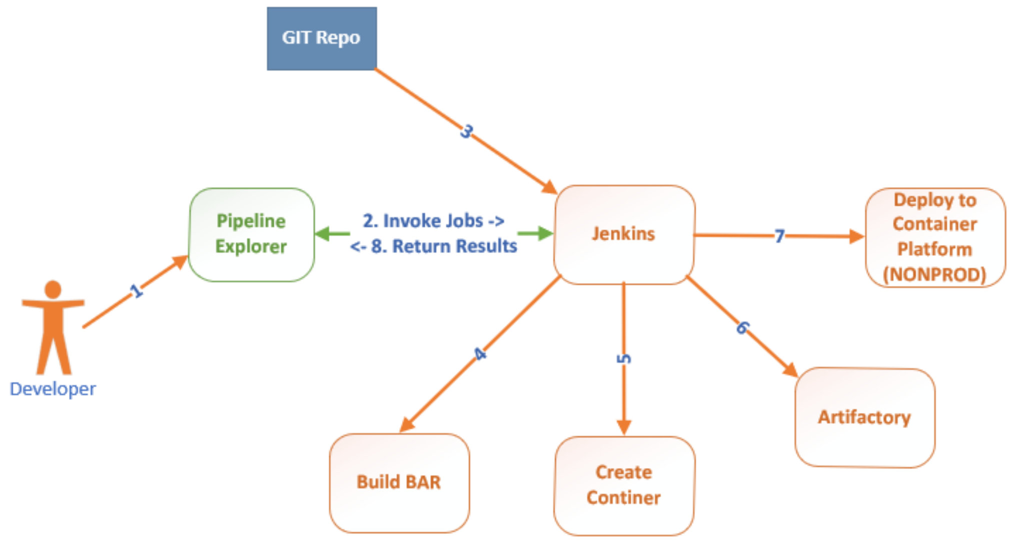
7.1 NextGen - Version 1



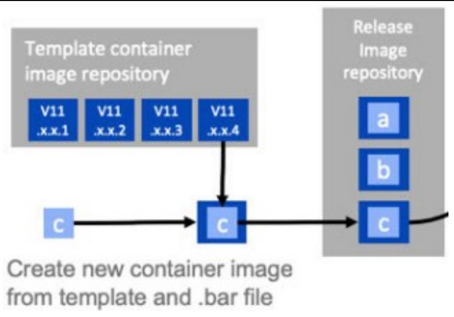
Overall, Pipeline v1 for NextGen Integrations is an introductory iteration for the organization. Activities such as Unit Testing through a Test Harness or Code Quality/Code Reviews take place outside of the Pipeline. As an introduction version, Pipeline will be used for basic “seeding” of the Pipeline Explorer Configuration Database as its role is to be the one source to know what integrations/flows are deployed.

This is important so the new Pipeline begins to play a role in promoting from NONPROD to PROD environments. Each time the Pipeline is used, it contains more information about the ESB in use. Learn more about Pipeline Explorer in **Section 4**.

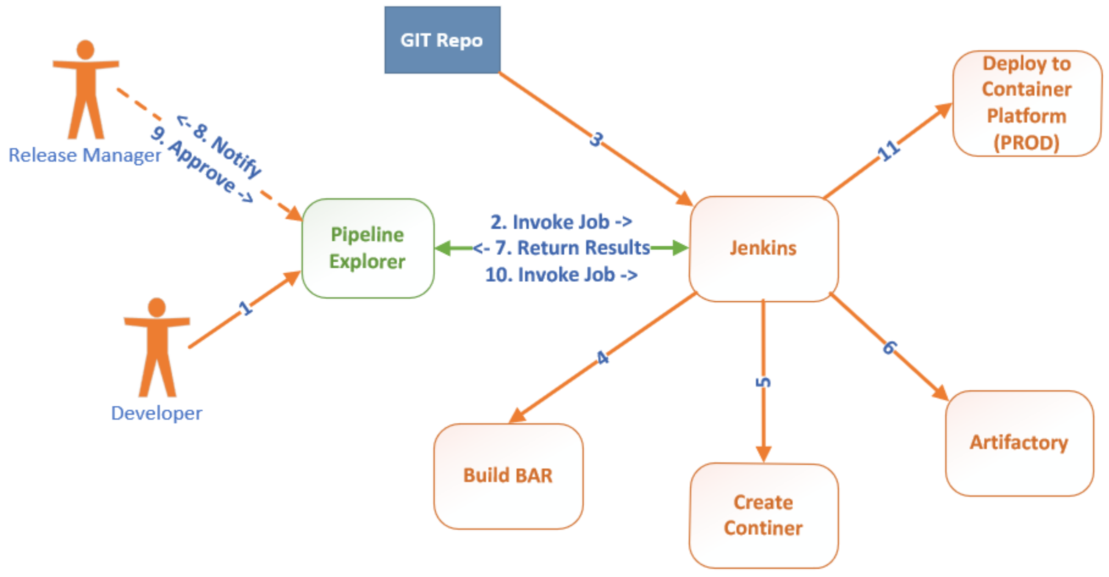
Activities for NONPROD



Step Number	Description	Notes
1	Create a "Change Package" indicating source location (GIT Repo), Config File location on GIT to BAR File Creation/Overrides for NONPROD, and Target Container Platform	The GIT Repo will include additional config file details and other artifacts to support the Pipeline. The intentions are these config files contain details (in JSON or YAML) on how to build and override BAR files.
2	Pipeline Explorer invokes Jenkins via REST API with information provided in the "Change Package" and other details contained within its internal Config Database	The "Change Package" has an ID for tracking
3	Jenkins, though custom scripting, checks out the necessary source files from GIT, performs some dependency checks to determine build parameters are satisfied	Can be a combination of Jenkins plugins and custom scripts within the jenkinsfiles
4	Build the BAR with appropriate overrides	Build BAR activities can be performed as an ANT or bash script.
5	Create Container and associated config artifacts supporting the new NONPROD integration	The base container type could be Podman as it is OCI (Open Container Initiative) compliant and be deployed to cloud or on-prem container platforms such as OpenShift. Other examples of containers are Docker.

6	Check BAR file(s) and Container Build for NONPROD into Artifactory	 Template container image repository V11 .x.x.1 V11 .x.x.2 V11 .x.x.3 V11 .x.x.4 Release Image repository a b c c c Create new container image from template and .bar file Example of Steps 5 and 6 Source: IBM Redbook: Accelerating Modernization with Agile Integration
7	Deploy to Target NONPROD Container Platform	
8	Results are returned to Pipeline Explorer to update its config database and for the Developer to determine next steps	A Developer next step outside of the pipeline would be to manually run the Unit Test app to ensure the deploy was successful

Activities for PROD



Step Number	Description	Notes
1	Developer Updates the Change Package to include Config File location on GIT for BAR	The GIT Repo will include additional config file details and other artifacts to

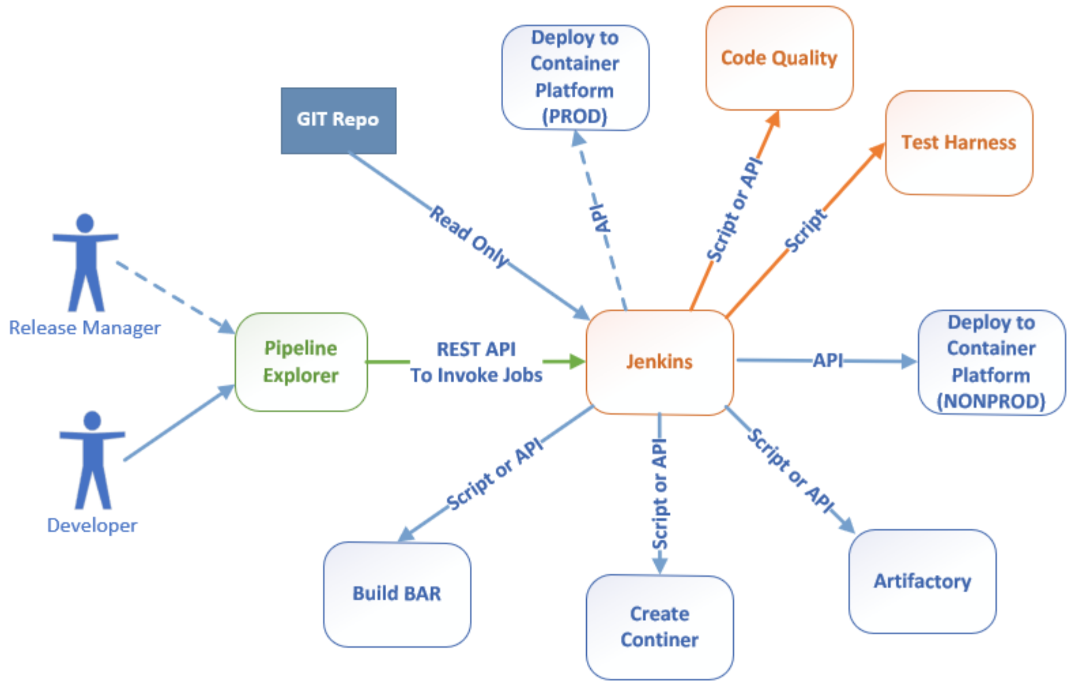
	File Overrides for PROD and Target Container Platform	support the Pipeline. The intentions are these config files contain details (in JSON or YAML) on how to build and override BAR files.
2	Pipeline Explorer invokes Jenkins via REST API with information provided in the "Change Package" and other details contained within its internal Config Database	The "Change Package" has an ID for tracking
3	Jenkins, though custom scripting, checks out the necessary source files from GIT, performs some dependency checks to determine build parameters are satisfied. Also, it interacts with Artifactory to grab the BAR file already stored from the NONPROD process to use in Step 4	Can be a combination of Jenkins plugins and custom scripts within the jenkinsfiles
4	Jenkins performs activities to build the BAR with appropriate overrides for PROD	Build BAR activities can be performed as an ANT or bash script.
5	Create Container and associated config artifacts supporting the new PROD integration	The base container type could be Podman as it is OCI (Open Container Initiative) compliant and be deployed to cloud or on-prem container platforms such as OpenShift.
6	Check BAR file(s) for QA into Artifactory	
7	Results are returned to Pipeline Explorer to update its config database.	If the build is successful, Step 8 is performed. If the build fails, the developer resolves issues to re-submit Change Package
8	If the build is successful from Step 7, then Package Explorer sends an email notification to the Release Manager	
9	The Release Manager, after performing the code review and other activities, logs into Package Explorer to approve/reject the Change Package	This action notifies the Developer their Change Package is approved or rejected
10	When the Change Package is approved, this action invokes another Jenkins job described in Step 11	
11	Jenkins interacts with Artifactory to get the Container file and deploys to the PROD Container Platform	Pipeline Explorer Config Database is updated. Outside of the Pipeline, the Developer perform post-deploy

		validation

Activities outside of Pipeline

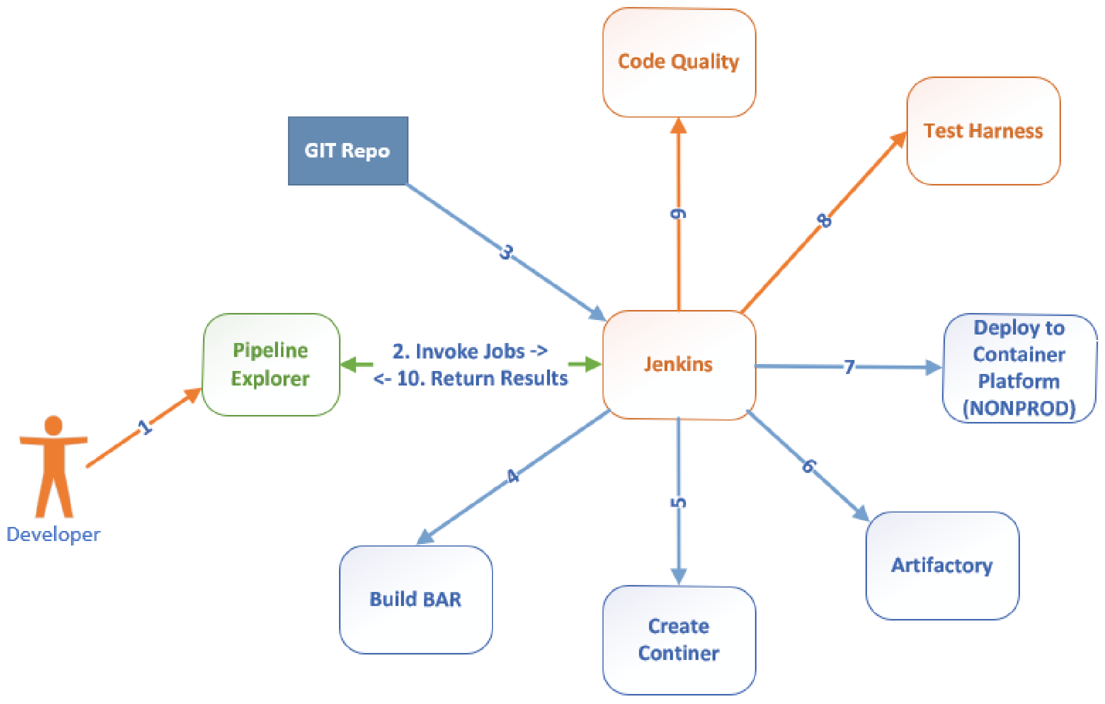
The Developer may have a need to create a ServiceNOW request for related tasks depending on needs. Post-deploy validation, UAT and establishing monitoring activities are performed outside the Pipeline.

7.2 NextGen - Version 2



Pipeline v2 for NextGen Integrations builds upon its predecessor to include a Test Harness and Code Quality. The Pipeline Explorer Configuration Database continues to grow as its role is to be the one source to know what NextGen integrations/flows are deployed. This is important so the new Pipeline begins to play a role in promoting to NONPROD and PROD environments. Each time the Pipeline is used, it contains more information about the NextGen in use.

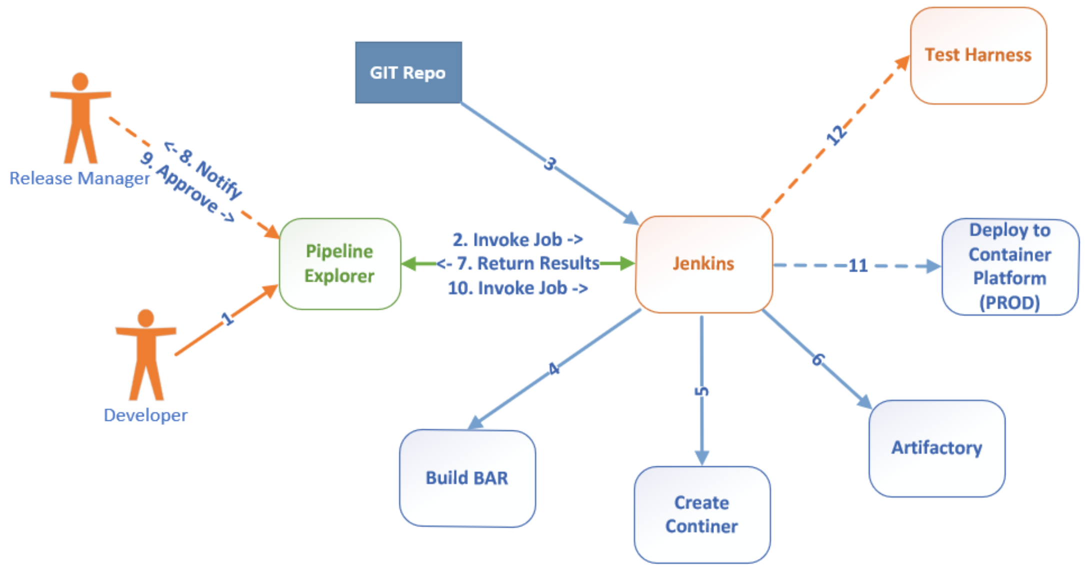
Activities for NONPROD



Step Number	Description	Notes
1	Create a "Change Package" indicating source location (GIT Repo), Config File location on GIT to BAR File Creation/Overrides for NONPROD, and Target Container Platform	The GIT Repo will include additional config file details and other artifacts to support the Pipeline. The intentions are these config files contain details (in JSON or YAML) on how to build and override BAR files.
2	Pipeline Explorer invokes Jenkins via REST API with information provided in the "Change Package" and other details contained within its internal Config Database	The "Change Package" has an ID for tracking
3	Jenkins, though custom scripting, checks out the necessary source files from GIT, performs some dependency checks to determine build parameters are satisfied	Can be a combination of Jenkins plugins and custom scripts within the jenkinsfiles
4	Build the BAR with appropriate overrides	Build BAR activities can be performed as an ANT or bash script.
5	Create Container and associated config artifacts supporting the new NONPROD	The base container type could be Podman as it is OCI (Open Container Initiative) compliant and be deployed

	integration	to cloud or on-prem container platforms such as OpenShift
6	Check BAR file(s) and Container Build for NONPROD into Artifactory	
7	Deploy to Target NONPROD Container Platform	
8	**Jenkins invokes the Test Harness as part of Unit Testing to verify the flow was successful. The output HTML report could be either put into GIT by Jenkins, or stored in the Pipeline Explorer Config Database**	**Unit testing with the custom Java App already in use can be used for this purpose. The Unit Test app acts as a test harness for message injection to the flow(s), and review the HTML output report of the results**
9	**Jenkins invokes a routine to do an automated code review of the flows and ESQL.**	**Examples of products to assist with the automated ACE code review are SonarCube with BCT Plugin (3rd party product).** **An alternative could be simple inspection of flow names to ensure they adhere to standards. This could be written even with bash scripts as part of a POC.** **This would be considered a basic code review and not a replacement for the code review process already in place performed by peers.**
10	Results are returned to Pipeline Explorer to update its config database and for the Developer to determine next steps	

Activities for PROD



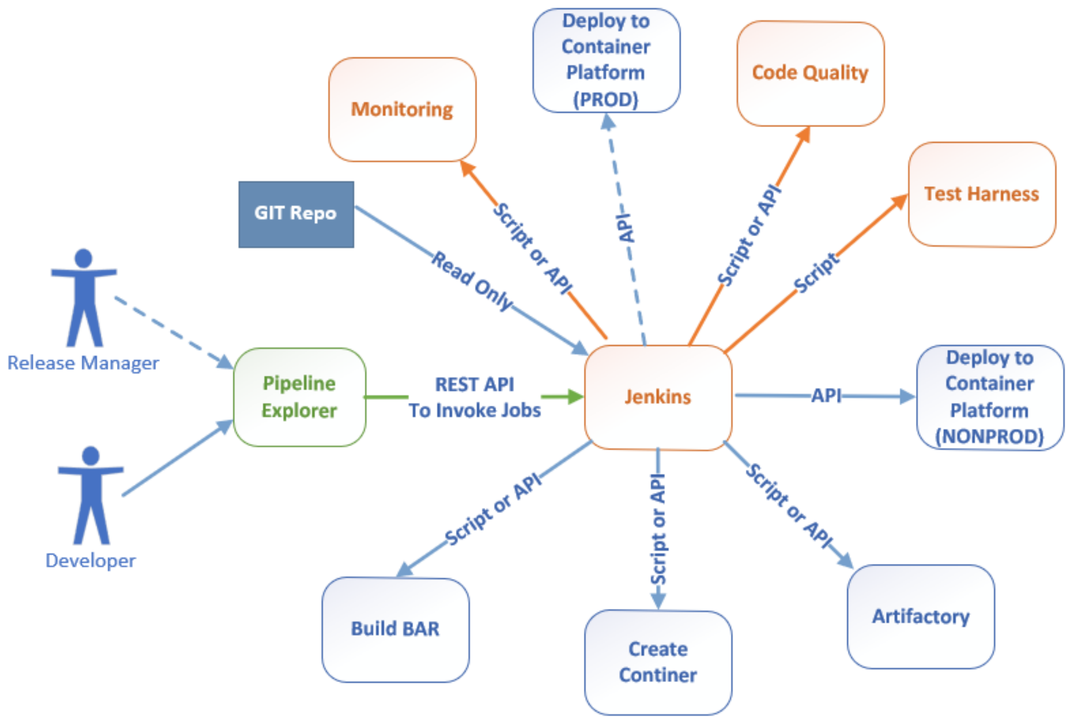
Pipeline v2 activities for PROD are essentially the same as in v1. Refer the steps from **Section 7.1** to describe these steps in more detail. The only modifications made at this point would be:

- Additional checks made to ensure the Unit Test and the Code Quality as part of NONPROD deploy was complete. This could be stored in the config database as a Boolean (true/false the unit test was performed, etc.)
- Step 12 (Test Harness to validate PROD) – This could be an *optional* part of an iterative post-install process to do basic validations of the deployment.

Activities outside of Pipeline

The Release Manager performs a code review and manually checks the unit test reports for quality before Approval/Rejection. When the deploy to PROD is complete, UAT and establishing monitoring activities are also performed outside the Pipeline.

7.3 NextGen - Version 3



Pipeline v3 for NextGen Integrations builds upon its predecessor with enhancements to:

- The Test Harness to include more functionality for Unit Tests
- Code Quality (TBD depending on tools used)
- Establish monitoring rules on the deployed container

The Pipeline Explorer Configuration Database continues to grow as its role is to be the one source to know what NextGen integrations/flows are deployed.

Through iterative Pipeline versions, more of the manual tasks performed could be automated.

8. Recommendations for Agile Integration and Containers

Implementing an Enterprise-Oriented container management system such as *OpenShift with IBM Cloud Pak for Integration (CP4i)* may be a logical choice for your organization. A few reasons for suggesting this:

- They're IBM products and could be implemented in multi-cloud environments supporting OpenShift (I.E. AWS, Azure, GCP)
- Since CP4i are containers built by IBM, they'll provide defect support (NOTE: you can build your own containers – and many organizations do build on their own. IBM will

support the products you put on them (MQ, ACE) but they will not support the container itself if there are defects such as security issues.)

- CP4i provides IBM written operators for OpenShift to be used to deploy CP4i capabilities and runtimes. Operators extend a Kubernetes cluster by adding and managing additional resources in the Kubernetes API. Learn more here: https://www.ibm.com/docs/en/cloud-paks/cp-integration/2021.2?topic=installing-operators

As a long-term strategy, the use of OpenShift and CP4i should work well for managing containers at a large scale, but the Pipeline detailed within **Sections 3 (CI/CD Pipelines), 4 (Pipeline Explorer) and 7 (Pipeline for NextGen Integration)** would still need to be constructed. In short, what CP4i provides is more of the deploy and operational nature of containers and not what the above sections provide.

From a practical point of view, the “runway” and skills needed to stand up such container management systems may not be feasible in the short term. This alone can hamper business initiatives that NextGen Integrations can provide.

In order to move forward with NextGen Integrations, and to create value from early iterations of the Pipeline, it’s recommended to use an open source, IBM and RedHat endorsed container called Podman. *The experience gained with this approach can be leveraged with OpenShift, Kubernetes or even other OCI compliant cloud deployment options.*

The Open Container Initiative (OCI) is a lightweight, open governance structure (project), formed under the Linux Foundation, for the express purpose of creating open industry standards around container formats and runtime. The OCI was launched on 2015 by Docker, CoreOS and other leaders in the container industry.

References:

https://www.ibm.com/cloud/integration/agile-integration/

https://community.ibm.com/community/user/integration/blogs/kim-clark1/2021/11/23/iib-ace-series

https://opencontainers.org/

9. QuickStart for Container Operations

Linux containers have emerged as a key open-source application packaging and delivery technology, combining lightweight application isolation with the flexibility of image-based deployment methods.

9.1 Building, Running and Managing Containers on RHEL

Red Hat provides a set of command-line tools that can operate without an enterprise container engine. These include:

- podman: for directly managing pods and container images (run, stop, start, ps, attach, exec, etc.)

- buildah: for building, pushing, and signing container images

- skopeo: for copying, inspecting, deleting, and signing images
- runc: for providing container run and build features to podman and buildah
- crun: an optional runtime that can be configured and gives greater flexibility, control, and security for rootless containers

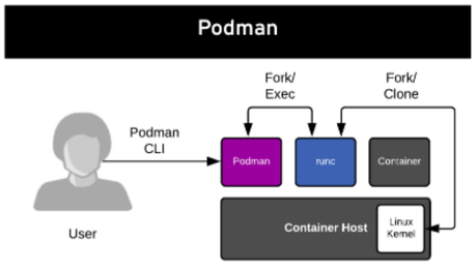
Because these tools are compatible with the Open Container Initiative (OCI), they can be used to manage the same Linux containers that are produced and managed by Docker and other OCI-compatible container engines such as OpenShift. However, they are especially suited to run directly on Red Hat Enterprise Linux, in single-node use cases.

The main advantages of these tools include:

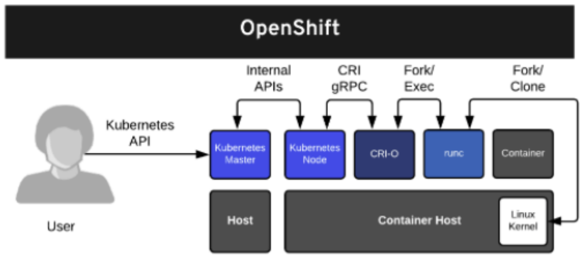
- Running in rootless mode: rootless containers are much more secure, as they run without any added privileges
- No daemon required: these tools have much lower resource requirements at idle, because if you are not running containers, Podman is not running. Docker, on the other hand, have a daemon always running
- Native systemd integration: Podman allows you to create systemd unit files and run containers as system services

The concepts for container management listed above can be incorporated into the Pipeline for NextGen to gain experience in NONPROD and PROD for on-prem.

9.2 More about Podman



How containers run with a container engine

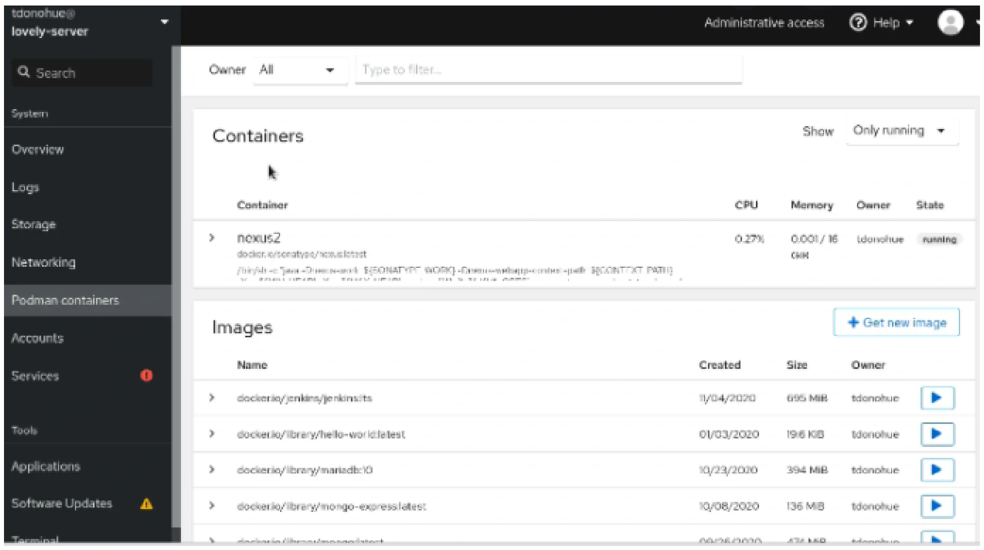


How containers run in a Kubernetes cluster

Podman was released with Red Hat Enterprise Linux 7.6 and 8.0 as the next generation of Linux container tools, is designed to allow faster experimentation and development of features. OpenShift shares many of its underlying components with Podman. This allows developers to leverage knowledge gained in experiments conducted in Podman for new capabilities in OpenShift.

Podman provides a Docker-compatible command line and provides a socket activated REST API service to allow remote applications to launch on-demand containers. This REST API also supports the Docker API, allowing users of docker-py and docker-compose to interact with the Podman as a service.

9.3 Administer Podman Containers from Cockpit



Containers

Container	CPU	Memory	Owner	State
nexus2 docker.io/sonatype/nexus:latest	0.27%	0.001 / 16 GiB	tdonohue	running

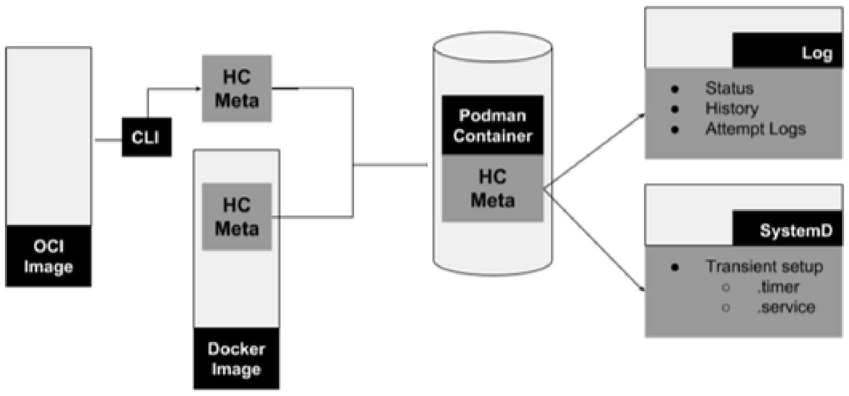
Images

Name	Created	Size	Owner
docker.io/jenkins/jenkins:lts	11/04/2020	695 MiB	tdonohue
docker.io/library/hello-world:latest	01/03/2020	19.6 KiB	tdonohue
docker.io/library/mariadb:10	10/23/2020	394 MiB	tdonohue
docker.io/library/mongo-express:latest	10/08/2020	136 MiB	tdonohue

Podman containers can be managed from the command line or through a web-based app called Cockpit.

Cockpit is a server administration tool sponsored by Red Hat, focused on providing a modern-looking and user-friendly interface to manage and administer servers. It is a web-based graphical interface for servers that can be used by admins to administer Podman containers.

9.4 Podman Healthcheck



A “healthcheck” is a way in which a user can determine the status of the primary process running inside of a container, such as ACE. It is more than a simple “is my container running?” question as it's more like “is my application ready?” So, a healthcheck is really a way to verify that both the container and its applications are responsive.

Podman’s implementation of healthchecks can be categorized in three pieces:

1. Image and container metadata

2. Logging

3. Scheduling

The OCI Image Specification does not include a healthcheck definition. The fact that an OCI image cannot embed or retain the healthcheck metadata actually has resulted in a very flexible approach to defining them, unlike with Docker where the metadata is embedded in a container.

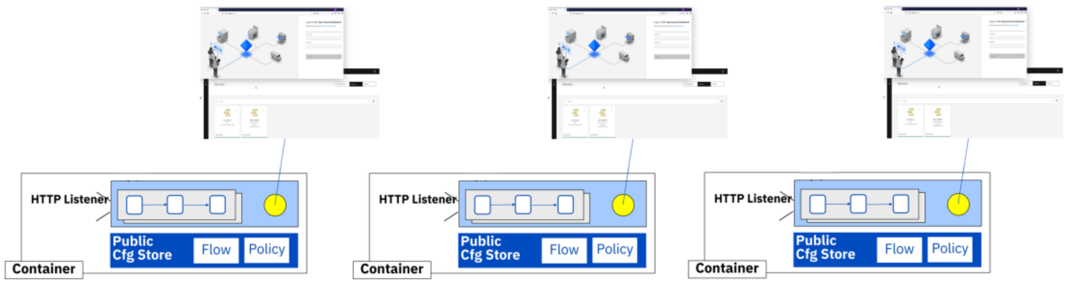
A log file for healthchecks is also created. It retains the container's health status as well as history including previous healthcheck attempts. And finally, the scheduling is done by systemd. When a container starts and has a healthcheck, Podman performs a transient setup of a service and timer file.

```
$ sudo podman run -dt --name hc --healthcheck-start-period 2m --healthcheck-retries 5 --healthcheck-command "CMD-
SHELL curl http://localhost  || exit 1" quay.io/libpod/alpine_nginx:latest
164747229c31eb0214c2aa63997171104e38f93d8aa4d5e315b16679213c078d
```

```
$ sudo podman inspect hc
[
    {
        "ID": "164747229c31eb0214c2aa63997171104e38f93d8aa4d5e315b16679213c078d",
        "Created": "2019-03-26T14:27:21.928727195-05:00",
        "Path": "nginx",
        "Args": [
            "-g",
            "daemon off;"
        ],
        "State": {
            "OciVersion": "1.0.1-dev",
            "Status": "running",
            "Running": true,
            "Paused": false,
 ...
            "StartedAt": "2019-03-26T14:27:22.439699271-05:00",
            "FinishedAt": "0001-01-01T00:00:00Z",
            "Healthcheck": {
                "Status": "healthy",
                "FailingStreak": 0,
```

You can use Podman to interact with healthcheck results and status. The most direct way is using podman inspect on the container. The example above is to test nginx in a container, but this could be modified and used to test ACE apps, too. Since these are commands, they can be put into a custom app to collect or even an AppDynamics plugin to part of an organization's enterprise monitoring strategy.

9.5 Administer ACE Message Flows



Each ACE instance within a container provides access to the Admin Web UI, if enabled. This allows developers and admins to view details of deployed flows for each container.

```
2021-10-14 07:00:32.293988: BIP2269I: Deployed resource 'Echo' (uuid='Echo',type='MessageFlow')
started successfully.
2021-10-14 07:00:33.050416: BIP2866I: IBM App Connect Enterprise administration security is
inactive.

2021-10-14 07:00:33.065014: BIP3132I: The HTTP Listener has started listening on port '7600' for
'RestAdmin http' connections.
2021-10-14 07:00:33.066482: BIP1991I: Integration server has finished initialization.
2021-10-14T07:00:37.217Z Integration server is ready
2021-10-14T07:00:37.217Z Metrics are disabled
2021-10-14T07:00:37.217Z Starting integration server commands API server
2021-10-14T07:00:37.217Z Integration API started
```

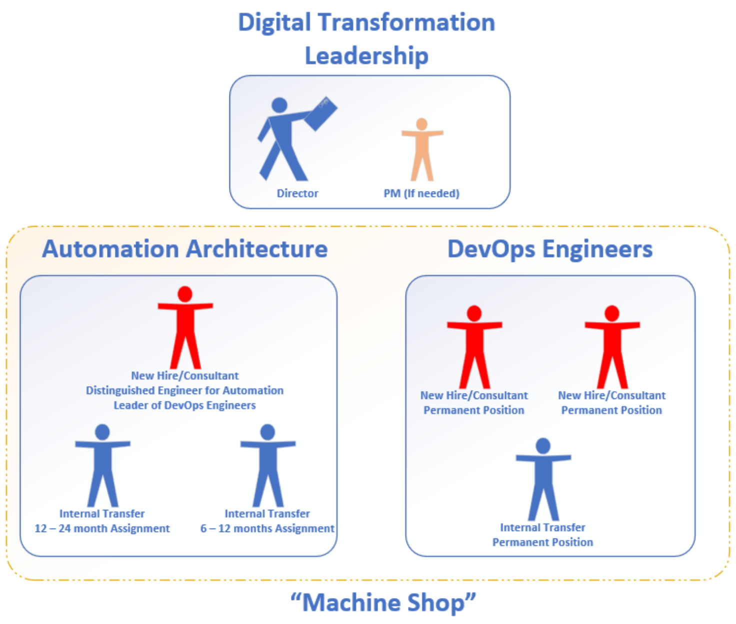
Another option would be to connect to an individual container's command prompt. This way, you can inspect the filesystem including the ACE logs.

Reference:

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/building_running_and_managing_containers/assembly_starting-with-containers_building-running-and-managing-containers

10. Responsibility Assignment Matrix for Pipeline

DevOps (which includes CICD initiatives) has its best success when it's a coordinated, multi-team effort. At its heart DevOps is about empowerment of developers and operations to work together, accelerating construction/testing/delivery of new business initiatives.



This should be considered a strategic endeavor requiring formation of new teams: The ***Automation Architecture Team*** and ***DevOps Engineering Team.*** Some organizations where such teams operate are collectively called a “***Machine Shop***” as that’s what they do: a workshop for making or repairing mechanical items.

10.1 The Automation Architecture Team

The ***Automation Architecture Team*** charter is to design and develop strategies that will allow an organization to automate its processes. They work with company leadership, departmental stakeholders, and IT departments to see the holes in an automation process before it starts.

Automation architects are problem solvers and creative thinkers who understand both the business and technology sides of an organization’s work environment. Basic recommendations for this team:

- Team size should be 3 members reporting to some director-level resource overseeing digital transformation. The director should already be someone with that role within your organization to give credibility and guidance to this team

- 2 to 1 ratio for internal vs external hires (this provides a composite of team members who have experience internally with the company’s business and technology side, and new team members (external hires) who bring their perspective and experience to automation)

- At least one of the team members should have a proven track record for automation and recognized as a contributor in the space through published articles, webinars, open-source contributions and work experience. Most likely, this will be your new hire and will be the Distinguished Engineer for Automation and leading/contributing to the DevOps Engineering team

- The team itself is an accelerator for a unified DevOps strategy for the organization. Its formation and charter may only need to exist for two to three years to design and develop strategies that will allow an organization to automate its processes. Team members can eventually move to the DevOps Engineering or application teams, depending on needs

10.2 The DevOps Engineering Team

The ***DevOps Engineering Team*** charter is to be 1) the product owners and implementers of the tools and processes chosen by the Automation Architecture team and 2) advisors/consultants to application teams requiring their skills to collaborate on building out key pieces of a Pipeline.

This team's efforts reduce DevOps complexity, closing the gap between actions needed to quickly change an application, and the tasks that maintain its reliability. Basic recommendations for this team:

- Keep team size around 3 (minimum) and 6 members (max), reporting to the Distinguished Engineer of Automation in the Automation Architecture Group
- 2 to 1 ratio for external vs internal hires (this provides a composite of team members who have experience internally and new team members (external hires) who bring their perspective and experience to automation)
- At least half of the team members should have a relevant work experience for automation
- Team members must have skills that span both development and operations, including interpersonal skills to help bridge divides between siloed application teams
- This team should be considered permanent compared to the Automation Architecture Team, as this team will be the product owners of the Pipeline for future enhancements and rollout to additional application teams

10.3 RACI Chart for ESB Pipeline

ESB Pipeline	Developer Team	Admin Team	DevOps Engineering Team	Automation Architecture Team	Reference Sections
Pipeline Explorer	C	I	A	R	4.2 6
GIT Repository Changes*	A	I	C	R	n/a
Jenkins	C	I	A	R	3.3 5 6
Build BAR	RA	I	C	I	5 6
Artifactory	C	I	A	R	3.4 5 6
IIB Runtime Deploy	RA	C	C	I	5 6
Test Harness	RA	I	C	C	3.5 6.2 6.3

Code Quality	C	I	A	R	3.6 6.2 6.3
ServiceNow	C	C	A	R	6.2 6.3
Monitoring	C	A	C	R	n/a

10.4 RACI Chart for ESB Pipeline

NextGen Pipeline	**Developer Team**	**Admin Team**	**DevOps Engineering Team**	**Automation Architecture Team**	**Reference Sections**
Pipeline Explorer	C	I	A	R	4.2 7
GIT Repository Changes*	A	I	C	R	n/a
Jenkins	C	I	A	R	3.3 5 7
Build BAR/zip distro	RA	I	C	I	5 7
Create Container	**A**	**C**	**C**	**R**	**5** **7** **9**
Artifactory	C	I	A	R	3.4 5 7
Deploy to Container Platform	**C**	**A**	**C**	**R**	**5** **7** **9**
Test Harness	RA	I	C	C	3.5 7.2 7.3
Code Quality	C	I	A	R	3.6 7.2 7.3
ServiceNow	C	C	A	R	n/a
Monitoring	C	A	C	R	7.3

10.5 RACI Chart Notes

R = Responsible, A = Accountable, C=Consulted, I=Informed

* - may or may not be any related changes **n/a** – no reference in doc

RED Text – Represents tasks that are unique may not be cross-functional between respective Pipelines

Responsible: Doing the Task	This team actions the task or deliverable. They are responsible for making the decisions.
Accountable: Owning the Task	This team or role is responsible for the overall completion of the task or deliverable.
Consulted: Assisting	This team provides information useful to complete the task or deliverable. There will be two-way communication between those responsible and those consulted. This team are often the subject matter expert.
Informed: Keeping Aware	This team will be kept up to date on the task or deliverable. This could be on progress, or when the task or deliverable is completed. They may not be asked to give feedback or review, but they can be affected by the outcome of the task or deliverable.

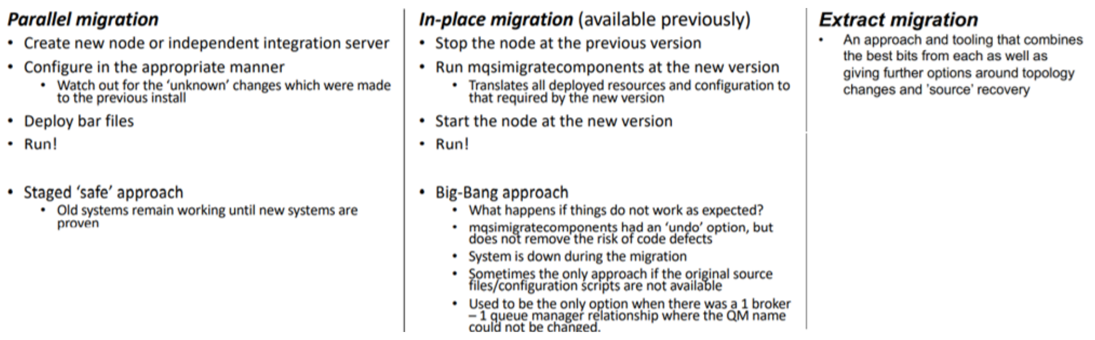
11. ESB Upgrade Strategy

ACE v12 and above combines the existing IIB concepts for integrations along with new cloud-based composition capabilities. However, a more fundamental change is the continued focus on enabling container-based deployment of the on-premises software runtime. ACE does not mandate a move to containers, so one can deploy workloads in the more centralized ESB pattern, if desired.

Many organizations have a large investment in place with the current ESB. It is possible with the time and resources to completely re-architect the ESB to NextGen with the use of containers. From a practical standpoint, it's advised to make incremental changes to this topology over time as the ESB serves its purpose with core systems. That said, there are opportunities for basic changes prior to migration.

As a note: There is no change to the Eclipse project types between IIB and ACE. Application, Library and Shared Library are the same.

11.1 Types of Migrations for ACE



Parallel migration

- Create new node or independent integration server
- Configure in the appropriate manner
  - Watch out for the 'unknown' changes which were made to the previous install
- Deploy bar files
- Run!

- Staged 'safe' approach
  - Old systems remain working until new systems are proven

In-place migration (available previously)

- Stop the node at the previous version
- Run mqsimigratecomponents at the new version
  - Translates all deployed resources and configuration to that required by the new version
- Start the node at the new version
- Run!

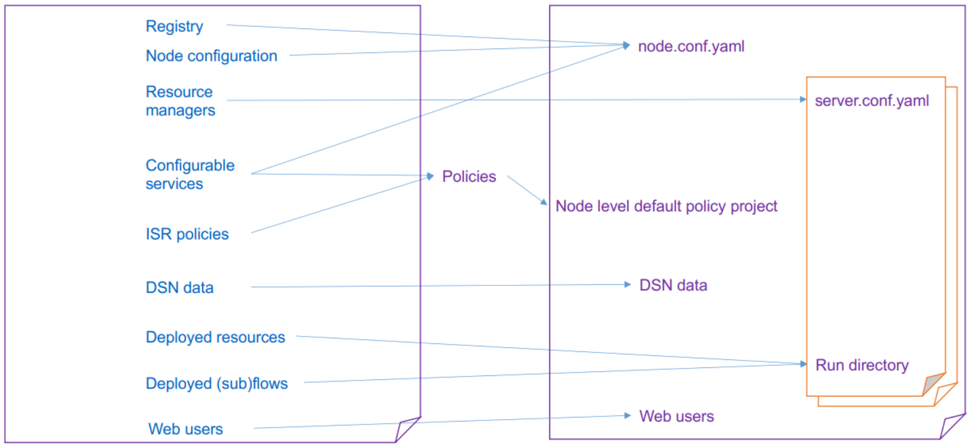
- Big-Bang approach
  - What happens if things do not work as expected?
  - mqsimigratecomponents had an 'undo' option, but does not remove the risk of code defects
  - System is down during the migration
  - Sometimes the only approach if the original source files/configuration scripts are not available
  - Used to be the only option when there was a 1 broker – 1 queue manager relationship where the QM name could not be changed.

Extract migration

- An approach and tooling that combines the best bits from each as well as giving further options around topology changes and 'source' recovery

ACE v12 and above itself offers three paths for IIB upgrades: Parallel Migration, In-Place Migration and Extract Migration. Parallel and In-Place have been part of upgrade strategies for years. Extract migration is new with ACE and the recommended option.

11.2 Extract Migration



The Extract Migration process comes with a new command called mqsiextractcomponents, where extraction of configuration and resources come from an IIB backup file the admin creates. This produces an integration server work directory containing yaml and other files to be used by ACE. The process is repeatable, cross-platform, and can be used to migrate to different topologies or even cloning integration nodes. This is a major step forward for DevOps enablement.

For Extract Migration, two options are available:

- Parallel Migration
  - Install ACE environment
  - Use mqsiextractcomponents to configure ACE (this will help to create policies for you from config services for example) to help ensure you don't forget configurations from IIB
  - Discard the run directory contents
  - Extract a whole node to get the configuration, move out all of the directories (integration servers) in the server's directory, then move each server back in one by one to migrate a server at a time
  - To pick-up any changes since the original backup, just redo the backup and run extract again targeting a temporary node, then just copy the newly extracted files or settings into your original node/server
- In place
  - Install ACE environment

- Use mqsiextractcomponents to configure ACE (this will help to create policies for you from config services for example) to help ensure you don't forget configuration from IIB
- Don't discard the run directory contents when complete as you'll still need them

NOTE: mqsiextractcomponents is not a complete like for like replacement of mqsimigratecomponents in-place approach, but the same result can be achieved

As a recommendation, using ***Extract Migration with the side-by-side is preferred***.

11.3 Pre- and Post-Upgrade Activities

The following represents a summary of activities to consider:

- Survey of ESB PROD flows to determine

    1) Flows that can be retired

    2) Flows that can be re-located to other ESB EGs

    3) Flows to be either consolidated or divided up more for functionality

    4) Flows that are candidates to move to NextGen

- Perform those changes (for 1, 2 and 3) in IIB before upgrading

- Create a Test Harness/Unit Test Java App to be functional with ACE

- Perform a side-by-side extract migration from IIB to ACE for ESB Integrations

- Migrate #3 flows to NextGen architecture

11.4 Considerations for Developing IIB Flows Moving Forward

This subsection provides general guidance and recommendation in regards to any net-new IIB integrations created, in an effort to ensure a smoother transition to ACE when that time comes. ACE does not mandate a move to containers, so you can continue deploying workloads in the more centralized ESB, hub-and-spoke pattern supporting core systems as they are today. That said, here are some "work smart" suggestions for any net-new IIB integrations and flows:

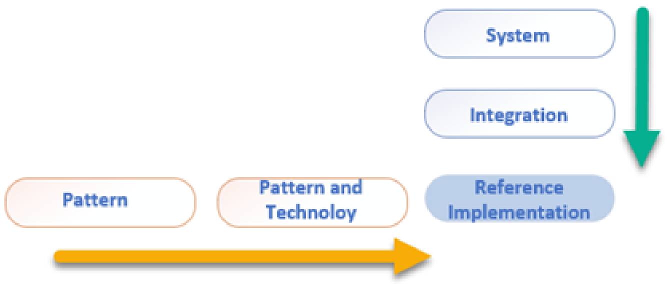
- Ensure nodes such as SCA (SCAInput, SCAOutput, etc.), Decision Services, and PHP are no longer used. These nodes have been removed in ACE v11 and above

- Consider use of MQ Client connections instead of Local Queue Manager connections with MQ nodes (MQInput, MQOutput, etc.) This is to prepare developers to think in terms of the message broker (ACE) is no longer on the same server as MQ. Thinking/developing this way also works for NextGen integrations with containers

- Create flows and associated objects as part of being either an Application or Library, instead of Independent Resources. Thinking/developing this way also works for NextGen integrations with containers

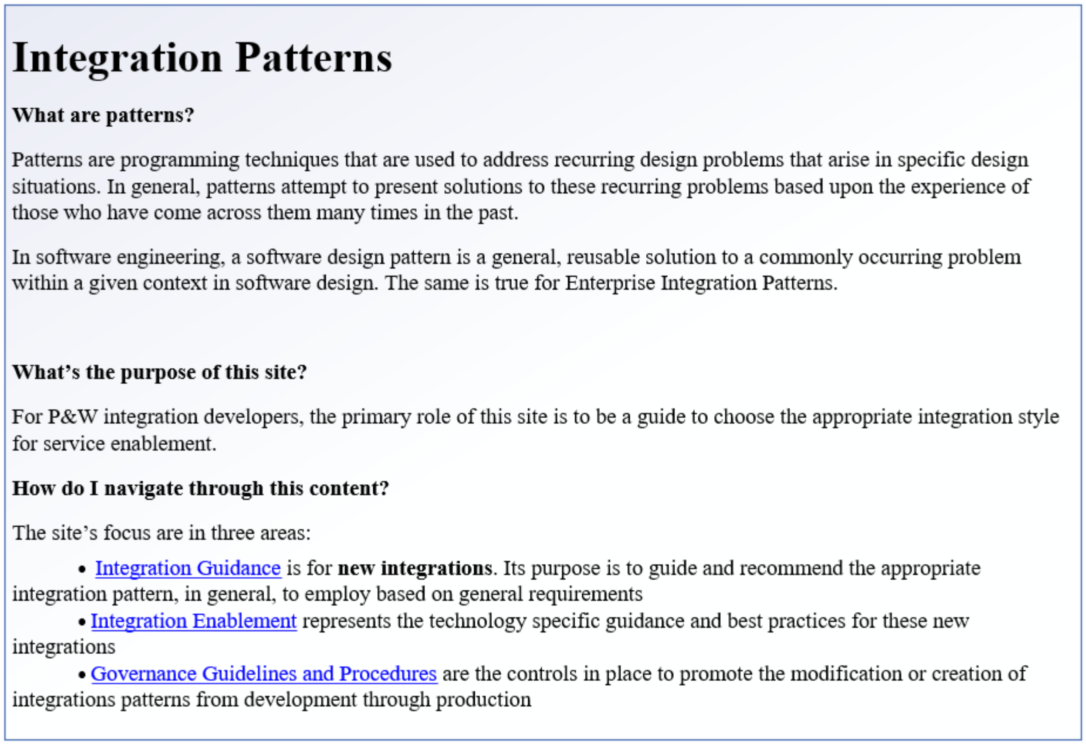
- When designing new flows, identify the ones that can be created that are more isolated/independent of other core flows, and separate into their own execution group; these are your candidates for NextGen integrations

12. Integration Patterns Catalog



A key architecture objective for the integration catalog is to ensure successful development and delivery of new high performing integrations based on successful integration patterns already in use at your shop, or leveraging integration patterns used throughout industry.

12.1 Confluence's Role with the Catalog



Integration Patterns

What are patterns?

Patterns are programming techniques that are used to address recurring design problems that arise in specific design situations. In general, patterns attempt to present solutions to these recurring problems based upon the experience of those who have come across them many times in the past.

In software engineering, a software design pattern is a general, reusable solution to a commonly occurring problem within a given context in software design. The same is true for Enterprise Integration Patterns.

What's the purpose of this site?

For P&W integration developers, the primary role of this site is to be a guide to choose the appropriate integration style for service enablement.

How do I navigate through this content?

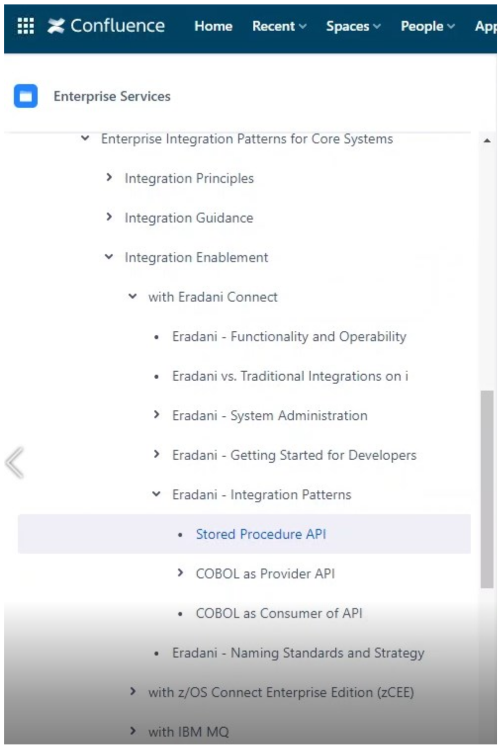
The site's focus are in three areas:

- Integration Guidance is for **new integrations**. Its purpose is to guide and recommend the appropriate integration pattern, in general, to employ based on general requirements
- Integration Enablement represents the technology specific guidance and best practices for these new integrations
- Governance Guidelines and Procedures are the controls in place to promote the modification or creation of integrations patterns from development through production

Confluence is a collaborative tool for teams and can be used as the starting point for the Integration Patterns Catalog. Examples of use would be to provide:

- Integration guidance for new integrations on how to make a decision on the type of pattern to employ
- Integration enablement on the "how to" for a pattern with a given technology (ACE, APIc, MQ, etc.)
- Governance and procedures to explain the SDLC

12.2 Example of use within Industry



Enterprise Services

- Enterprise Integration Patterns for Core Systems
  - Integration Principles
  - Integration Guidance
  - Integration Enablement
    - with Eradani Connect
      - Eradani - Functionality and Operability
      - Eradani vs. Traditional Integrations on i
      - Eradani - System Administration
      - Eradani - Getting Started for Developers
      - Eradani - Integration Patterns
        - Stored Procedure API
        - COBOL as Provider API
        - COBOL as Consumer of API
      - Eradani - Naming Standards and Strategy
    - with z/OS Connect Enterprise Edition (zCEE)
    - with IBM MQ

A large financial services company uses the Integration Patterns Catalog concept as part of their API enablement strategy on IBM i and z/OS platforms. Additions were made beyond the catalog content to include (for each category of technology):

- Getting started for Developers
- Functionality and Operability
- Naming Standards and Coding Conventions

An Integration Patterns Catalog in use can help accelerate new development for ESB and NextGen Integrations and be part of other strategic initiatives within the organization.

13. Conclusion

We at TxMQ, Inc. appreciate your interest in learning more about modern, practical CICD Pipelines with IBM App Connect Enterprise and IBM MQ.

Want to learn more? Reach out to us at TxMQ, Inc. Our deep industry expertise allows us to utilize subject matter experts to solve your most complex challenges. TxMQ, Inc. has solutions and innovations helping you do business in an ever-changing world. We're here to guide your organization through the digital transformation journey.

We look forward to partnering with you in future endeavors to help get the most out of IBM MQ, App Connect Enterprise, a CICD Pipeline and everything in-between.

TxMQ, Inc.

Imagine. Transform. Engage.

We're here to help you work smart in the new economy!