13 E-Business Suite Development Tips

DEVOPS FOR ORACLE APPS

Table of Contents

EBS Development Processes

The most effective development processes are tailored to the company and the product IT teams are working with. The most effective teams develop standardized processes that help reduce errors and speed up the software delivery lifecycle, while increasing efficiency for their specific product.

DevOps spans people, process, and technology and can be integrated into every IT team. Combined with best practices for each team's technical landscape, it can create more effective processes that deliver high-quality software to market in less time.

That's especially true for IT teams working on Oracle apps such as E-Business Suite. Combining DevOps practices with specific development tips for E-Business Suite will make entire teams more effective and produce greater results for end users.

At Flexagon, we've been helping IT teams integrate DevOps practices and tools into their current processes for years - and have deep expertise in Oracle technology, including E-Business Suite. We've combined our experience with E-Business Suite with general DevOps and development best practices to create a list of 13 development tips for Oracle EBS.

These tips will help developers standardize their processes, save time, and reduce errors throughout development, testing, and deployment to production. They may even contribute to your efforts to shorten the software delivery lifecycle, and will compliment continuous methodologies like continuous integration and continuous delivery.

Best Practices for E-Business Suite Development

1 Standardize Naming Conventions

Don't waste time naming database objects based on standard conventions, then using ad hoc naming for file system objects. Instead, come up with naming conventions that can be easily replicated in scripts, executables, reports, forms and more. By taking this approach from the beginning, any out-of-context coding, for example in a JIRA ticket, can be easily understood.

2 Use FNDLOAD for AOL Files

Oracle Application Object Library (AOL) is a collection of pre-built application components and facilities, which consists of forms, subroutines, concurrent programs, reports, database objects, messages, menus, responsibilities, flex field definitions, and various other library functions.

Virtually every EBS implementation will have AOL definitions and require synchronization of the data across EBS instances. Oracle provides a concurrent program, FNDLOAD, for both downloading and uploading AOL objects. This becomes the core building block for migrating the data across instances. Other, less desirable options include manual synchronization (error-prone) or RPA technology (complicated).

You can use FNDLOAD to download the source AOL definition as .LDT files from the source. You can also use FNDLOAD to upload the .LTD files to the target instance, completing the migration.

3 Track AOL Changes in Your SCM

Development teams most often implement AOL changes in a shared EBS Development environment. Since EBS does not keep track of revisions of changes, it is a best practice to extract each AOL change to an .LDT and record the revision as a commit into your source control system.

You can use DevOps tooling to automate the migration of the AOL changes using the FNDLOAD program. As with any file, you will have visibility into which revisions of your AOLs where deployed, when, where, and by whom.

4 Use SCM Systems and DevOps Tools to Manage Source Control Folders

Source Control Management (SCM) Systems, such as Git or Subversion, provide a platform for managing the lifecycle of documents and other files. Every change to a file is recorded with a revision number and provides easy recall of any change. Other common features include branching, merging, and revision history. A mature source control management practice is critical for an effective DevOps implementation.

The key capabilities include:

- **Visibility into the full history of every change made to a file - what changed, who changed it, and why**
- **Easily revert to any previous revision of a file**
- **No need to block all other users from updating a single file while you complete your changes, since every change is a revision and you can easily merge changes together**
- **Teams can work against a single file repository, without a requirement of explicitly keeping backups and keeping track of which files are the "latest"**

DevOps tools provide lineage with the SCM to provide insight into your SDLC, right down to the file level. After an issue is reported, you can easily identify who made a change to a configuration file at what time, as well as when that change was deployed to production. In comparing the change to the previous revision, you can see exactly what changed and view a description of why the change was made.

DevOps tools often provide pipelines for managing how changes are propagated across environments in your infrastructure, including capabilities for code reviews, approvals, scheduling, and more. Integration with the SCM enables you to associate the deployment requests to particular revisions of the source files, providing context for the request to support an approval (e.g. code reviews). And, since the ultimate process prevents issues in the first place, the lineage also arms operations teams with the data to satisfy IT auditors.

5 Simplify Server Management and Reduce Problems with Automation

Oracle E-Business Suite consists of many different servers, services, and workflows. Each of these components requires different handling to stop, start, restart, enable, or disable. Managing these different server components is not an easy task.

Automating server management is critical for production environments for the following purposes:

- **Determining which server/component to start instead of starting/stopping/ restarting all the servers**
- **Reducing downtime by replacing manual processes with automation**
- **Combining with deployment operations to ensure deployed changes go into effect**
- **Adding approvals and/or notifications to increase control and visibility**

6 Improve Versioning for Package Migration

Unfortunately, it's common for IT teams to migrate packages, then find out that they aren't working with the latest version of the code. Not only is that frustrating, it wastes time and extends your software delivery lifecycle.

Instead of struggling to identify which version of code you're working with, make reviewing package versions easier by adding a version code right after the header revision comments in your PL/SQL package. This can be done manually by the team, or using DevOps tools and continuous processes to ensure this is done each time code is migrated.

While some companies use version codes, DevOps tools are by far the preferred choice. The best DevOps tools allow you to clearly see what revisions of the source are included with the build and how they tie back to the source control system. Not only do they improve visibility, but they eliminate the manual versioning tasks that many teams without DevOps tools do regularly.

7 Structure Your Code

All coding – not only packages being readied for migrations – should follow standard protocols. The aim here is to make the application's code as simple and readable as possible. Use the case function or indent code consistently to make code fragments easy and predictable to read.

Whatever code structure you use, keep it consistent throughout your organization. Everyone should be able to easily view and understand code using the same system.

8 Know When to Use Patch vs. Run Edition

Oracle E-Business 12.2 and above comes with editions (Run and Patch). You can deploy your customizations directly to Run Edition or apply to Patch Edition (Online Patching) by following ADOP patching phases.

Online Patching cycle consists of the following major phases:

- PREPARE
- APPLY
- FINALIZE
- CUTOVER
- CLEANUP

Online patching reduces downtime and is also called 0 downtime deployment. You can also deploy changes directly into the run edition. This option should be used when changes are small or when the new online patching options are not adopted.

It's also important to note that while you can deploy the patch edition without runtime, applying it to the live system requires a cutover process that can take a significant amount of time, especially for large patches. For example, FlexDeploy customers typically deploy all customizations directly to the run edition.

9 Understand Post-Refresh Processes

Cloning/refresh is a process used to create an identical instance from an existing instance. It is also used to refresh an existing instance, (mostly a non-production EBS instance,) from another, (mostly production EBS,) instance. It is normal practice in the EBS-world to copy/refresh the EBS instance from production to one or all the non-production instances every few months. In the normal cloning process, the cloned instance's file system and DB gets replaced from source instance (prod).

Due to the clone/refresh all the files that were deployed recently (that are not in production yet) will get replaced with whatever is in production. Developers need to re-deploy all the lost files.

10 Create Reusable Frameworks

Your IT team doesn't need to spend time learning new development frameworks and standards unless there's a specific need. So, as you create frameworks to develop your current Oracle applications, be sure they can be replicated in future implementations.

For example, you can develop a standard set of 20 inventory categories that follow a set of naming conventions that can be used across projects. Use these frameworks across teams and with all your Oracle EBS development tools to create in-house standards that save time.

11 Reduce Manual, Repeatable Tasks with Idempotent Database Scripts

EBS implementations are composed of large volumes of database objects and PL/SQL programs. Like any EBS object, these definitions must be synchronized across many environments in your topology. Often, these objects are migrated across environments by the DBA by executing DDL/SQL scripts provided by the development team.

One of the biggest challenges in automating the process is managing updates to existing Oracle database objects. This is pretty simple for objects which have CREATE OR REPLACE objects, but that is not the case for many of the object types (e.g. tables, indexes, sequences, etc.). For example, you cannot execute a create table statement if the table already exists. This is most often managed by writing scripts that are dependent on the state of the object in the target environment. When it comes to automating the process, this becomes a challenge.

One technique is to create all DDL/SQL scripts to be idempotent, yielding the same result even when applied several times. If you follow this practice, you can maintain a single script that is used to apply changes across all environments, track the revision of the script across environments, and foster healthier relationships between developers and DBAs.

There are different techniques used, which vary by object type. In most cases, however, the scripts must check the current state and adopt its behavior. For staging tables or stateless objects, simply dropping and re-creating the object is the best method (taking care not to fail if the object does not exist). For stateful objects, such as permanent tables and sequences, you must handle insert/update conditions, all while preserving any state.

12 Avoid Hard Coding

Hard code or configure? The choice seems obvious, but it bears repeating: always configure to avoid complications and additional work down the road.

In times when you feel inclined to assign values to your code – either strings or numeric – consider where else you can get this data. If it's not available elsewhere, it is best to take the time to developing custom profile options, descriptive flex fields (DFF) or key flex fields to more effectively itemize categories.

13 Automate and Group Tasks Together

Most organizations have a few environments for various purposes like development, system testing, performance testing, production, etc. For successful delivery of features to production, it is essential that development teams adopt automation practices.

Automation can be employed to do various mundane tasks like configuration and code deployment, quality testing, and more. Automation should help avoid errors and force consistency, which means a reduction in downtime and outages. In cases where tasks need to be done during late night or weekend hours, automation and scheduling can also help improve work-life balance.

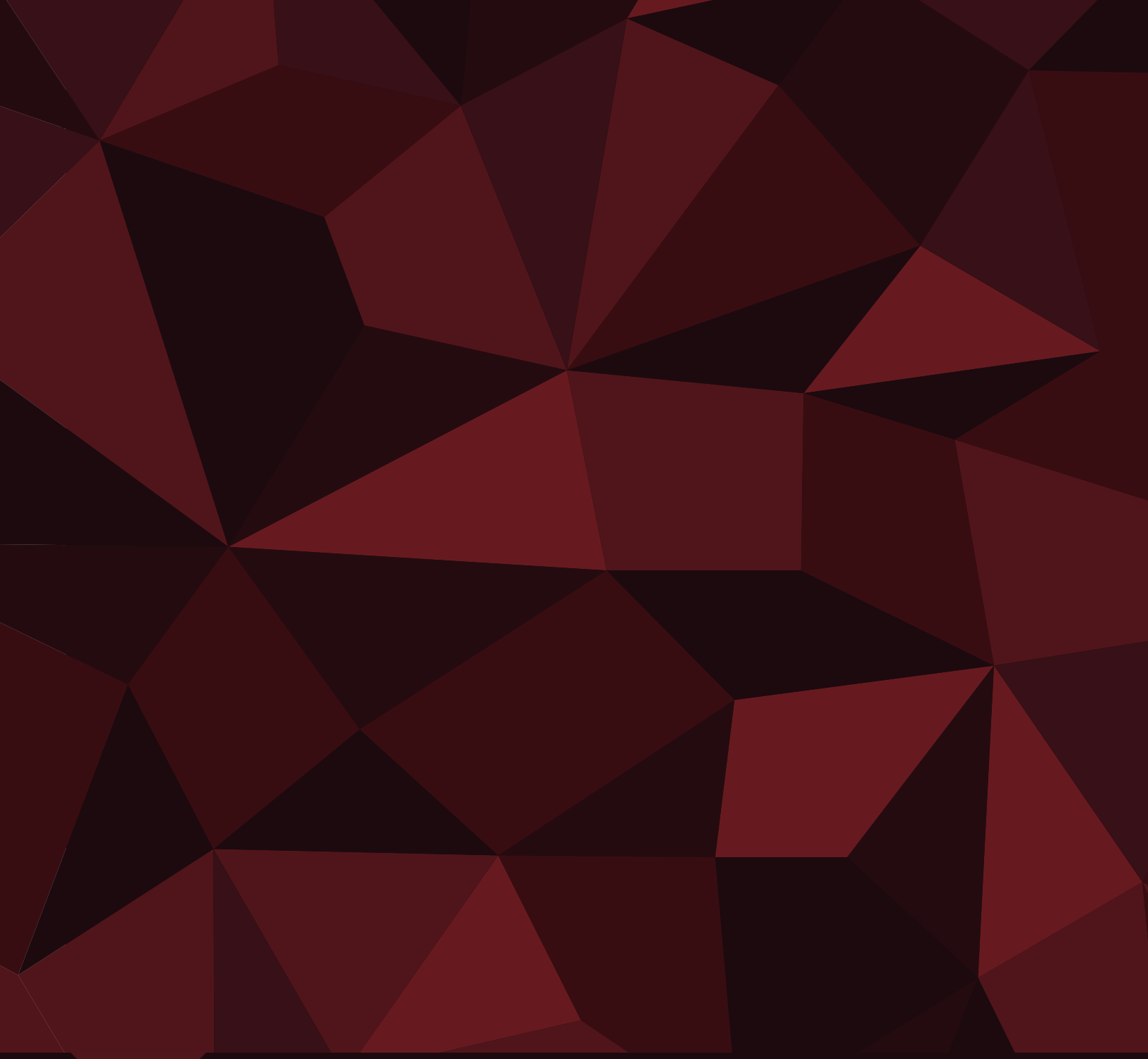
As organizations adopt automation, it is good idea to provide tools for visibility into automated changes to diagnose problems or for audit tracking for various compliance processes. Tools that make it easy to group changes together for promotion may also help you manage your EBS software delivery lifecycle that spans anywhere from a few days to weeks or months.

Use Standards and Automation to Improve EBS App Development

Oracle products are highly specialized, and IT teams working on them typically develop their own in-house processes based on guidelines from Oracle and experience with E-Business Suite. Framing that into standards and automating processes to reduce manual, repeatable tasks and cut down on errors can greatly improve app development.

> **DevOps tools often provide pipelines for managing how changes are propagated across environments in your infrastructure, including capabilities for code reviews, approvals, scheduling, and more.**

To supplement your in-house processes and best practices, you can also consider adding DevOps tools with application release automation and other functionality to your IT toolset. These tools will further alleviate the burden of manual work on your team, helping streamline processes and mainstreaming the best practices that are already in your processes.



DEVOPS FOR ORACLE APPS

13 E-Business Suite Development Tips

FLEXAGON