White Paper

Ensuring Deployment Success by Validating Releases with Evolven

Incidents still regularly occur in production following the transition of changes to production environments, and many incidents are related to the configuration, demanding that environments be validated following changes, to identify configuration consistency issues.

Evolven’s release validation capabilities compares and verifies what you had already checked and changed, discovering differences and apply advanced analytics to help IT teams to zoom in on critical differences. Evolven also identifies any manual changes made around automated deployments, so you can finally close the change and release processes loop.

EVOLVEN

TABLE OF CONTENTS

1. CHALLENGE: Any Little Mis-configuration Can Jeopardize the Success of a Release

Release validation is critical for ensuring smooth deployments. Just like in software development you need to validate the code for the program to work. So, the same is true with deployments, since an automated deployment may not detect missing configuration changes.

IT organizations regularly transition changes to Production environments, checking changes throughout a set of Pre-production environments that can include system Test, Performance test, UAT, Staging etc. (Changes are also mirrored in a Disaster Recovery environment). IT operations managers expect (or hope) to see that changes are consistently and accurately transitioned between these environments, where the resulting application files, underlying software infrastructure components and all the configuration settings match across the environments. However, as incidents still regularly occur in Production following such changes, and many incidents are related to the configuration not being synched with the live environment, environments must be validated following changes, to identify configuration consistency issues.

INCIDENTS

Incidents still occur regularly in production following changes and many incidents are related to the configuration not being synched with the live environment.

COMMON CHANGE SCENARIOS IN OPERATIONS

To understand this situation better, this document will start by exploring a number of different common scenarios where changes play a central role, affecting stability.

- Change doesn’t reach Production
- Restore from disaster recovery fails
- Missing changes during deployments
- Automated deployment script with an inaccurate parameter creates an incorrect configuration

CHANGE DOESN'T REACH PRODUCTION

During an upgrade, several problems are detected in the acceptance environment. The development team works around the clock to fix the problem so that the upgrade can still be rolled out into Production on time. While applying fixes directly to the acceptance environment, a developer forgets to document one of these changes. That specific change ends up not reaching Production. Subsequently, the application fails in Production, leaving the team baffled as to the cause, since the upgrade had worked properly in the acceptance environment, prior to transition to Production.

RESTORE FROM DISASTER RECOVERY FAILS

A problem in Production forces an organization to turn to their Disaster Recovery (DR) environment for backup. The organization does not use real time or near real time replication of the Production environment, but IT made sure to keep deploying versions to the DR environment. Nevertheless, the main application fails to start. Numerous differences exist between the actual DR environment and Production, since the most recent changes had not yet been deployed to DR. Now the operations team has to manually check a huge amount of parameters to identify the differences between DR and Production. Without a clear indication of significance of these differences, they basically have to guess which difference is the cause of the problem.

MISSING CHANGES DURING DEPLOYMENT

Every deployment can contain changes, affecting everything from a single parameter to hundreds of parameters, across different system components. For various technical reasons, deployments can miss some of these changes, like:

- A WAS manager fails to synchronize files on a specific server because of communication problems
- A deployment tool or a manual update fails to update a file because a handle is left open on that file
- A server was forced to restart before changes took effect

AUTOMATED DEPLOYMENT SCRIPT WITH AN INACCURATE PARAMETER CREATES AN INCORRECT CONFIGURATION

Automated deployment scripts are parameterized to account for differences between Test and Production environments. Scripts are updated and the change package is then rolled out in a Test environment, for Testing. However with a mistake in parameterization, the same scripts executed in Production result in one of the configuration parameters being set incorrectly, impacting system performance. Since the operations team needs to trace the issue down to a particular change in the deployment scripts, IT operations spends long hours to try to understand performance was affected.

SUMMARY: DEPLOYMENT SUCCESS REQUIRES RELEASE VALIDATION

There are many reasons for deployment errors and inaccuracies, but they all have a few things in common. These errors are hard to detect due to the large amount of changing parameters occurring at a high pace of change. Even if all of the changes could be listed, finding the ones that weren't deployed successfully would take too just long to be effective.

Releases are a part of daily IT operations, especially as more and more organizations adopt agile approaches and work in continuous delivery modes. The integrity of releases faces many threats:

VALIDATION

IT operations needs to ensure that the release is intelligently validated.

- Failed deployment of changes
- Unauthorized or undocumented changes
- Inherent differences between the environments
- Lack of holistic view of changes to the system
- Difficult to understand the impact of mis-configurations

To handle these challenges and the amount of changes carried out across what amounts to overwhelming numbers of configuration attributes, application files, database schemas and other environment resources, IT operations needs to ensure that the release is intelligently validated. This means that all the changes are captured and analyzed for accuracy, completeness and legitimacy across the release life-cycle, from Testing to Production, as well as DR.

2. SOLUTION: Automatically Validate Releases for Accurate Deployment

IT operations needs to improve the integrity of releases by validating releases over the application lifecycle. The release validation process should cover:

- Inherent differences between the environments
- Deviation from the expected deployment results in the actual deployed change

To reliably close the loop in the release process, both issues can be prevented by incorporating Evolven in the following steps:

2.1. Compare baseline environments

2.2. Fix harmful differences

2.3. Deploy

2.4. Check consistency of the deployment across similar servers in the target environment and consistency between the target environment and its predecessor from the lifecycle point of view, for example Production vs Acceptance

2.5. Analyze the results to identify inconsistencies that need to be remediated

2.6. Review release bill-of-material for risky changes

While steps 2.1 and 2.2 are carried out prior to the deployment as a pre-requisite check, consistency between environments needs to be continuously evaluated as a regular health check to ensure that the Pre-production environment sufficiently represents Production, and that Production is aligned with DR etc.

2.1 COMPARE BASELINE ENVIRONMENTS

For environments with long histories, natural drift occurs, with consistency diverging between them. By comparing baseline environments, IT can enhance their ability to identify risk introduced by environmental drift.

This step shows where consistency in environments can easily fall out of control for IT.

Example: Update Misses a Server
There are 3 Production servers running and handling the load for a large numbers of users. Originally the environment contained only two Production servers but due to an increase in the amount of users, a third server was added. The new server was created from a VM image and was assumed to be identical to the 2 existing servers. However, the existing servers were changed, having OS updates installed a few weeks prior to the creation of the third server. Those updates were not deployed to the stored images. This means that the third server has drifted from the configuration of the other servers, lacking OS updates, which might be required for future deployments.

Example: No Visibility to Critical Differences Between Environments
The Test and Production environments were setup long ago.

When the Test environment was built, the max memory setting was set to a lower value than the max memory setting value in Production and that worked fine. In time, the personnel who set up the initial environment went on to other projects, leaving the current team that is working on these environments unaware of the difference in max memory settings.

With the release of the new, more robust version of the application, more memory is consumed when working in Production. For the first time the application is actually consuming the max memory that was defined. Since this parameter was restricted in the Test environment, the application worked well, however in Production, the increased memory consumption prevents other processes from successfully executing, degrading performance.

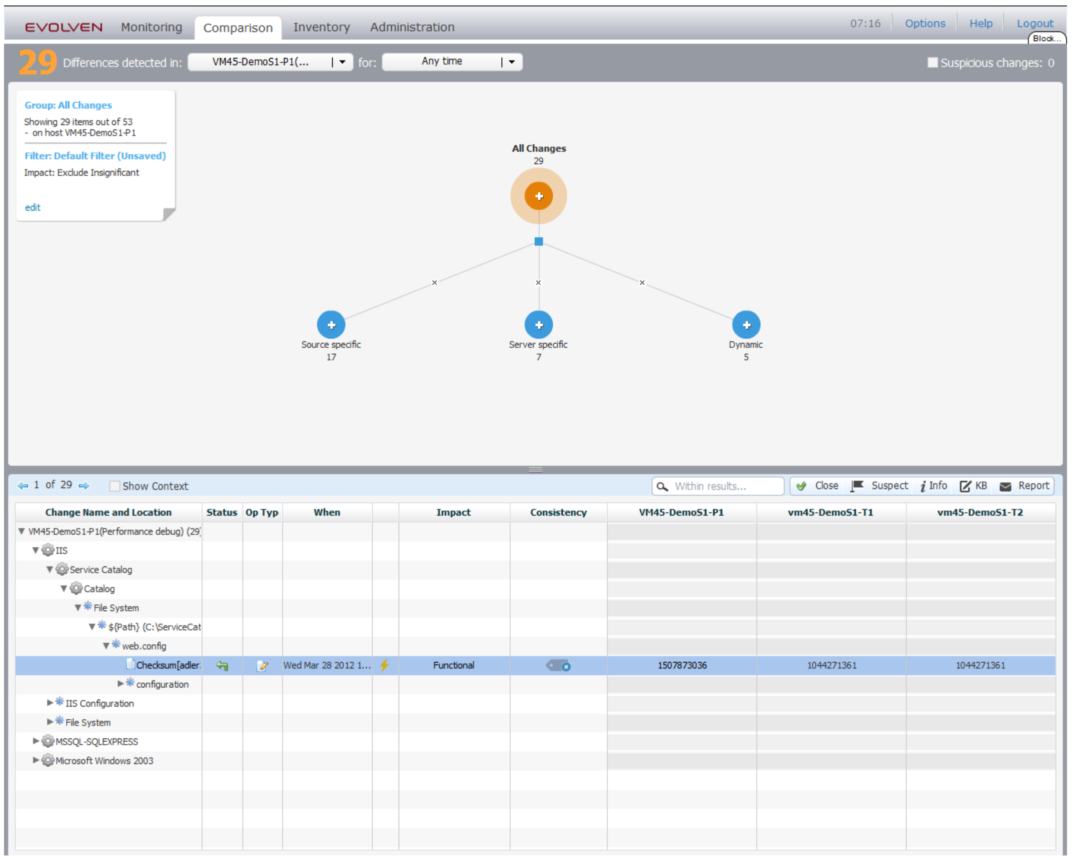
These examples show how historical differences could be created and what their impact to the environment may be.

After installing the Evolven agents, IT operations can capture an accurate snapshot of the current environment configuration. IT Operations can then run comparisons between different sets of environments to proactively prevent drift:

- Compare Production servers to each other
- Compare Acceptance servers to each other
- Compare between Acceptance and Production
- Use the same approach for any other Pre-production and Test environments

COMPARE BASELINE

By comparing baseline environments, IT can enhance their ability identify risk introduced environmental drift.



29 Differences detected in: VM45-DemoS1-P1(... for: Any time

Suspicious changes: 0

Group: All Changes
Showing 29 items out of 53
- on host VM45-DemoS1-P1

Filter: Default Filter (Unsaved)
Impact: Exclude Insignificant

edit

1 of 29 Show Context Within results...

Change Name and Location	Status	Op Typ	When		Impact	Consistency	VM45-DemoS1-P1	vm45-DemoS1-T1	vm45-DemoS1-T2
▼ VM45-DemoS1-P1(Performance debug) (29)									
▼ IIS									
▼ Service Catalog									
▼ Catalog									
▼ File System									
▼ ${Path} (C:\ServiceCat									
▼ web.config									
Checksum[adler			Wed Mar 28 2012 1...	⚡	Functional		1507873036	1044271361	1044271361
▶ configuration									
▶ IIS Configuration									
▶ File System									
▶ MSSQL-SQLEXPRESS									
▶ Microsoft Windows 2003									

This shot shows a comparison of a production server to 2 test servers. The time filter is set to "any time" in order to include historical differences.

2.2 FIX HARMFUL DIFFERENCES

Running the comparisons described above can provide IT operations with a full inventory of differences. These results can be passed on to various operations specialists for review. The breakdown of differences by the type of environments allows IT managers to create separate sets of differences for each specialist team (DBAs, system administrators, application server administrators etc.). They can then set which differences are expected and which need to be fixed.

This results in three sets of data differences:

DIFFERENCES

Create separate sets of differences for each specialist team

- **Differences to be fixed.**
  Prioritized (assisted by Evolven's knowledgebase). There should be a plan and action items for fixing these differences.
- **Expected differences.**
  Marked by Evolven as expected differences so that they do not appear in future comparisons between the environments
- **Differences to be ignored.**
  Updated in Evolven's knowledgebase so that when Evolven analyzes, it will consider them as less significant in the next environment review

Note: The "Analyzing the results" step will explore this further.

It's important to emphasize that the first time the analysis is done the amount of differences that appear will be significant. However once the initial assessment is completed, IT can proceed with review of the new differences only added in addition to the differences that have already been reviewed, i.e. only review consistency of new environment deltas.

Some of Evolven's customers use the following approach to minimize any operational overhead:

> Accept existing differences between environments as legitimate (if they did not create any historical problems), from there, start analyzing consistency of all new differences introduced within and across environments

Now IT can be sure of the similarity of environments and that they have documented everything that is still different.

Process Integration for Fixing Harmful Differences

The alignment of environments can be done at any time. However it is recommended as the first step in Evolven integration into operational processes .

Note: While it is not a requirement for the next steps, there is added value in thoroughly carrying out this process first, as the lessons learned here can also be used for the next steps.

- The differences should be reviewed by members with the right expertise. For example database related differences should be reviewed by a DBA, WAS differences by the WAS admin, etc. In addition the person in charge of the environment (i.e.: the environment manager or a DevOps team lead) should make sure all changes that were reviewed
- When adding a new server to the environment, it should be compared to existing servers. Before going into Production, the engineer responsible for the environments needs to compare the new server to existing servers. This will ensure that both Infrastructure and applications are aligned.

2.3 DEPLOY

With the environments aligned, the environments are ready for deployment.

Since Evolven agents would have been deployed already, they automatically collect all of the changes.

- **Integrate to Deployment Script**
  When using an automated deployment tool, Evolven can be integrated into the deployment script. This entails adding an Evolven scan and report as the last step of the deployment.

- **Run Deployment Report**
  When not using automation, Evolven can either be scheduled to run a report based on the time of deployment or the results can be manually analyzed from the Evolven UI, as explained in more details in "Set Up Analysis Plans For the Different Environments"

2.4 CHECK CONSISTENCY

The consistency report tracks all of the changes made to the environment as part of a deployment, so that IT can:

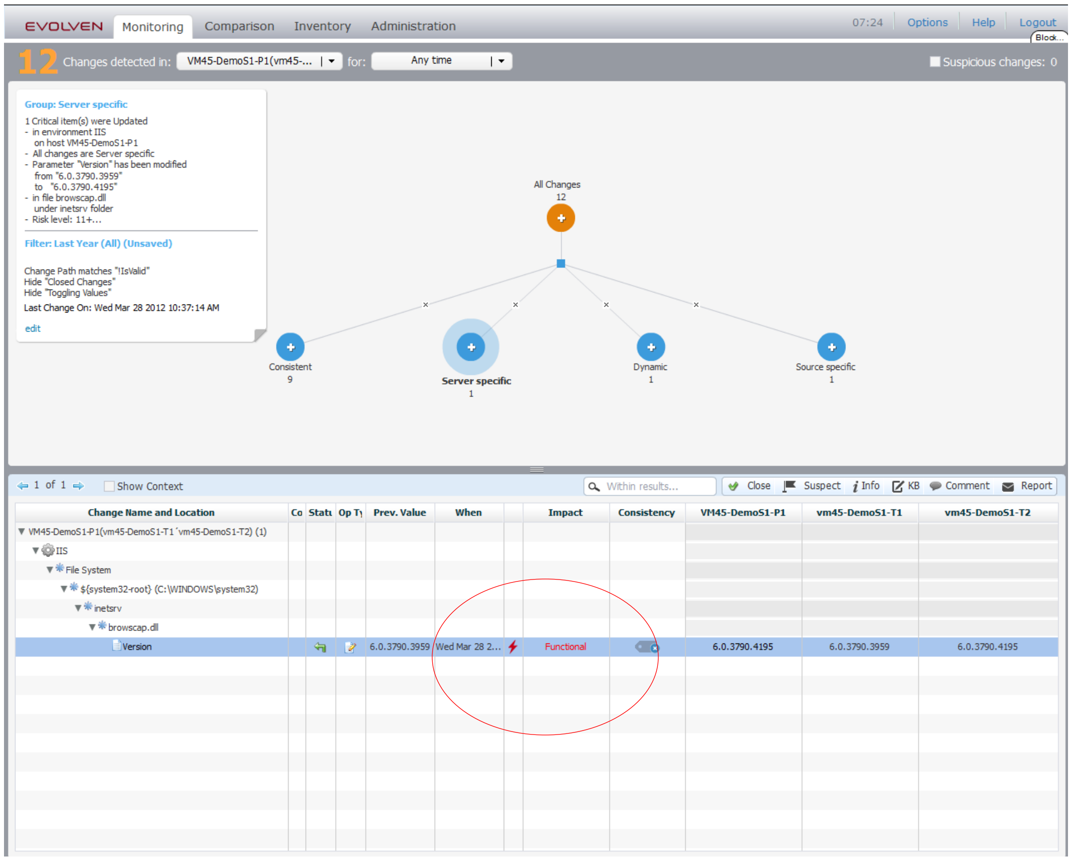
- Audit the changes made as part of the deployment
- Make sure that the changes were consistent

The audit trail is useful for the operations specialists. Each of them can review the changes done in their components (App servers, databases, applications) to assure only desired changes took place.

In addition, the release manager can check for consistency. There are two types of consistency checks that can be carried out:

- Consistency between two environments
- Consistency within an environment

Since the Evolven agents track all changes, the first thing the report will show is all of the parameters that were changed, what their old value was and what their new value is.



Group: Server specific

1 Critical item(s) were Updated
- in environment IIS
  on host VM45-DemoS1-P1
- All changes are Server specific
- Parameter "Version" has been modified
  from "6.0.3790.3959"
  to "6.0.3790.4195"
- in file browscap.dll
  under inetsrv folder
- Risk level: 11+...

Filter: Last Year (All) (Unsaved)

Change Path matches "!IsValid"
Hide "Closed Changes"
Hide "Toggling Values"
Last Change On: Wed Mar 28 2012 10:37:14 AM

edit

Change Name and Location	Co	Statu	Op Ty	Prev. Value	When		Impact	Consistency	VM45-DemoS1-P1	vm45-DemoS1-T1	vm45-DemoS1-T2
▼ VM45-DemoS1-P1(vm45-DemoS1-T1 ´vm45-DemoS1-T2) (1)											
▼ IIS											
▼ File System											
▼ ${system32-root} (C:\WINDOWS\system32)											
▼ inetsrv											
▼ browscap.dll											
Version				6.0.3790.3959	Wed Mar 28 2...		Functional		6.0.3790.4195	6.0.3790.3959	6.0.3790.4195

The shot shows the changes made to a system and were checked against the testing environment. Note that one DLL file was not updated.

Example: Old Value Remains in Configuration

Complaints have been received from specific users who suffer from sporadic problems. Though the reports are very detailed, the team can't recreate those problems. Such problems are usually caused by a deployment that left an old value on one of the servers under the load balancer. Only users who are directed to the problematic server experience these issues, and so the issues are extremely difficult to solve.

Set Up Analysis Plans For the Different Environments.

Analysis plans include instructions for which environments are included in the consistency check. So for example, when deploying from the Acceptance environment to Production an analysis plan will be setup that checks consistency across the Production servers and a consistency check of Production with the Acceptance environment.

- **Automated Deployment Approach.**
  The deployment script needs to be added to the initiation of the consistency checks. This will provide a fully automated, self-validating deployment process. Through a Web Services API, the script will initiate a consistency check that will scan the changes made to the servers, check for consistency according to the analysis plan and either email a report or save it to a local file system
- **Manual Deployment.**
  The engineers who do the actual deployment must know that the last stage of deployment is to run a consistency check. They will be able to log into the application and:
  - Run a scan to refresh the current configuration
  - Run consistency checks that will determine which of the change are consistent and which aren't. The results can be viewed with Evolven's UI or exported into a report

For both steps, a report is produced including all the changes that happened to the system. While the operation specialists can review the entire list of changes to make sure all of the desired changes took effect in their area of expertise, the release manager will focus on reviewing consistency. Reviewing the consistency results should be done using a filter to exclude the following parameters:

- Dynamic
- Installation specific
- Expected differences

Note: In addition, it is recommended to schedule daily consistency reports to be inspected by DevOps or operational teams in search of inconsistent changes that were made to the system outside of the regular release process. As explained in more details in "Set Up Analysis Plans For the Different Environments"

2.5 FIX

The results of the consistency check can be divided as:

- **Consistent changes**
  These are changes that were deployed in the same manner to all servers. With the goal to have all changes fall into this group
- **Inconsistent changes**
  This is the suspects list. The changes here have different values on different servers, when they shouldn't.

The inconsistent changes should be reviewed by the deployment manager. The consistency report should show that changes were consistent both within the environment and between the different environments. However, especially with consistency between environments there might be a few changes that are inconsistent. Just as was seen with comparing baselines, the results should be able to be split up as:

- Differences to be fixed
- Expected differences
- Differences to be ignored

CONSISTENCY

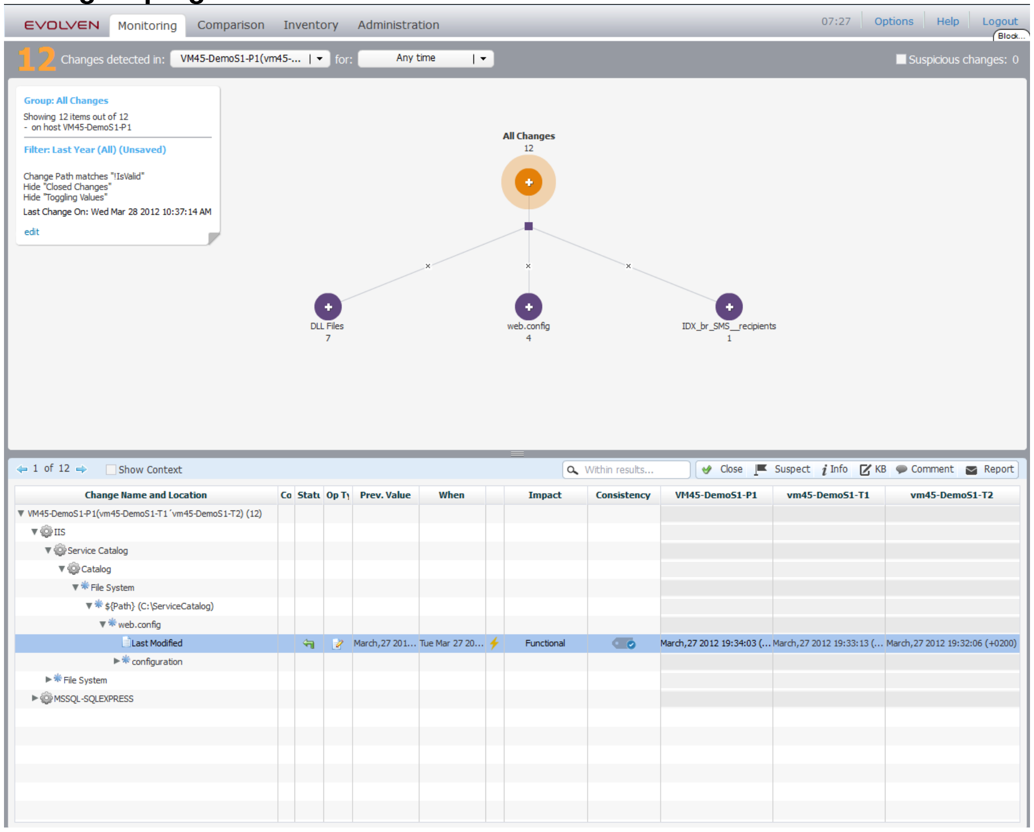
The consistency report should show that changes were consistent both within the environment and between the different environments.

2.6 ANALYZING THE RESULTS

Analyzing a set of results is not always an easy task. There are a few rules of thumb that can be used to reduce the time it takes to go through the list and help zoom in on critical differences.

- **Critical parameters**.
  With Evolven's built-in knowledge base, parameters are categorizes . The search can begin with reviewing the parameters that are marked as critical.

- **Auto grouping.**



12 Changes detected in: VM45-DemoS1-P1(vm45-... for: Any time

Suspicious changes: 0

Group: All Changes
Showing 12 items out of 12
- on host VM45-DemoS1-P1

Filter: Last Year (All) (Unsaved)

Change Path matches "!IsValid"
Hide "Closed Changes"
Hide "Toggling Values"
Last Change On: Wed Mar 28 2012 10:37:14 AM
edit

1 of 12 Show Context

Change Name and Location	Co	Statu	Op Ty	Prev. Value	When		Impact	Consistency	VM45-DemoS1-P1	vm45-DemoS1-T1	vm45-DemoS1-T2
VM45-DemoS1-P1(vm45-DemoS1-T1´vm45-DemoS1-T2) (12)											
IIS											
Service Catalog											
Catalog											
File System											
${Path} (C:\ServiceCatalog)											
web.config											
Last Modified				March,27 201...	Tue Mar 27 20...		Functional		March,27 2012 19:34:03 (...	March,27 2012 19:33:13 (...	March,27 2012 19:32:06 (+0200)
configuration											
File System											
MSSQL-SQLEXPRESS											

The auto grouping is a powerful tool that helps to quickly go through multiple objects, and will divide a list of changes and differences into groups by their similarity. The search is focused on the smaller groups as it is parameters that standout that have the highest potential to be harmful. Auto grouping is used recursively, splitting the large groups into smaller ones, focusing specifically on the group titled "more" - the group that contains parameters which stand out and had no similar changes.

- **Keep updating the expected differences list.**
  When reviewing the results, inconsistencies need to be marked as expected differences when encountered, making for a cleaner report each time the consistency check is run.

- **Mark Inconsistent changes while reviewing.**
  Expected differences are differences that contain values such as local IPs. Machine names, domains, etc. These are referred to as expected differences because they are different by design. Moreover, when marking differences as expected, this indicates to Evolven that the operator should

be notified if these parameters are overrun with identical values.

Handling Overriding Configurations.

Marking parameters as expected will produce a warning, if it is overrun with the same value. Once all the expected differences are marked as such, only changes that should be fixed remain. Ideally, the administrator would like to fix all of the differences that are left but in practice some differences can be skipped and the description of these changes can be set to “ignore”, so they can be reviewed separately and filtered from the current view.

Example: Overriding Configuration

While migrating a new version from Acceptance to Production, the deployment script overrides the value of the pointer to the database and the Production environment now points to the Acceptance database. These parameters are different by design.

3. INTEGRATION into the IT Organization

To successfully close the release loop, integrating Evolven IT Operations Analytics into existing day to day processes is crucial. This tool can help improve the effectiveness of a variety of key roles in the operations team for keeping releases on track, and maintaining system performance.

Here's a review of typical role integrations:

- **Release Manager.**
  As the release manager's primary goal is focused on successfully releasing all changes to their target environments. This tool can provide the release manager with the consistency reports, indicating both successful deployment and which parameters need fixing.
- **Operations Specialists.**
  The operations specialists receive baseline reports. These provide an opportunity for, reviewing inherent differences between environments. Then when setting up for deployment, they can decide which differences should be fixed and which are expected.

  In addition, the specialists can review and validate the content of deployments to make sure deployments contained the changes that were intended and that changes were not overlooked.
- **DevOps Lead.**
  The DevOps lead validates that consistency is maintained for every deployment. By scheduling automated consistency checks with this tool, especially within Production, environments can be validated that their configurations are not drifting apart. Overall the DevOps lead should own the release validation effort, ensuring that the release manager and operations specialists review results from Evolven's analysis and that no outstanding differences need to be fixed.

4. BENEFITS

Evolven’s release validation capabilities compares and verifies what you had already checked and changed, identifying any changes that were not implemented by automated deployment tools, so you can finally close the change and release processes loop

Evolven is an effective tool for safeguarding environments. Collecting detailed configuration information from everything from applications to hardware across environments, Evolven identifies differences and applies advanced analytics to help IT teams to zoom in on critical differences.

Any little mis-configuration of a single parameter can possibly instigate a high impact incident, putting releases into long, drawn out stabilization periods, that even result in Production outages. Evolven provides the detailed visibility of what is being released down to the most granular level of the configuration parameter; validating that the integrity of the live environment is protected and that execution adheres to the release plan.

EFFECTIVE

Evolven identifies differences and applies advanced analytics to help IT teams to zoom in on critical differences.

With Evolven, IT teams can:

- Validate the changes in application deployments and software deployments, and know that they were implemented correctly, avoiding downtime.
- Monitor and proactively detect any unauthorized changes and configuration drift, preventing downtime.
- Investigate incidents by analyzing if the changes or re-configurations are the root cause of the incidents - cutting mean-time-to- resolution (MTTR).
- IT operations working with Evolven as described here can achieve greater system stability , regardless of the amount of changes, spending less time on time-consuming deployment stabilization processes (and thus a faster time-to-market), and more time on support of business needs.

About Evolven

CORPORATE HEADQUARTERS
2500 Plaza 5, 25th floor,
Harborside Financial Center
Jersey City, NJ 07311
Email: info@evolven.com.
Tel: 1-888-841-5578
UK: +44 (0) 20-3002-3885

R&D CENTER
16 Ha'Malacha St.
Rosh Ha'Ayin, 48091 Israel
Email: info@evolven.com
Tel: +972-77-777-5999
Fax: +972-77-777-5900

Evolven's IT Operations Analytics provides intelligent answers to key IT operations challenges: how to accelerate incident resolution, how to avoid harmful and risky changes, and how to assess and optimize IT operations performance.

Evolven's new analytics approach to the chronic change & configuration challenges dramatically minimizes the risk of downtime and slashes incident investigation time.

Leading industry analyst, Gartner selected Evolven as a 2013 Cool Vendor in IT Operations Management recognizing Evolven as "the only vendor to marry IT Operations Analytics to configuration and change management". In 2013, Evolven was selected as a winner of the Red Herring Top 100 North America award, a prestigious honor that recognizes the year's most promising private technology companies across North America. Adding to this recognition, other industry analysts have recognized Evolven for "transforming change and configuration management" and as the "Industry's most adaptive change management analytics."

Evolven is a privately held company headquartered in the U.S. and has a development center in Israel. Evolven's executive team and advisory board include world-renowned experts from the world of enterprise software. Evolven is backed by leading venture capital firms: Pitango (www.pitango.com) and Index Ventures (www.indexventures.com).

See more about Evolven at www.evolven.com.