White Paper

Cutting Incident Investigation Time with Evolven

In today's complex IT environments, it doesn't take much to cause a high impact incident. Any minute misconfiguration or omission of a single configuration parameter can quickly lead to an incident with high impact: reputation damage, dissatisfied customers, financial losses, legal liabilities, and full re-organization. As IT incident management teams transform into 'firefighters', running against time to stabilize high-priority crises, productivity drops drastically.

Evolven's IT Operations Analytics capabilities let you investigate environment incidents and quickly identify configuration changes and differences that can be the incident's root-cause.

EVOLVEN

TABLE OF CONTENTS

1. PROBLEM

Any organization deals with high priority incidents on a regular basis. This can directly and indirectly impact an organization's revenues as well as productivity, as IT operations are kept busy firefighting incidents rather than working on new projects that support critical business requirements.

One of the first questions asked when investigating incidents is 'what changed?' And the answer to this question is difficult to produce at the right level of details. There are many reasons why it's so hard to answer this question when under pressure. Among the reasons are:

INCIDENTS

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

- The complexity of environments
- Low visibility into the content of each environment
- Rapid pace of change
- Silo-based management of environments
- Difficulty to separate harmful changes from desired changes

To address this challenge, most IT organizations assemble "war rooms" to manage critical incidents. Virtual teams meet, comprising key members from development, production support and other IT operations groups. Using e-mail and phone, members of the team collect and share information to identify the root cause of a critical incident and remediate it. This approach to incident management takes too long to be effective, and is costly. A new approach is needed to automatically gather all the required data, make sense out of it and deliver clear insights to IT teams that can act upon them to quickly resolve incidents.

1.1 COMMON INCIDENT INVESTIGATION SCENARIOS IN OPERATIONS

Example: Forgotten Debug Switch
For a couple of days, a production environment suffered from performance issues. The only clue the team had was that a few days prior, they suffered another incident that was resolved. After a long investigation process, the team understood that as part of the previous investigation effort, one of application developers changed the logging level to try and understand the root cause of the incident. Working under stress to get the environment up and running, the debug switch was left turned on, causing multiple logging writes to the hard drive, and slowing the environment down.

Example: DLL Overlooked in Upgrade
Following a recent update, users complained about unstable application behavior, complaining about a specific action that the application does. However users also said that after logging out and back in, they were able to perform the action. Trying to recreate the problem, the QA team did not succeed. Digging into the application's configuration and code, the team discovered that on one of the servers in the production environment, comprised of several servers, under a load balancer a DLL was not overwritten in the last upgrade. As a result, users experienced a problem, directed to the faulty server and trying to use the feature for that DLL.

Example: Unsynchronized Servers
A new version of an application was deployed on a clustered IBM WebSphere application server, on the deployment manager node. The deployment manager then deploys to different clusters. A network error caused one of the clusters to remain unsynchronized. The issue is not discovered and users directed to the faulty node keep getting errors.

Example: Undocumented Change Crashes Environment
Multiple teams deploy updates to the production environment, one team is an external vendor. The vendor provides a new patch for the environment. The documentation of the patch states that the patch only includes changes to the vendor's application. After installation, the production support team discovers that the patch caused the environment to crash. After quickly uninstalling the patch, the environment remains down.

Further investigation reveals that the patch actually made undocumented changes to the OS, causing the environment to fail.

Example: Automated Script Goes Wrong
A base image is used to create new servers. The image is then updated using automated scripts, bringing it up to the latest configuration version. The infra team changed a script that configures the .Net version. After the change, when loading the application, the application does not work because the script sets up the wrong .Net version.

Example: Critical Hardware Differences Not Checked
When system performance suffers under a big load, another server is added under the load balancer. However the new server does not create the reports the system is meant to create. Through a comparison of the new server with a working server, differences are revealed in the graphical card. The report generation software only relays on a specific type of graphical card, and the problem is solved by switching to the correct card.

Example: Independent Update Stops Application
The network team carries out another update, updating the firewall policies. They make the update without talking to the application team. This update blocks the ability of critical components in the application to communicate with each other, causing the application to stop working.

Example: Human Error Releases Untested Change to Production
An engineer intended to implement a planned change to the Acceptance testing environment. However, due to human error, the change is applied to the Production environment instead. This results in the change being deployed while still untested, causing stability problems to the production systems.

Example: Automated Script Deletes Database Index
A performance issue arises in the database. It turns out that the schema was changed and a new index was added, however an automated script that re-indexes the database deleted the new index and created an older one. Without the new index the system's performance dramatically deteriorates.

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.

2. SOLUTION

2.1 OVERVIEW

To resolve incidents, IT teams need to understand if there were changes to the system that could be the cause of the incident. According to estimates of numerous industry experts, 40% to 60% of stability issues are caused by changes. These changes could occur anywhere in the environment stack, starting at the hardware layer and go all the way up to changes in the application code.

This demands answers to the following questions:

- Which parameters were changed recently?
- Which of the parameters became inconsistent as a result of changes?
- Are there any parameters that deviate from the setup of systems that keep running seamlessly?
- What changes were made to the bill-of-materials of the environment (files, database schemas, registry keys and other items comprising the environment)

When answering these questions, the amount of data produced could be significant. So it is essential to narrow down the results to the relevant information linked to a cause of the investigated incident.

The first step should be to make sure to collect the right information. To learn more about configuration collection, refer to “APPENDIX 1 - COLLECTION CONTENT”.

The next step is to identify granular information that is related to the current incident.

Evolven delivers three main sources of environment information that is analyzed, zooming down into the root cause of the investigated incident:

- Changes that occurred since the system was working fine
- Comparison of a working instance of the environment such as a test environment with the non-working environment
- Comparison to a “golden baseline” of the investigated environment

Whatever method of analysis is used, Evolven can scan the environment where the incident occurred, to make the most recent changes in the investigation are included.

2.2 3 MAIN SOURCES OF ENVIRONMENT INFORMATION

2.2.1 Recent changes

Evolven agents have the ability to scan environments at any frequency defined by a user looking for changes. Evolven best practices suggest having the agents scan the entire system once an hour. This gives sufficient granularity of changes with negligible overhead to the system. For systems with a lower or higher pace of change, the frequency can be set to a lower or higher value respectively.

In Evolven's Monitoring tab, a time filter can be set to include only those changes that occurred prior to the point when an incident was reported. To choose a reasonable time period, one need ask, when was the environment working well at the last checked time? Was it immediately after the deployment? Was it this morning? The more frequent an agent scans, the more granular can users get with regard to the analyzed time periods.

It is important to consider that some changes impact environments long after deployment. The investigation process can review changes iteratively expanding the range from several hours to several days.

2.2.2 Comparison to a working system

Most organizations have multiple environments supporting business systems lifecycle (pre-production, staging, production etc.) Looking across these environments one can find those that should be similar to the faulty environment. For example:

- Clustered or load balanced servers
- An acceptance testing environment that is similar to production, at least from the point of view of the logical architecture.

These servers or environments run the same business applications on similar infrastructure. Their configuration should be close enough except parameters that are unique to each environment. For example host name and IP, or are not essential for environment purposes (like a difference in the memory size between test and production servers). Evolven can normalize such parameters so that a comparison of environments will remove legitimate differences from the result set, focusing on parameters that should not be different.

2.2.3 Comparison to a "golden baseline"

What happens if there is not a working copy of the system? It is still an imperative to see how the system deviates from the intended configuration.

In such a case, it is recommended to maintain two types of golden baselines for comparing the environment:

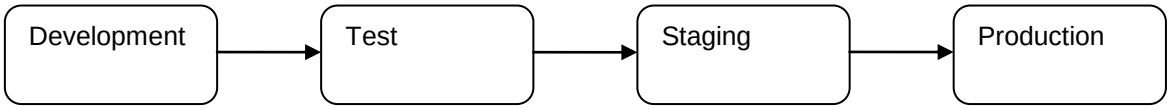
- Infrastructure golden baseline
- Application golden baseline

Infrastructure Golden Baseline

An Infrastructure golden baseline is a snapshot of an environment configuration and bill-of-material ('bookmark' in Evolven terms) covering all the required infrastructure objects such as hardware, OS, databases, messaging platforms, etc. This baseline could be used for validating the infrastructure stack of new servers that are created and should also be used for incident investigation. This comparison will validate that the provided infrastructure is up to the required standard or known working configuration and will detect if there are deviations that can be a root cause of investigated incidents.

Application Golden Baseline

An application has a lifecycle where it moves from environment to environment. In a classical flow, a new version or patch will start in the development environment and then be deployed to test, staging and production, as described here:



Capturing a snapshot of the Staging environment immediately after the deployment creates an application golden baseline that contains all of the application components in the configuration that should be present. When an incident occurs, the non-working environment is compared to this baseline to determine if there are any differences at the application level which could be the cause of the incident.

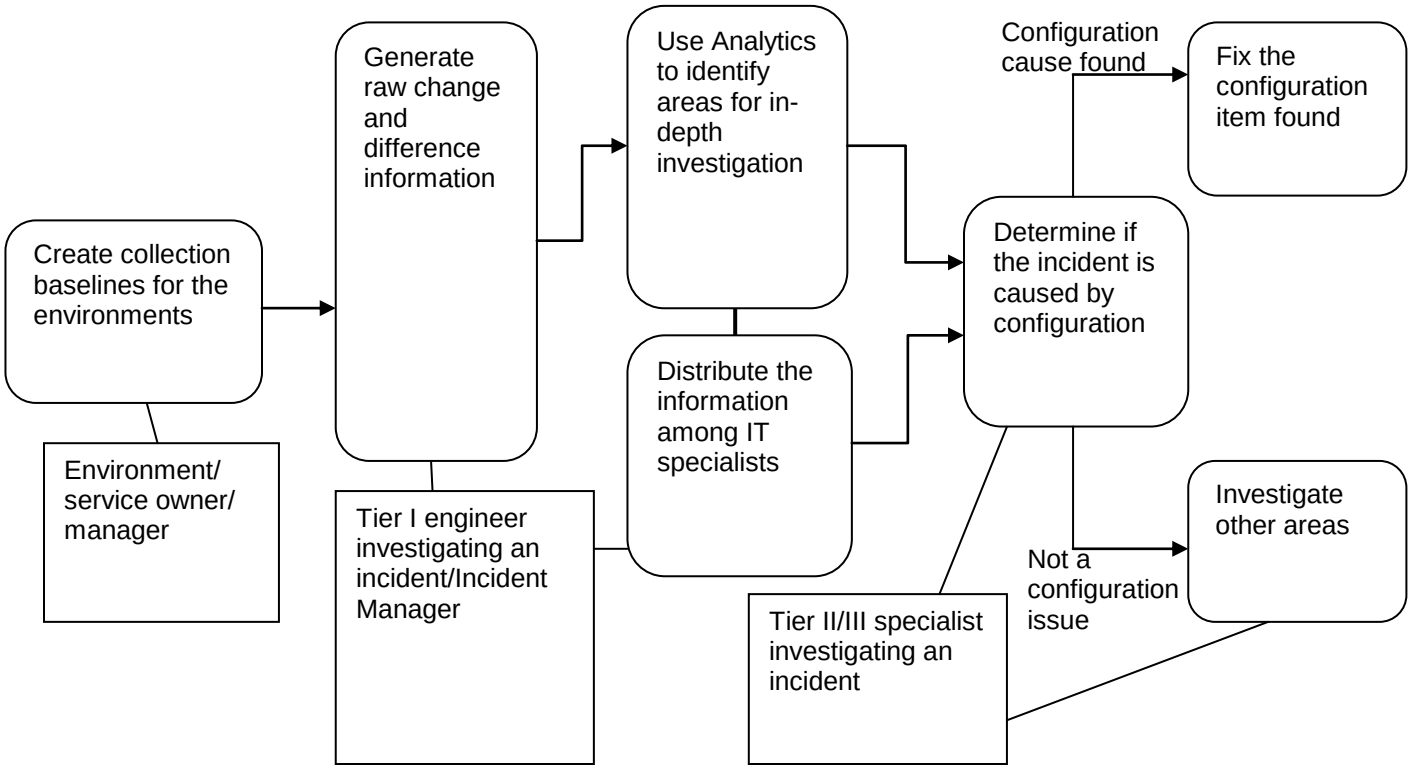
2.3 ANALYSIS

After collecting the information, Evolven's analytics can zoom into the change and difference information that put the environment at risk. Evolven analyzes and breaks down detected changes and differences using multiple data dimensions:

- **Knowledge base** – Evolven comes with a built-in knowledge base that categorizes the configuration parameters of common technologies by their criticality and possible area of impact. Evolven can automatically exclude changes to the parameters that are marked as insignificant in the knowledge base from the analysis or, on the contrary, highlight changes to the critical configuration parameters. In addition, analytics can filter and group changes based on a type of investigated incident (performance, availability, functionality, security etc.)
- **Unauthorized changes** – Changes that were reviewed and approved are less likely to be a cause of the investigated incident. Evolven breaks down authorized and unauthorized changes, while the unauthorized changes obviously get higher priority in analysis. To identify the changes that are authorized, With Evolven, time windows can be defined in which authorized changes can occur. Alternatively, Evolven correlates detected granular changes with approved change requests managed in IT change tracking systems such as, a service desk.
- **Inconsistent changes** – Evolven assumes that similar sets of changes should be deployed to similar environments (e.g. clustered servers). It analyzes changes across multiple similar environments to identify configuration parameters or bill-of-material items with values that were changed inconsistently.
- **Frequency** – Configuration parameters are evaluated by the frequency and pace with which they change. Evolven assumes that more frequent changes have lower risk than a change to a parameter that was detected for the first time
- **State** – Evolven allows users to manually mark certain changes as suspicious. These changes will get maximum priority in the analysis.
- **Environment layer (databases, middleware, application)** – Evolven can focus on changes that occur at specific layers of the application while still getting a holistic view of the changes to the system.

Evolven's analytics allow users to break down detected changes or differences for selected time periods using the dimensions listed above as well as a number of additional data properties. It also applies statistic algorithms to automatically group changes that have similar context for easier analysis.

Evolven's analytics can easily dissect large amounts of granular changes and differences, to zoom in on the configuration parameters or bill-of-material items that could be a root cause of an incident being investigated.



3. INTEGRATION INTO THE IT ORGANIZATION

3.1. MAIN IT ORGANIZATION ROLES

3.1.1 Environment / Service owner / Manager

The environment or service owner is responsible for creating a baseline of the environment. The Evolven agent collects common configuration information out of the box, and the environment manager will review the baseline and decide if an additional collection of proprietary configuration is required. The environment manager may decide to involve other IT specialists to review information in their domain of expertise. See "*APPENDIX 1 - COLLECTION CONTENT*" for more information on environment configuration collection. Once the initial collection is done Evolven automatically detects all of the changes made to the environment.

3.1.2 Tier I support

When an incident occurs, the Tier I support executes procedures in their run-book to determine the course of action. A review of the information provided by Evolven could be applied for the following:

- Determining whether the incident is a configuration incident. If no changes were made to configuration since the environment was working well, this can free up the team to investigate other possible issues
- If changes do exist, the team can look for the following:
- Are there any unauthorized changes in the list?
- Does the knowledge base point to changes related to the incident? E.g. if the incident is related to performance, are there changes marked as affecting performance?

In general, Evolven analytics can be used to determine if there are changes that should be escalated. The next step is to list changes and share them with relevant Tier II and III specialists.

3.1.3 Tier II and III IT specialist

In a "war room" scenario, different IT specialists will review the changes made to the system. The review will determine which changes could be the root cause of the incident and should be remediated and which should be accepted as legitimate changes.

3.1.4 Incident manager

The incident manager needs to be able to understand any incident at least on a basic level in order to use the appropriate resources. Additionally, it is the incident manager's responsibility to ensure that the team has enough information to start the analysis. An Evolven report provided by tier I support can be used both to update teams on changes that have happened and might be the cause of the incident, narrowing down the scope of the investigation.

A review of the changes made to the system can be helpful in determining priorities for investigation as well as identifying the resources that need to be used for the investigation.

4. BENEFITS

A key step of the incident management process is incident investigation that leads to incident resolution. Any IT organization strives to reduce incident mean-time-to-resolution to support business partner objectives.

Typically, incident investigation triggers numerous manual activities for gathering required information. IT teams involved in incident investigation need to have all the relevant information at hand as quickly as possible.

EFFECTIVE

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

Frequently, the most complicated problems are triggered by subtle changes. Any little mis-configuration of a single parameter can possibly instigate a high impact incident, putting releases into long, drawn out stabilization periods, that can even result in production outages.

Evolven collects the most granular data and analyzes it to ensure that no change will escape the attention of the applications and operations teams, letting them focus on the changes that actually matter during the critical time of an incident. Providing a single point of view on the environment state, recent and historical changes, Evolven can accelerate the investigation process.

APPENDIX 1 - COLLECTION CONTENT

Evolven offers automated configuration and bill-of-materials collections out-of-the-box including a variety of environments components across the business system stack. These components start at the hardware level traversing up through the operating system, messaging layers, databases, application servers, web servers and the applications themselves.

Evolven collects all of the relevant parameters, parsing and individually monitoring each granular parameter within the configuration files related to the discovered applications, database tables holding configuration, API based configuration sources etc.

In addition, Evolven monitors the environment's bill-of-material including non-configuration files such as executables and other environment resources. For each of these files Evolven collects as much information as possible without actually parsing them. Such information will include version (if applicable), last modified time stamp, and the file's checksum. In addition, if the application is a Java application, Evolven extracts and parses manifest files. This information provides a good basis to detect changes and differences in such files when monitored and compared.

In addition to the content that is automatically discovered, Evolven allows users to add their own applications developed in-house to the scope that is monitored.

When a detailed environment configuration is collected across the end-to-end IT environment a number of decisions need to be made. What information not collected out-of-the-box (typically, due to its proprietary nature) needs to be added to the collection? What information needs to be excluded from the collection to accelerate configuration scans without impacting completeness of the Analytics?

To identify areas of lesser value you should check the following conditions:

- Information in this area is not user/customer data changing frequently by the users themselves as part of application functionality
- Changes made by IT staff or deployment tools and not by the application itself (this is to exclude temporary files, log files and data)
- Information not auto-generated for internal system management (like for example object IDs)
- If you are not sure how to review collected information for completeness, take a look at the previous incidents. These are sample areas to focus on that will give you a good indication on what should be monitored by Evolven.

The types of information to look at can include things such as:

- Configuration files
- Values in the registry
- Content of database table
- Permissions on files
- ActiveDirectory objects
- Etc.

While the actual list of possible configuration sources to explore includes many more possibilities, these examples are meant to emphasize the collection possibilities of the Evolven agent.

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.