A Solution Blueprint for DevOps

Executive summary

Industry surveys and reports describe the highest achieving companies as those turning to a DevOps infrastructure implemented in accordance with best practices to support rapid-paced Agile and non-Agile based product development processes with ever more frequent release deployments to customers. But many organizations are challenged by the migration to these new processes.

The DevOps solution blueprint encompasses Continuous Integration (CI), Continuous Test (CT), Continuous Delivery (CD) and Continuous Change Management (CCM) capabilities with automated orchestration of all operations necessary for the rapid-paced product development essential for business performance.

This white paper prescribes a DevOps solution blueprint that incorporates the following capabilities:

- Integrated architecture for optimum CI, CT, CD and CCM operations
- Hierarchical test execution model to automate all lifecycle test phases
- Tools framework including Restful APIs to orchestrate all DevOps operations
- Expert services that support a logical phased implementation plan and a smooth transition from the existing infrastructure minimizing disturbances of the existing development, test and release process
- Key Performance Indicators (KPIs) and a dashboard that provides the requisite visible measures necessary for Continuous Change Management (CCM) and monitor operational performance of the DevOps system itself to guide operations and continuous improvements.

Without continuous control, the code development complexity curve becomes unstable, creating a “big bang process” that results in process failures such as late deliveries, poor quality release content and cost overruns. CT together with CCM delivers progress monitoring and control features required to reduce complexity by ensuring corrective actions are taken early and often before problems build up into a big bang.

The DevOps solution blueprint applies whether an organization is just getting started with Agile or currently employs Agile, waterfall or waterfall-scrum processes. It also applies whether the environment is hosted on physical systems, virtual systems or a combination of the two.

Readers!

Please take note: acronyms, abbreviations and definitions used in this document are listed at the end of this document.

A Solution Blueprint for DevOps

While the implementation of a best practices DevOps infrastructure is not easy, the benefits are compelling:

- Catch product defects as early as possible in the development cycle
- Perform CI, CT and CD in frequent and short cycles
- CapEx and OpEx expenses saved due to efficient resource monitoring
- Accelerates product delivery velocity, innovation and time-to-market
- Improves infrastructure administration
- Improves process controls with data-based dashboard decision tools
- Reduces down time by ensuring effective backup, archival and recovery
- Improves Intellectual property protection and security

The implementation of a best practices DevOps solution requires an array of capabilities including test orchestration tools, assessment services, 3rd party integration services, KPI/dashboards, and resource utilization optimizations that may include virtualization of DevOps components within private and hybrid cloud deployments. Case studies based on real deployments by experienced DevOps experts have proven that a DevOps solution, when implemented according to best practices, quickly delivers powerful and compelling ROI.

Problem and historical context

Problems context

Software development best practices are trending toward ever more rapid and frequent release deployments to customers. The Agile software development methodology is typically employed. An effective Agile environment dictates that software development work be done in small steps within continuous cycles and that new software feature and fix releases be fast-and-frequent-to-market so customers get new features and fixes quickly, and suppliers get revenue and customer feedback quickly. The efficiency of releases is a competitive advantage for a business because it can have a major impact on business performance.

In environments where best practices are invoked, the Agile software development process is supported by a highly automated and controlled DevOps infrastructure. At the heart of a DevOps system, which is implemented according to best practices, are efficient continuous operations that automatically orchestrate all the tools and process components needed to realize a rapid cycle including source code changes, merging of code changes, host system initialization, build process execution, automated release packaging, multiple levels of automated testing, and automated results reporting and analysis. The end-to-end integration cycle, from code change commit, to reporting of results, must be extremely fast, usually in the order of minutes or hours compared to older software development practices that typically take days to complete the equivalent steps. The speed and frequency of the DevOps cycles necessitates automated orchestration of all CI, CT, CD and CCM operations.

When the DevOps infrastructure is not implemented according to best practices, then the following problems typically result:

- Large, complex software merges bottle-neck integration because large merges require longer build and test times which violate the best practice that prescribes frequent, quick change cycles.
- Excessively long software build and test cycles caused by lack adequate resources in the build and test environment.
- Proliferation of costly dedicated equipment to ensure there are sufficient resources to meet peak demands of the DevOps system.
- High rate of defects found by Quality Assurance (QA) or customers causes by rapid but inadequate test coverage.
- Time-consuming defect troubleshooting and fixes if there are too many changes or failed tests per cycle.
- Integration problems found late in the release may occur if the test coverage is not continuously checked during the development period.
- Missed delivery dates due to efficiency and quality problems.
- Inadequate process control in which problems persist across multiple code commit cycles without being noticed until later, necessitating costly reverts to earlier versions in order to fix the cumulative effects of the problems which crept in over time.
- Fault -intolerance, backup problems, and recovery problems are very costly and prevalent in poorly designed DevOps systems that fail to consider the need to have redundancy, and frequent back-ups for all the build, test, and tool data.
- Security and Intellectual Property protection problems occur when access restrictions to ALL of the components are overlooked. This may occur when the various DevOps system components do not follow a consistent security model for access controls and don't take care to obfuscate credentials or source code IP within orchestration scripts.

In a DevOps environment implemented in accordance with best practices, elastic on-demand build and test resources accomplish fast, frequent cycles, without sacrificing test coverage or dedicating costly under-utilized equipment. DevOps environments may demand multiple copies of systems which is expensive if those resources are dedicated physical systems. The solution to control the cost of multiple on-demand computing resources needed for parallel building and testing is to virtualize the components of the system being built and tested and also virtualize the test systems whenever possible. In this way a test system/system under test combination can be deployed on-demand as virtualized resources. The use of virtualized resources allows the host equipment to be deployed on-demand and re-used for other code lines or other purposes when not needed for a particular product or code line build or test job.

A Solution Blueprint for DevOps

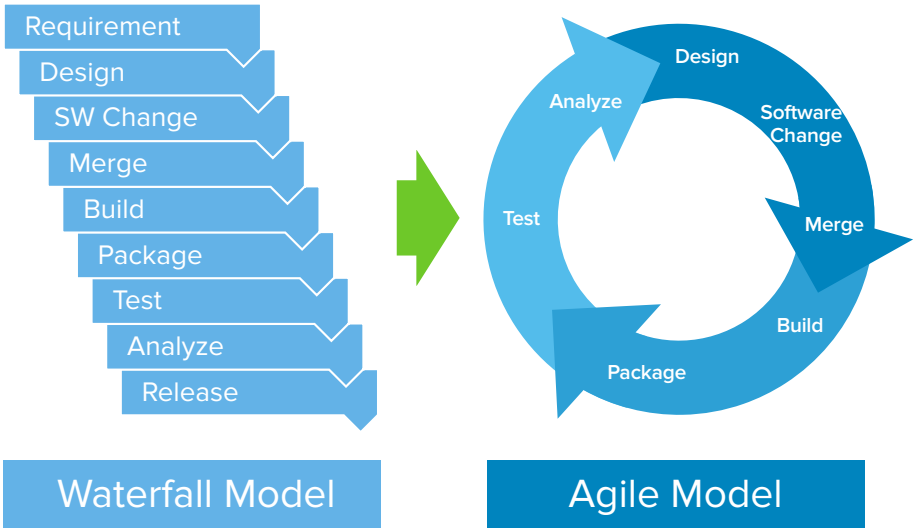


Figure 1: Waterfall and Agile Software Development Models

Historical context

Prior to Agile, the most prevalent software development methodology was the sequential "Waterfall model". Figure 1 compares the sequential nature of waterfall with the iterative, cyclic nature of Agile. Agile process typically would iterate many times to achieve the same results as one waterfall cycle, but with much less effort and higher quality.

In the waterfall model it is assumed each step in the process is completed for the entire system being developed prior to completing the next step. This resulted in a long time between requirement definition and release to customers. Each step is dependent on the prior step so problems in any one step would cause delays for the entire project. All too often the long development period and the stepwise delays causes projects to miss their deadlines. In many cases, the customer requirements change by the time the project is completed. The problems associated with the Waterfall Model ultimately drove the creation of the Agile Development Model.

According to http://agilemethodology.org/ "It's easy to see the problems with the waterfall method. It assumes that every requirement can be identified before any design or coding occurs. Could you tell a team of developers everything that needed to be in a software product before any of it was up and running? Or would it be easier to describe your vision to the team if you could react to functional software? Many software developers have learned the answer to that question the hard way: At the end of a project, a team might have built the software it was asked to build, but, in the time it took to create, business realities have changed so dramatically that the product is irrelevant. Your company has spent time and money to create software that no one wants. Couldn't it have been possible to ensure the end product would still be relevant before it was actually finished?"

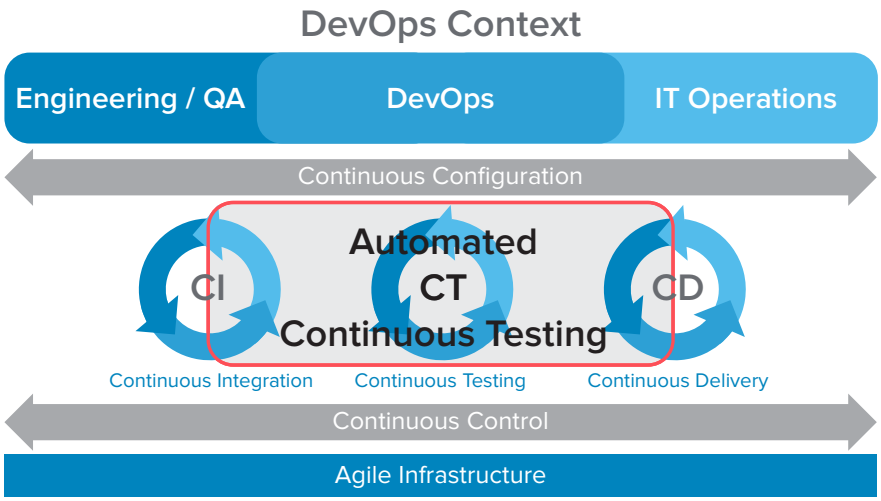


Figure 2: Continuous Testing is the heart of the DevOps System

According to http://en.wikipedia.org/wiki/Agile_software_development "Agile software development is a group of software development methods in which requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development, early delivery, continuous improvement and encourages rapid and flexible response to change. It is a conceptual framework that focuses on delivering working software with the minimum amount of work. The Agile Manifesto introduced the term in 2001. Since then, the Agile Movement, with all its values, principles, methods, practices, tools, champions and practitioners, philosophies and cultures, has significantly changed the landscape of the modern software engineering and commercial software development in the Internet era." Figure 1 shows the relevant concepts of the Agile Software Development Model.

Agile software development involves continuous integration (CI) cycles and each CI employs automated continuous testing (CT) to verify the result before the next change cycle is started. In this way the software changes are controlled continuously.

An efficient DevOps infrastructure consisting of automated processes, tools and methodologies is essential to fully realize the goals of Agile and rapid product development. According to http://en.wikipedia.org/wiki/DevOps, DevOps "aims to help an organization rapidly produce software products and services. Simple processes become clearly articulated using a DevOps approach. The goal is to maximize the predictability, efficiency, security and maintainability of operational processes. This objective is very often supported by automation."

Figure 2 illustrates the DevOps Model and shows how Continuous Testing is at the heart of system.

A Solution Blueprint for DevOps

DevOPs solution blueprint architecture

The DEVOPS Solution Blueprint architecture, presented in Figure 3, addresses the problems described in section 2.1 by implementing Continuous Integration (CI), Continuous Testing (CT), Continuous Deployment (CD) and Continuous Change Management (CCM) sub-systems within a modern DevOps infrastructure context described in section 2.2. The DevOps solution blueprint architecture addresses these problems regardless of whether they occur in Agile or non-Agile development environments. The DevOps blueprint supports physical, virtualized, and mixed physical/virtualized host and lab network environments. The DevOps blueprint architecture supports configuration and control across multiple product boundaries, service layers, and software versions, even in cases where systems must be verified in parallel, on-demand, and in real time.

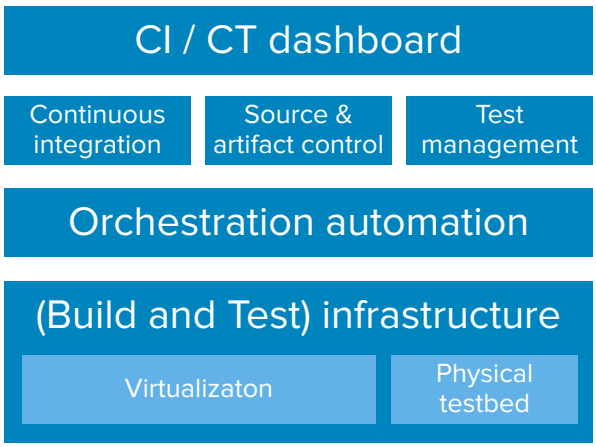
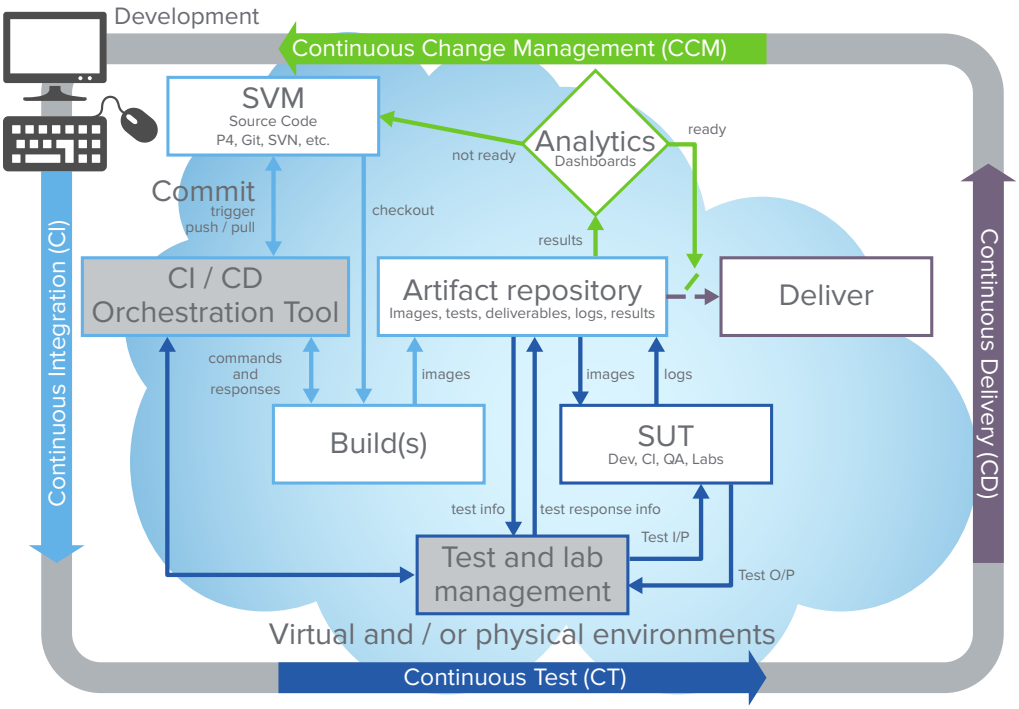


Figure 3: DevOps Solution Blueprint Architecture

The DevOps Solution Blueprint architecture is composed of the following modules:

1. **Dashboard:** At the highest level of the architecture a dashboard provides information for continuous change management. The actual dashboard implementation varies according the Key Performance Indicators (KPIs) that are most relevant to a particular customer. Therefore the Dashboard is typically customized for each installation.
2. **Continuous Integration:** Automation framework that provides centralized orchestration of the entire DevOps system.
3. **Source and Artifact Control:** Database of source code. Software packages that are used by CI system build processes.
4. **Test Management:** Database of test scripts and artifacts needed for testing
5. **Orchestration Automation:** automates tests and test topologies
6. **Build and Test Infrastructure:** Virtual and/or physical test beds.



DevOps tools framework

The DevOps Solution Blueprint framework presented in Figure 4 consists of the following elements:

1. **A Software Version Management (SVM)** system supports software development by providing software configuration control and identification, visibility into the status of software changes, and management of versions of the tools needed to implement the build process. In a best practices DevOps system the SVM includes a well-designed, flexible repository structure and supports effective software branching and merging capabilities. This capability is critically important for Agile development processes, as developers merge changes back into the repository frequently. Effective Software Change Management (SVM) systems often include a mix of open-source and commercial software with small, custom modules tailored to the specific development organization. Examples of SVM Systems are: Subversion, Git, Mercurial, Perforce, and ClearCase.

2. A **CI/CD Orchestration Tool** is a programmable system that executes scheduled and event-driven, jobs. The most popular CI/CD orchestration tool is Jenkins.

3. A **Build System** compiles and packages build artifacts. CI provides rapid feedback to the entire team about the current state of the software. This feedback improves quality, lowers costs, and enables management to make informed decisions on how to most effectively allocate resources to meet mission objectives. DevOps leverages this practice by automating the build process and then triggering the start of a quality cycle when a set of conditions is met. A trigger is typically a commit into a monitored branch, though multiple commits can be cached and generate a single trigger event. DevOps orchestration tools manage the build servers, physical and/or virtual, spinning them up as needed. This feature allows the organization to efficiently make use of private, hybrid or public cloud computing resources.

Figure 4: DevOps Solution Blueprint Tools Framework

A Solution Blueprint for DevOps

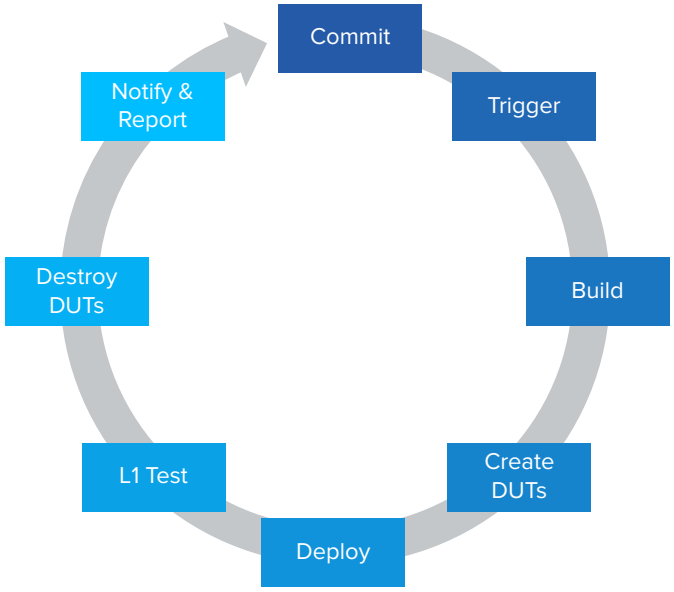


Figure 5: DevOps Cycle with Virtualized SUT

4. An **Artifacts Repository** contains completed build packages and information required to test the packaged builds. The repository is an organized store of artifacts related to the product. In addition to source and build artifacts the repository may include, test artifacts such as test scripts, logs and test results data related to issue and ticket tracking, compliance and regulatory artifacts, development and user documentation, and configuration data. Examples of Artifacts Repositories include an NFS file share, Zephyr and HPQC. The repository may be implemented as a single database or multiple databases.

5. The software for the System Under Test (**SUT**) or device under test (DUT) may be hosted by a physical or virtualized device, or network of devices. As shown in Figure 5, when the SUT is virtualized the CI cycle includes steps to create virtual machines to host the SUT software and when the tests are completed the DevOps system destroys the SUTs so the host resources are available for other purposes.

6. **Test and Lab Management Systems** may also be hosted on physical or virtualized platforms. In a DevOps system implemented according to best practices, functional and system level QA testing are fully automated. Many organizations are reluctant to perform automated QA testing as part of CI due to increased cycle times and unreliable automation. A DevOps system implemented according to best practices, is able execute stable QA regression runs while dynamically managing virtualized and physical hardware resources and scale the tests dynamically according to resource availability for the allotted test time. Examples of Test and Lab Management Orchestrations Systems are Spirent's iTest and Velocity.

7. A **Delivery system** packages release artifacts such as software packages in formats consumable by customers and provides controls to determine when to promote a packaged product version to customer release status.

8. Analytics collect and analyze reports and logs from the CI, CT and CD systems to determine whether any additional changes are required to complete a quality cycle. For example, tests that result in failures may automatically determine that a change in the SVM system must be reverted and may automatically trigger the revert process.

9. A DevOps host environment may be private, public or hybrid cloud services (e.g. AWS, Rackspace). The cloud is an abstract term used to generalize various types of remote storage, computation and user access services. While cloud-computing infrastructures may consist of many types of services, most implementations provide one or more of the following service models:

   a. Infrastructure as a Service (IaaS) – This service model provides basic computing functionality, typically in the form of a Virtual Desktop Infrastructure consisting of virtual machines and support resources. The users typically access their desktops from a software client running on low-cost terminal hardware.

   b. Platform as a Service (PaaS) – This service model provides platform specific services, typically in the form of virtualized servers such as Software Defined Networking (SDN), domain controllers, web, database, and development servers.

   c. Software as a Service (SaaS) – This service model provides access to virtualized software application running in the cloud. This feature provides platform independence and does not require the user to install or maintain the application locally.

   The increasing number of organizations adopting the cloud model indicates the value they are finding in the cloud. Cloud computing is an extremely cost-effective method of deploying, maintaining and securing desktop and server applications. Rather than purchase hardware and software licenses for each member, an organization can serve them up based on demand, not only reducing costs but also enhancing scalability. This applies to DevOps CI, CT, CD and CCM services also.

   By virtualizing DevOps applications, an organization can centralize and streamline management processes. Tasks such as backup, recovery, and installing updates and security patches can be performed quickly. Cloud computing relieves IT departments of a great deal of the logistical burden of maintaining a desktop environment. Installs, updates, and other maintenance items are accomplished at a central location rather than having to travel to the user's site.

   The **Virtual Machine environment** (e.g. VMware VSphere) may be a variety of hypervisors and containers depending on the specific environments that each tool supports. The DevOps blueprint is agnostic regarding specific hardware or cloud-based virtualization technologies. Delivering build, test, and assessment services on these platforms allows the CI, CT, CD and CCM process to be extremely stable and elastic. The ability to perform DevOps at scale in a private or public cloud is a key driver for reducing CapEx and OpEx costs associated with software development. In a fully virtualized DevOps environment both the test system and SUTs use VMs that are deployed on-demand.

10. The DevOps blueprint solution includes back-up, and recovery of build and test artifacts such that in the case of system failure the entire system can be recovered from archived or backed up artifacts quickly.

11. The DevOps blueprint solution obfuscates system credentials and provides access controls to ensure unauthorized access to sensitive source code and Intellectual Property.

A Solution Blueprint for DevOps

Hierarchical DevOps solution blueprint test execution model

To fully integrate QA testing, a hierarchical model for test execution is recommended. This model consists of three levels of testing:

- **Test Level 1 (L1)–**QA regression run as part of CI cycle
  - Software static analysis
    - Compiler warnings
    - Coding standards compliance (Format and metrics)
    - Security assessment (Unsafe coding practices, implementation vulnerability detection)
  - Software dynamic analysis
    - Memory re-use and/or leaks
    - Thread contention/deadlock, multi-core scaling, cache access
    - Undefined behavior
  - Unit test and code coverage
- **Test Level 2 (L2)–**Nightly Integration and regression
  - End-to-end feature testing
  - Hardware/software integration test
- **Test Level 3 (L3)–**Weekly or release
    - Interoperability compliance (3GPP, IETF, ITU)
    - Regulatory compliance (FCC, HIPAA, FISMA)
    - Product/system development security assessment (HIPAA, PCI, NIST)
    - Performance assessment (Bandwidth, error rate, signal to noise ratio)
    - Operational security assessment (HIPAA, PCI, NIST)

The levels listed above provide different types of QA coverage and are executed at different intervals. L1 testing is a breadth-first test cycle focused on testing the entire system with each quality cycle. L1 regression runs are kept short and typically include strategic test points across the system. These test points focus on the integration of major components and detecting integration defects. L2 and L3 test are typically scheduled rather than being run as part of CI. The L2 regression is typically run each night and includes tests focused on system components that are changing in the current release. The L3 regression is a full battery of tests focused on detecting low-level defects. L3 is typically scheduled on a weekly basis or as part of release acceptance.

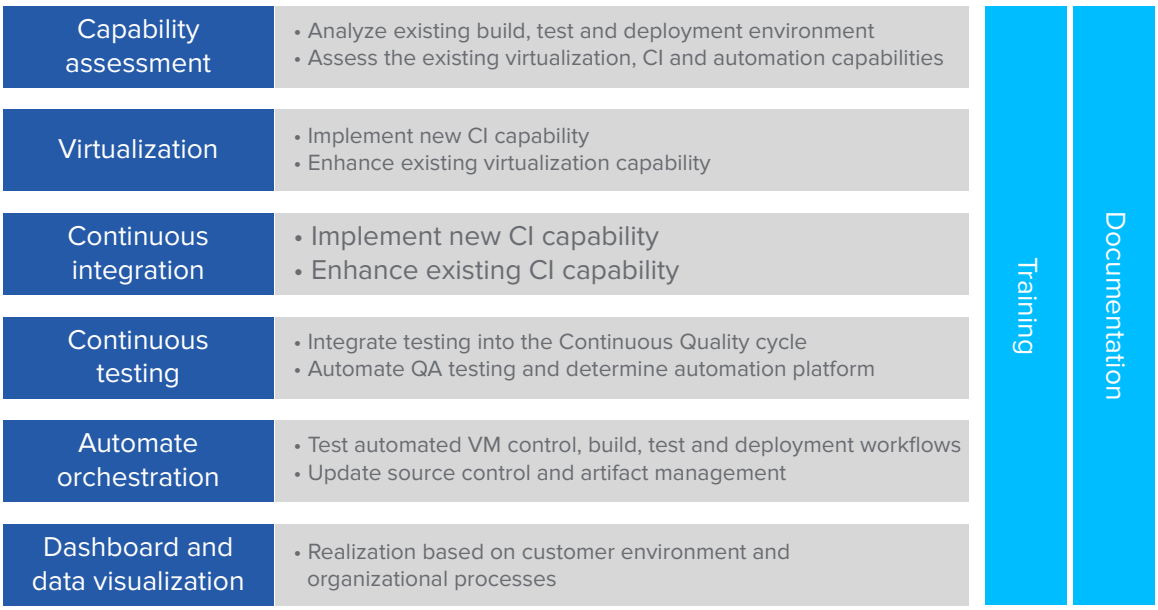


Figure 6: DevOps Blueprint Solution Services

DevOps solution blueprint services

No two organizations are the same. Many organizations have some level of automation, CI, test and virtualization but may not be operating at best practices levels for all parts of their DevOps CI, CT, CD and CCM sub-systems. It is important to engage experienced experts to assess an organization's current DevOps status, goals and implementation plans. Figure 6 defines DevOps Solution Blueprint Services that need to be considered.

Through the capability assessment, a strategic plan is developed that will migrate the team's existing infrastructure, with minimal impact to current operations.

KPI and Dashboards

Metrics or Key Performance Indicators (KPI) are frequently generated and updated to provide timely status reporting to organizational leadership. The specific KPI and presentation will vary depending on organization preferences.

The following are examples of measureable KPI's that are relevant and can be derived from DevOps deployments:

- Percent of tests automated
- Percent availability of automation test infrastructure
- Execution times for builds and tests
- Product Velocity: release frequency
- Test efficiency (e.g. number of acknowledged bugs divided by the total number of potential bugs reported)
- Total Capex and Opex cost reduction compared to other approaches (e.g. ROI case study)

A Solution Blueprint for DevOps

Benefits of the Devops solution blueprint

Figure 7 illustrates the primary benefit of a DevOps system implemented according to a best practices solution blueprint:

To provide automated continuous control of the entire development-test-release cycle in the context of Agile and non-Agile rapid continuous integration complexity.

Without continuous control, the complexity curve becomes unstable, creating a “big-bang” process that results in failures such as late deliveries and poor quality release content. CT and CCM bring the control required to reduce complexity.

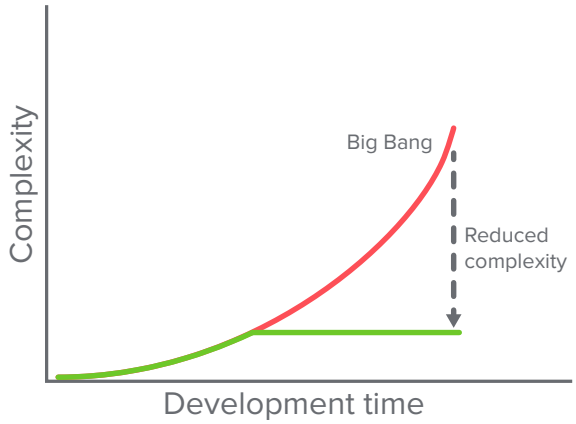


Figure 7: DevOps controls development time despite increasing complexity

The DevOps Solution Blueprint benefits include:

- Catch issues as early as possible
  - Perform CI (build) and CT (test) in short, frequent cycles
- Save on CapEx
  - Achieve efficient utilization of virtualization environment in combination with physical test beds
- Save on OpEx
  - Create control sequences quickly. Use Orchestration Automation to enforce change management and consistency in reporting, build environments, and test-bed interactions. Achieve a consistent, reliable, repeatable development process with a consistently managed environment.
- Focus on your business and accelerate time to market
  - Follow best practices delivered in professional services and tooling to focus on your products and not the supporting infrastructure.
- Make evidence-based decisions
  - Promote data-based decision by keeping everyone on the same page all the time with a CI/CT dashboard customized around your internal processes and existing systems.

Summary

Whether an organization is just getting started with Agile or similar fast paced product development, or currently employs Agile or DevOps processes but wants to improve the efficiency of their environment the DevOps solution blueprint described in this document provides a comprehensive framework and a step-wise approach towards implementing a solution.

Anyone considering installing or updating their Continuous Integration environment can utilize the DevOps Solution Blueprint to accomplish:

- Solution architecture suitable for optimum Continuous Integration
- Hierarchical Test Execution Model
- Tools framework including orchestration of CI, CT, CD cycles
- Virtualized CI, CT setups
- Services that support the strategic improvement plan
- A logical phased implementation plan that minimizes disturbances of the current system
- KPIs and Dashboard to measure performance of the CCM solution

A Solution Blueprint for DevOps

Spirent's capabilities for DevOps

Spirent has expertise and experience implementing successful DevOps deployments consistent with best practices and in accordance with the DevOps Solution Blueprint described in this white paper.

Spirent, through its test tool products and professional services capabilities, brings together open source software, third-party commercial software, and unique test automation capabilities to enable customers to improve quality and productivity in a scalable and cost-effective DevOps solution called "**CLEAR DevOps**".

The Spirent **iTest** and **Velocity** products are perfectly suited to provide test orchestration for best practices DevOps deployments . At the heart of the CLEAR DevOps solution is an integration of Spirent CT tools with CI called the Efficient Virtualized Continuous Integration (**EVCI**).

To support virtualized and cloud-based DevOps deployments Spirent has developed plug-ins. Additionally, Spirent has native session types that allow direct communications with enterprise virtualization management platforms.

Spirent professional services support DevOps upgrades or entire DevOps implementations depending on the customer goals and assessments. Project management, integration of iTest orchestration, implementation of virtualization and DevOps dashboards are examples of project components that Spirent offers.

Figure 8 presents an overview of the phased approach Spirent developed for CLEAR DevOps.

Case studies are available that demonstrate the benefits accomplished by Spirent implemented Clear DevOps (powered by EVCI).

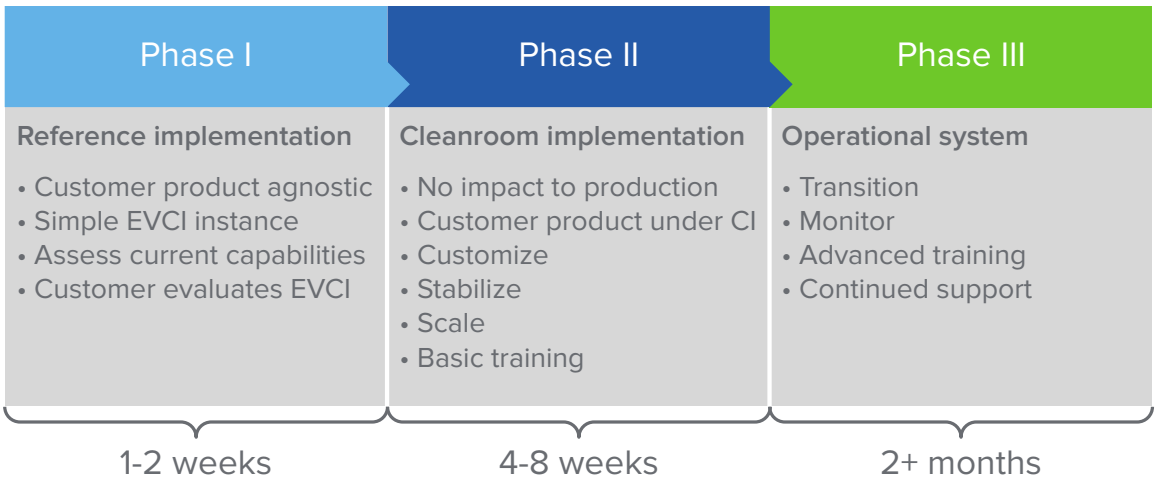


Figure 8: CLEAR DevOps Phased Implementation Approach

Reference links

Agile: software development methodology: http://www.agilealliance.org/the-alliance/the-agile-manifesto/

DevOps: software development method: http://en.wikipedia.org/wiki/DevOps

Jenkins: An extendable open source continuous integration server: http://jenkins-ci.org

Jira: proprietary issue tracking product, developed by Atlassian. It provides bug tracking, issue tracking, and project management functions https://www.atlassian.com/software/jira

Open source software for building private and public clouds: http://www.openstack.org/

Spirent iTest: http://www.spirent.com/Products/iTest

VMware: http://www.vmware.com/

Zephyr: Real Time Test Management System: http://www.getzephyr.com/

Acronyms, abbreviations, and definitions

API	Application Programming Interface
BDD	Behavior Driven Design
CapEx	Capital Expense
CD	Continuous Delivery
CI	Continuous Integration
CLEAR	Collaborative, Leadership, Exploration, Acceleration, Realization
CT	Continuous Test
DevOps	Development Operations
DUT	Device Under Test
EC2	Amazon Cloud Service
EVCI	Efficient Virtualized Continuous Integration
IaaS	Infrastructure as a Service
ILO	iTest Lab Optimizer
iTest	Spirent Capture Execute Report test tool
ITO	Infrastructure Test Optimization
Jenkins	extendable open source continuous i ntegration server
Jira	issue tracking product
KPI	Key Performance Indicator
KVM	Hypervisor
OpEx	Operating Expense
PaaS	Platform as a Service
QA	Quality Assurance
QEMU	Hypervisor
SCM	Software Change Management
STC	Spirent Test Center
SUT	System Under Test
SVN	Subversion
TaaS	Testing as a Service
TDD	Test Driven Design
VM	Virtual Machine
VMware	Company which produces VSphere
VSphere	Server virtualization platform
Xen	Hypervisor

White Paper

A Solution Blueprint for DevOps

spirent.com

AMERICAS 1-800-SPIRENT
+1-818-676-2683 | sales@spirent.com

EUROPE AND THE MIDDLE EAST
+44 (0) 1293 767979 | emeainfo@spirent.com

ASIA AND THE PACIFIC
+86-10-8518-2539 | salesasia@spirent.com

Rev A. 01/15