stonebranch

TECHNICAL BRIEF

Scheduling and Orchestration of Heterogeneous Docker-Based IT Landscapes

January 2017
Version 2.0 – For Public Use

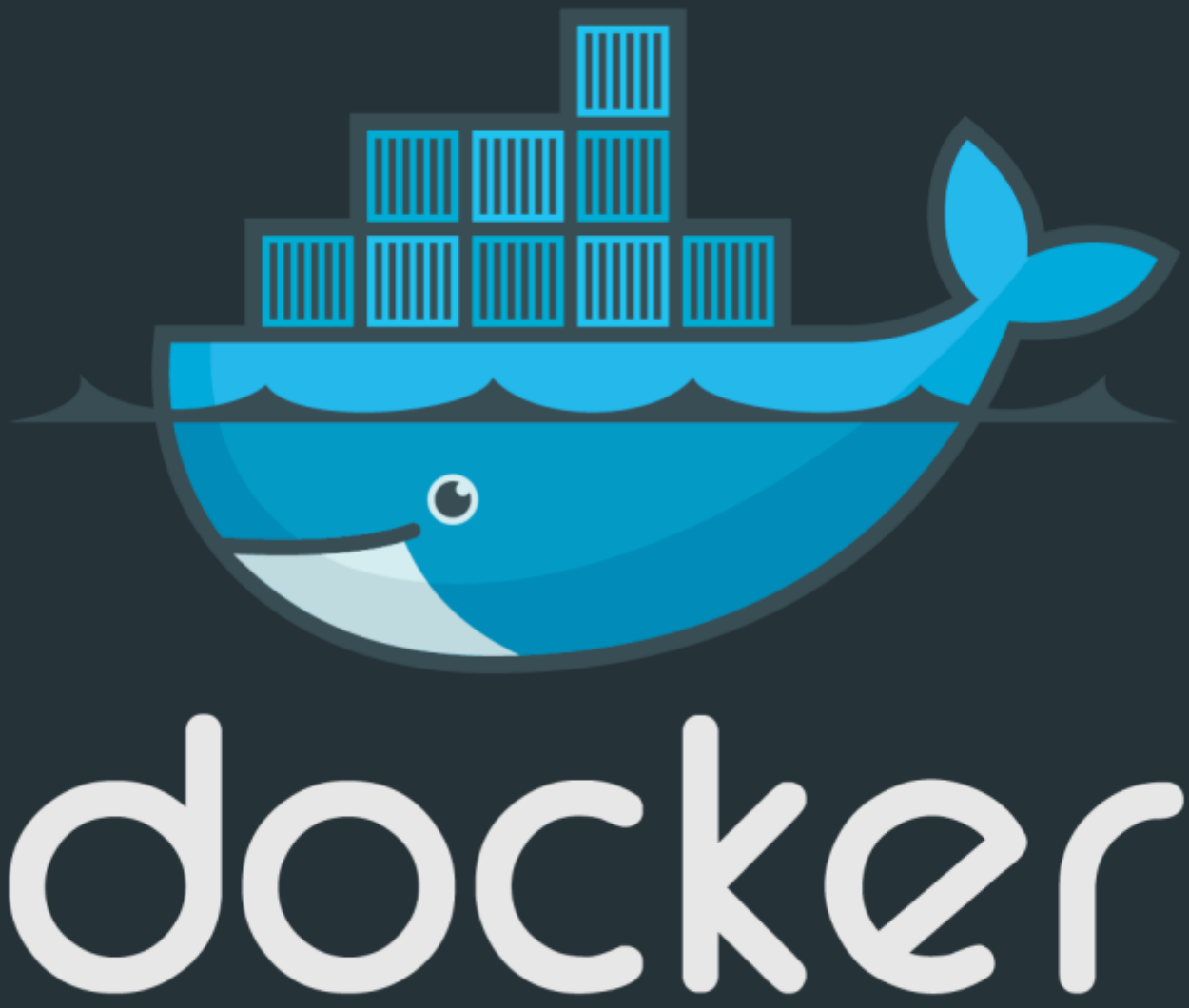

Table of Contents

1 Summary

This technical brief describes how Stonebranch seamlessly integrates your DevOps environments with legacy systems in your IT landscape without the need to re-design your business process logic. As you will learn, this results in a **shorter time-to-market, improved customer satisfaction, better product quality, more reliable releases, improved productivity and increased efficiency** to deliver the highest degree of Return on Automation.

Figure 1: Overview - Return on Automation Results

2 Introduction

Universal Automation Center enables you to **efficiently manage and integrate** your container strategy with **your legacy IT Systems**, without the need to re-design your current business process logic. As a result, you can make use of all the benefits provided by containers like portability from applications, simplified integration, optimized development, and increased scalability and performance, while simultaneously minimizing the risk associated with introducing a new technology.

Docker containers are ideal for all your applications, whether they are stateless or have their state in an external memory such as a database e.g. web applications, backend-API's, maintenance scripts, regular triggered reports, short-term jobs, etc. Those applications suit the container concept of a lightweight, independent, transportable application perfectly.

The idea of a container is not to replace a virtual machine, which is usually created to replace true computer hardware, but to provide an application in a defined portable environment, which can be started on demand when a job needs to be executed and dropped afterwards. The application can be as light as a single Unix service.

3 Stonebranch DevOps Concept

The concept of centrally managing all job-specific environment variables and scripts for container and legacy systems allows for an ideal DevOps approach in which no environment-specific settings have to be done as part of the deployment. If a workflow has been tested, it can be deployed to production without any manual configuration.

The web-based GUI found within Stonebranch's Universal Automation Center provides an end to end view of the entire business process consisting of container-based and legacy applications. This brings **cross functional teams** together, including architects, developers, testers, and operators. Each team will get the views and access rights for the information they require.

Once a business process has been tested, Stonebranch's lifecycle management system **"bundle & promote"** allows each team to package all configuration items they are responsible for, without affecting any other team. Each package can then be automatically promoted to the OPS landscape at a defined date and time without any manual configuration.

One of Stonebranch's major European banking customers has one team for each of their 400 applications. Each team can independently work on the schedules related to their applications without affecting any other team. Universal Automation Center made this possible.

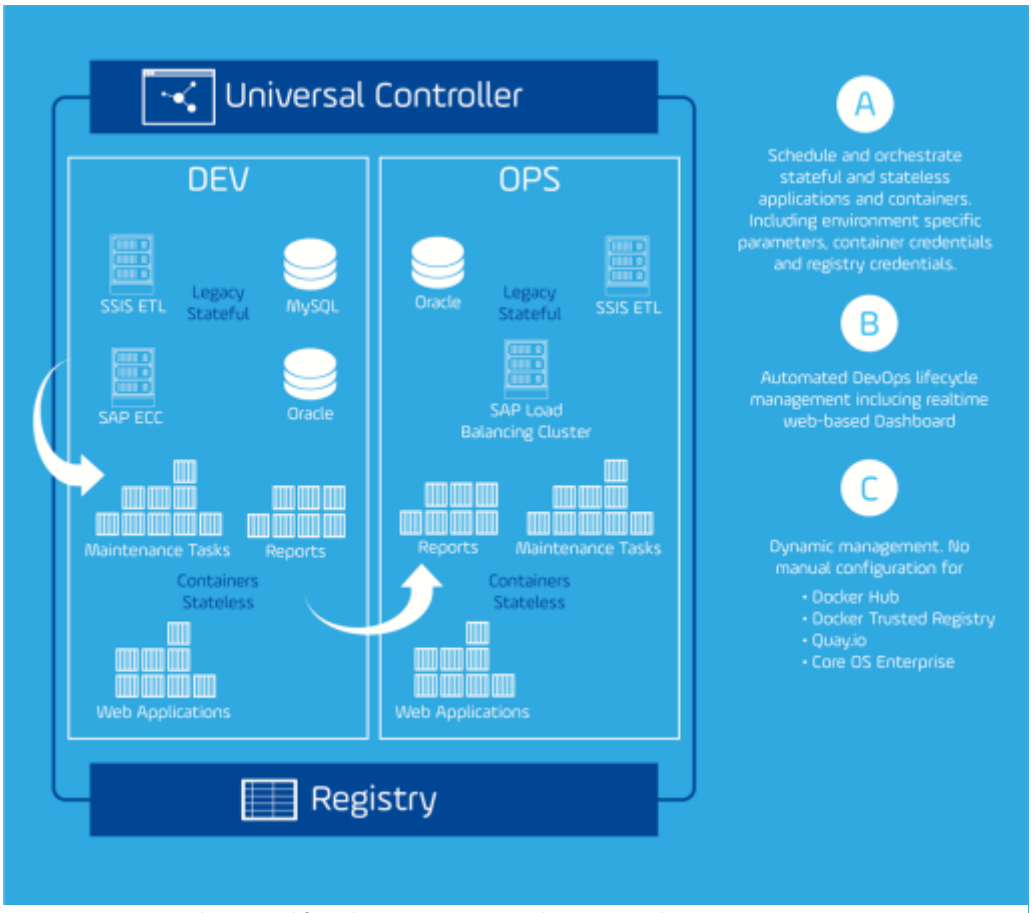

Figure 2: Automated DevOps lifecycle management without manual intervention

4 How it works

In order to start introducing containers we recommend you to start with lightweight applications that are either stateless or externalize their state in an external database. Once you have identified the applications in your business process you wish to run in a container, you can automatically schedule them, as you do today with your non-containerized applications.

If a scheduling condition is met, Universal Automation Center will dynamically download the latest image version from your containerized application (if it doesn't already exist) from a public or company internal registry, such as Docker Trusted Registry. The container is then started with all required parameters (credentials, IP-addresses, hostnames, ports, etc.) for the applicable environments (like dev, test, prod, etc.). On the development landscape, your container application needs to connect an SAP development user to an SAP development system, where on the production system your container application needs to connect via an SAP application user to the SAP production system. By using variables in the Universal Task for the credentials and the SAP connection, **no manual configuration** is required.

Figure 3: Real-time 100% Web-based Dashboard

The Stdout, Stderr generated by the containerized application are automatically imported into the Universal Controller for error handling, reporting and to define further processing rules in a workflow.

Once the Application has been successfully executed, the container is stopped and removed and the next task in the workflow is executed. All assigned operating system resources for the container are freed again.

Should the application in the container fail, the container remains started to handle further errors. This can result in a restart of the application, skipping the application or introducing a predecessor or successor task in the workflow before a restart of the application. Once the restart is successful, the container is stopped and removed. If a workflow fails, this has no effect on subsequently started workflows, as each workflow is an independent instance in the Stonebranch system.

In order to not overload the number of started containers per Docker server, you can dynamically control the number of containers of a certain type running in parallel on a Docker host by using Universal Automation Center's virtual resources.

The Universal Dashboard provides you the full control and real-time view on your entire workflow.

4.1 Example

The following provides a simple example to understand the Stonebranch DevOps concept. A Python based database maintenance script has to be executed every Sunday as part of bigger maintenance workflow containing several legacy tasks.

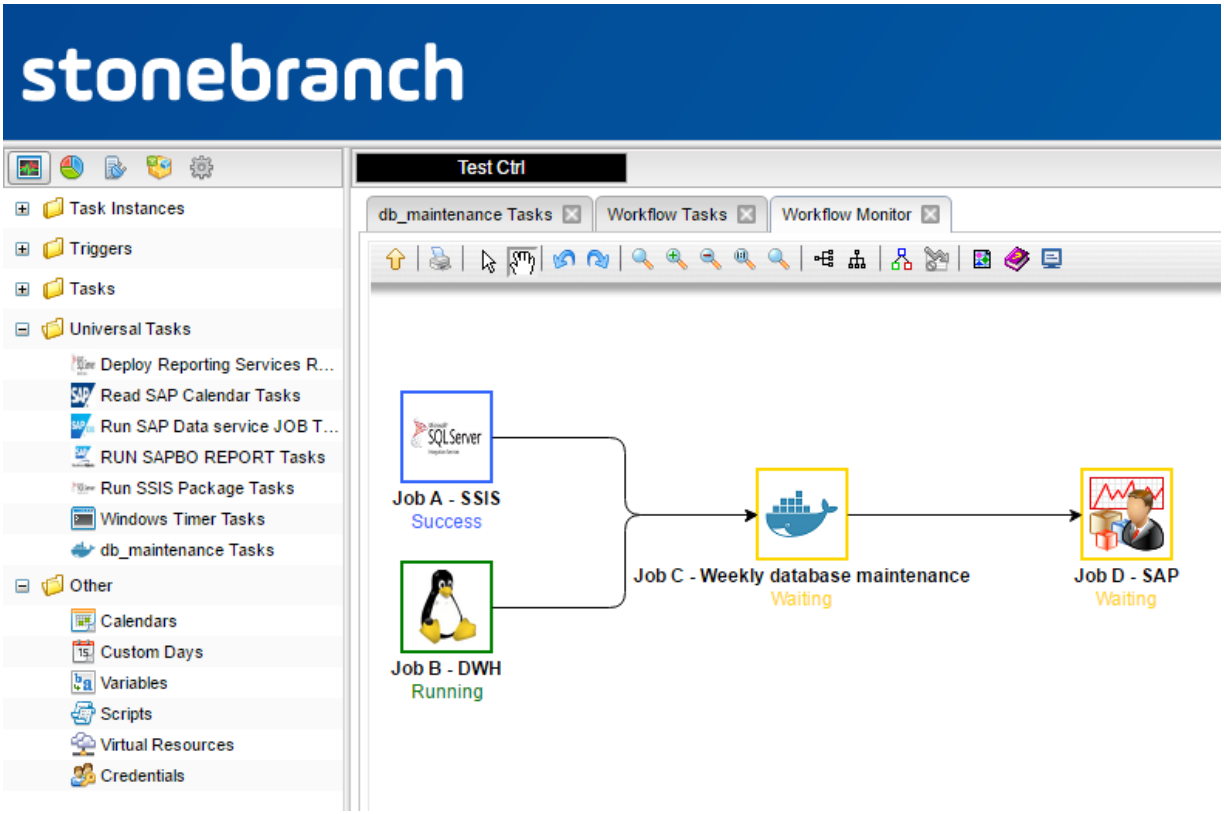

Figure 4: Workflow with Legacy and Containerized Jobs

Description:

The Workflow is automatically started every Sunday using a Universal Automation Center time trigger. Once the legacy SSIS-ETL Job A and the DWH Job B is successful, the container-based Job C is started as part of the Workflow. By launching Job C, a minimal Linux container which includes a database maintenance application in Python is automatically started from a pre-defined container image. The predefined image is retrieved from the selected registry e.g. a company specific Docker trusted registry. According a successful execution of the database maintenance application the container is removed and processing continues with the next task in the workflow. By removing the container automatically after a successfully execution of the applications, all used resources are freed again.

Universal Automation Center automatically handles the process of initiating the container, based on an image in the private or public registry. This includes the following tasks:

- Centrally maintain credentials for all used registries
  (Docker Hub, Docker Trusted Registry, Quay.io, etc.)
- Retrieve the image from the configured registry
- Initiate the container from the image with all environment specific
  parameters like environment type, container credentials, database credentials,
  connection parameters
- Starting the container
- Web-based monitoring and control of the entire process consisting of container
  and legacy application
- Removing the container after successful execution
- Reporting on the execution of the process

All required configuration parameters for a containerized application are defined in a **Universal Template** and configured using the corresponding **Universal Task for Docker**.

The following screenshot provides the Universal Task example for the weekly database maintenance application running in a Linux/Python container. The Universal Task consist of configurable Web-form and an underlying Universal Template. The Web-form contains all required input and connections parameters to automatically run the container based on the selected image. It is used for user friendly daily operations, hiding all information not required for daily operations.

The Universal Template contains the underlying Docker CLI based configuration including Docker, Docker composer and other configuration files. The developer can define here how his application is started using a single or multiple Docker container.

The following screenshot show the Web-form used by the daily operations team:

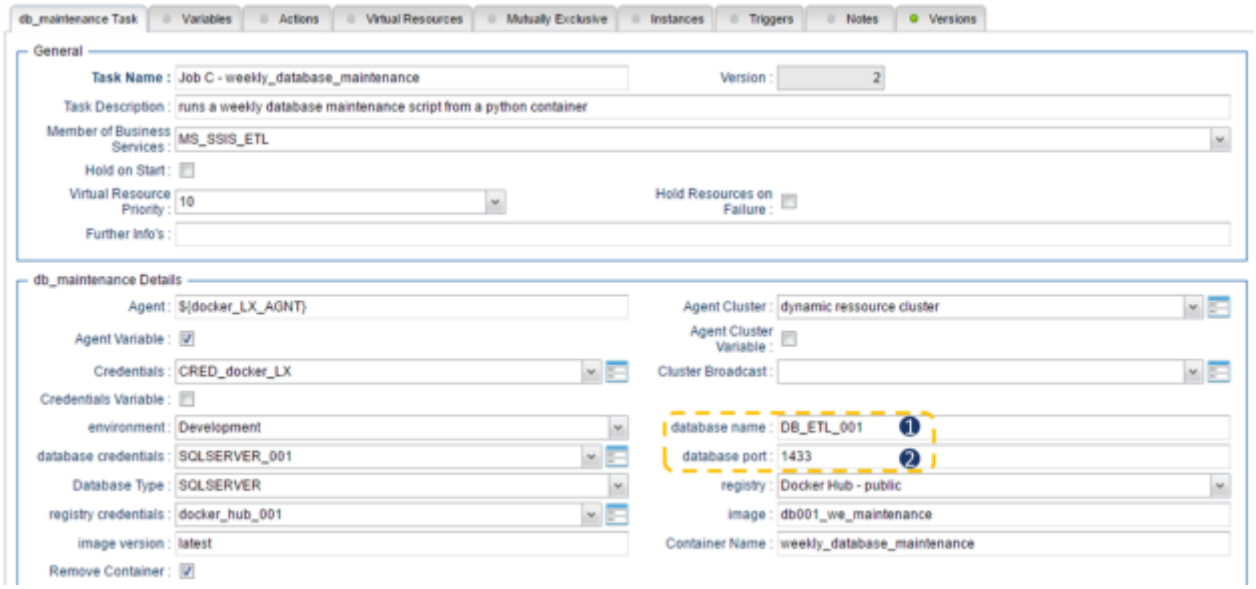

Figure 5: Fully configurable Universal Task for Docker

The following screenshot shows the underlying Universal Template:

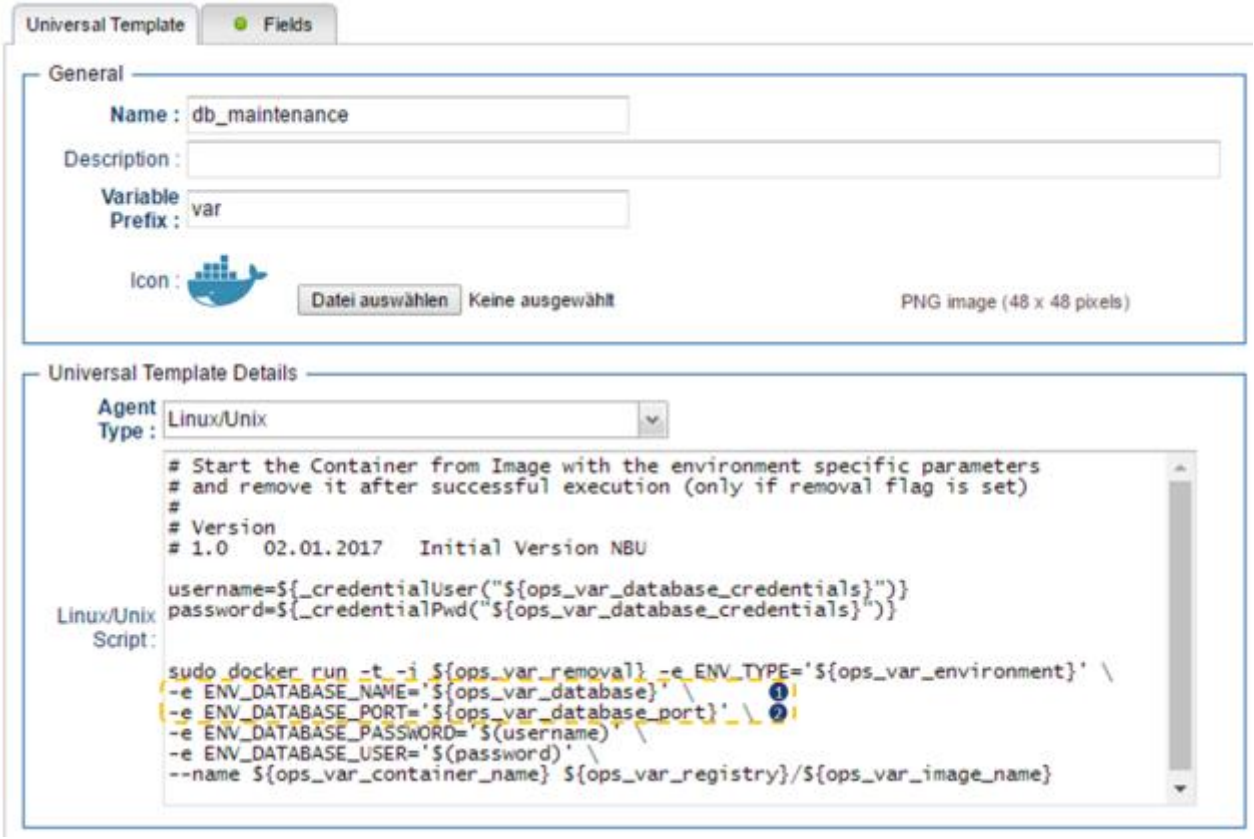

Figure 6: Universal Template for Docker

The Input Parameters defined in the Web-form e.g. ***database name, database port*** etc. are passed to the Universal Template for executing the defined Docker CLI command e.g. ***docker run***. In the given example the database name and port are mapped to the template parameter ***${ops_var_database_name}*** *and* ***${ops_var_database_port}***.

Note: The example above is used to explain the Stonebranch DevOps concept. In addition, more complex scenarios like dynamically starting or stopping container instances running a web-application in order to achieve the optional performance are possible.

Furthermore, Docker containers are generally not limited to a lightweight application providing just one service. Windows Server 2016 Containers running .NET applications or SQLSERVERS, for example, are quite heavy compared to a minimal Linux container A lightweight "Nano" Windows Server including .NET core requires ~400MB disk space compared to 12Kb for a minimal Linux image.

5 Security

When working with containers, security is key. Containers are more or less isolated from the host OS system, but they use the same kernel. Due to potential security risks, a container should be executed in using a non-privileged user account. In some cases, it may be necessary to use some kernel functionalities, when mounting a USB drive, for example. Universal Automation Center allows one to centrally maintain individual credentials for each container.

This security concept has been consistently validated by external BSI certified security companies, including detack and Secuvera.

6 Key Benefits

The following summarizes the main benefits of using Universal Automation Center for scheduling your heterogeneous IT landscapes containing legacy and containerized applications:

- DevOps
  - Optimal DevOps approach – no manual environment specific settings have to be done as part of the deployment process
  - All job specific environment parameters, credentials and scripts for containers and legacy systems are centrally managed
  - Lifecycle management to transfer tested business processes from Dev to Ops without any manual configuration in the Ops environment

- Web-based
  - Real-time dashboard providing an end-to-end view on the business process consisting of **container based and legacy applications**
  - Customizable Web-GUI, bringing cross functional teams together e.g. architects, developer, testers, operators.
  - Central script library for file transfer, database calls and shell scripts – no need to store scripts in a container image

- ## Security
  - Central Management of all users and groups with support for SSL LDAP/AD for legacy and containerized applications
  - Central Management of all credentials and application connection e.g. Container, SAP and Database connections, script credentials, etc.
  - Our security concept is constantly validated by external BSI certified security companies (August 2016 by detack and in 2015 by Secuvera)

- ## Auditability and KPI based Performance Analysis
  - The **Universal Controller Audit function** maintains a detailed audit record of all user interactions with the Universal Controller
  - **KPI reports** on all executed Jobs legacy and containerized are provided including runtimes, errors, re-tries, etc.
  - Historical data and **predictive analysis reports** allows you to compare your KPI's with past results and predict future performance figures.

- ## Future Readiness
  - Pre-configured Universal Tasks for all major Docker API commands. Any new commands can be added by configuration e.g. for Docker swarm
  - Support for all major Docker interfaces REST and cli-based e.g. Docker-cli including Docker PowerShell for Windows Server 2016, Docker API
  - Microservices scheduling - support for HTTP, SOAP, REST, JMS and IBM WebSphereMQ (Message Queue)

stonebranch