Upgrading Apps Deployed to Container Technologies with No Downtime

**Note:** Though we explicitly mention "applications" throughout this document, since this is the deployment unit for container-based hosting technologies, always have in mind that an "application" represents a set of "modules".

Previously, when the OutSystems platform interacted directly (and exclusively) with IIS — known as "Classic Virtual Machines" deployment technology in OutSystems 11 —, achieving no-downtime during the deployment of new application versions or when updating configurations had little to no impact.

Now, with the support for applications running in containers, achieving no-downtime when executing these processes requires additional steps specific to the hosting technology being used.

While the information provided in this document is mostly targeted at applications being staged between environments using **LifeTime**, there are a few scenarios where this is also relevant in a single environment scenario, e.g. when updating configurations of an application or when moving an application from "Classic Virtual Machines" to a container-based hosting technology using **Service Center**.

**Before you start**

This document assumes that:

- The target container-based infrastructure meets all the [recommended network pre-requisites](https://success.outsystems.com/Support/Enterprise_Customers/Maintenance_and_Operations/Upgrading_Apps_Deployed_to...);
- All other [container-specific requirements](https://success.outsystems.com/Support/Enterprise_Customers/Maintenance_and_Operations/Upgrading_Apps_Deployed_to...) are also guaranteed.

Additionally, the guidelines presented in this section assume that:

- The "Main Load Balancer and Reverse Proxy" has a **default rule** pointing to the OutSystems Platform Server for application modules that have no explicit rules defined.

- The "Main Load Balancer and Reverse Proxy" will have **specific rewrite rules** for the OutSystems applications modules deployed in the selected container-based hosting technology.
Although these steps can be performed manually, they should be performed automatically by using the "Container Run" extensibility hook provided by the "Automated Mode" in the Deployment Zone configuration.

Adapt the provided instructions as needed to your specific network infrastructure setup.

Deployment strategies

The provided guidelines for achieving no-downtime deployments in different hosting technologies follow one of these two common deployment strategies:

**Blue-Green**

The blue-green deployment strategy is a deployment strategy where we have 2 identical environments, a live one (blue) and an idle one (green), and we can test the green environment separately without impacting the blue one. After testing, we switch traffic from the blue environment to the green one.

This strategy is ideal for migrating from "Classic Virtual Machines" to container-based hosting technologies, thus minimizing the impact.
Rolling Update

The rolling update strategy is a deployment strategy where the application is updated in a incremental fashion. Each container with the new application version is launched, replacing the old version. This process is done incrementally for each replica.

This strategy optimizes resource usage and it's ideal for version updates.
See Also

No-downtime deployments with Pivotal Cloud Foundry (PCF)
No-downtime deployments with Docker
No-downtime deployments with Amazon ECS
No-downtime deployments with Azure

https://success.outsystems.com/Support/Enterprise_Customers/Maintenance_and_Operations/Upgrading_Apps_Deployed_to…
Updated: Mon, 04 Nov 2019 06:35:13 GMT