OutSystems Upgrade

Last Update: 06/2019

Upgrading to a new OutSystems version is a process that requires preparation since it will impact all your applications.

To perform an upgrade and understand the changes you need to make in your applications to make them fully functional in the new version you need to follow a four-step process containing, Analysis and Preparation, Planning, Execution, and Testing.

1 Analysis and Preparation

- Read the Release Notes and Security fixes (e.g., Platform Server 10.0, 11.0, Lifetime)
- Read about side effects and breaking changes on applications, in the new version (e.g., 10.0, 11.0)
- Evaluate Impact.

2 Planning

- How long will the code fixing take?
- How long will it take to publish all applications?
- Make sure that your team is involved (Development Team and Test team will be needed)
- Beware that an upgrade affects the release numbering.

If you are a cloud customer, please contact OutSystems Support to provide your Upgrade plan.

3 Execution

- In the Installation Checklist (e.g., 10.0, 11.0), select "Upgrade to a new Major Release", and follow the instructions

If you are a cloud customer, OutSystems will perform the Upgrade for you, based on the plan you provided.

- Proceed with the code fixing.

4 Testing

- Create and run tests to assure that your applications hot-points are not affected by the upgrade.

Preparing and Planning an OutSystems Upgrade

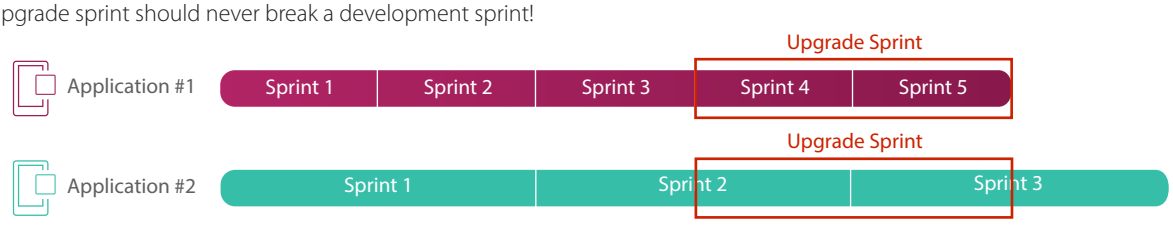
Before starting your Upgrade, you must keep the following in mind.

What to Consider?

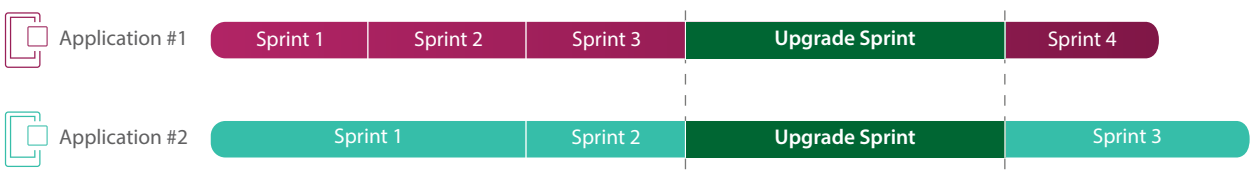
When performing an upgrade, you need to consider that:

- All your applications will have a new version after the upgrade.
- The upgrade can introduce breaking changes that require fixing.
- You must perform the upgrade in a way that is aligned with your release cycle.

An upgrade sprint should never break a development sprint!



An upgrade should always be between development sprints. (If you have teams at a different speed, make the necessary adjustments)



How to prepare it?

- Prepare your applications for a full staging from Development all the way to Production.

  This means that when stabilizing the code in Development, publish full solutions with all dependencies until no errors are reported.

  (!) Keep in mind that a consumer application will fail to publish until all its producers are fully updated.
- Have a test battery and test team ready to test the applications through all its stages, after the upgrade.

How to plan it?

- Validate the best time to upgrade the Production environment and plan the remaining environment upgrades from there.
- Have in consideration how long it will take to fix and test all your applications.

Environment Upgrade Strategies

To perform an Upgrade, there is the need to select the best strategy according to the existing conditions. **We recommend the LifeTime Upgrade**. However, all other presented strategies are valid.

LifeTime Upgrade

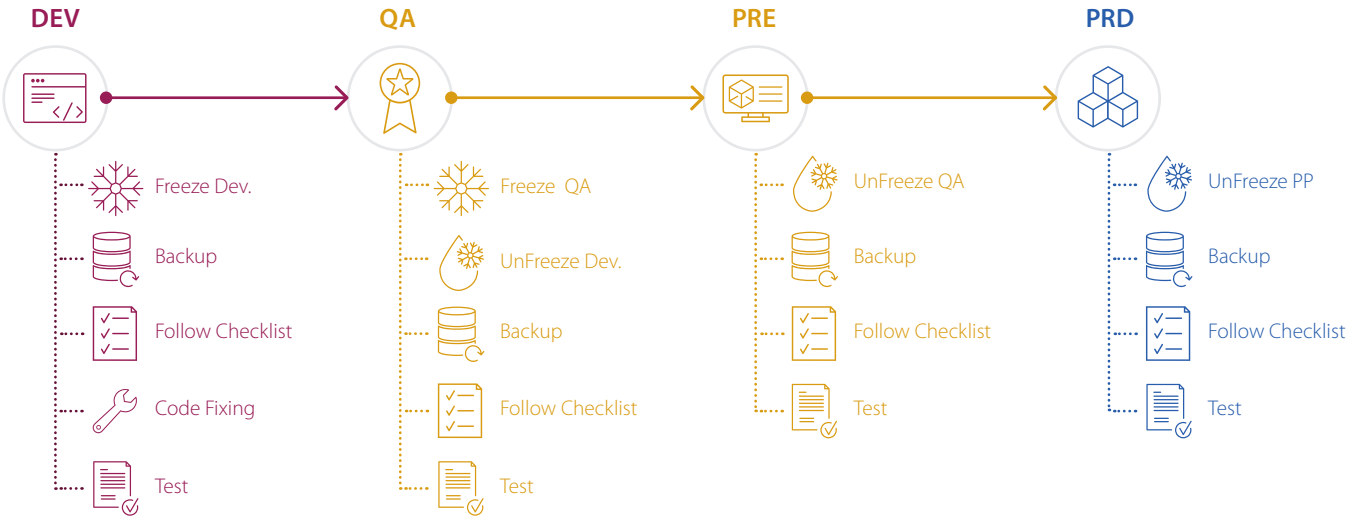
Be sure to have LifeTime in the latest version available to take advantage of the latest improvements.

(!) In case there are custom Lifetime plugins a Code-Based Upgrade must be applied, as these plugins do not exist on any other environment.

With this approach, system and applications on the Development environment are staged in LifeTime to the next environment with a new version of all your applications upgrade.

This approach allows you to roll out applications that you have tested or validated previously.

For (Pre-)Production you should create a custom checklist containing all the requirements and steps to be executed in production, following a downtime or zero-downtime approach.



Solution-Based Upgrade

A solution based upgrade is used a manual approach. Instead of using an automatic staging, like in a LifeTime upgrade, in this case, a Solution is manually staged between environments using Service Center.

Code-Based Upgrade

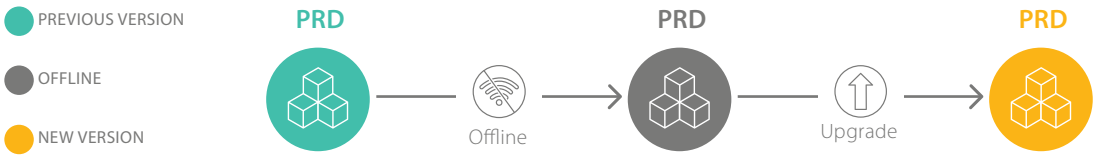
In a Code-Based upgrade, the environments are individually upgraded, and all fixing and testing (solving platform major version breaking changes) in applications must be executed directly in each environment(Dev, QA… PRD).
Use the installation checklist to upgrade your OutSystems platform on each environment.

Go Live Strategies

When doing an Upgrade is vital to have a strategy on how you are going live, after the upgrade. In our case, you can do a Downtime Upgrade or a Zero-Downtime Upgrade.

Downtime Upgrade (Production)

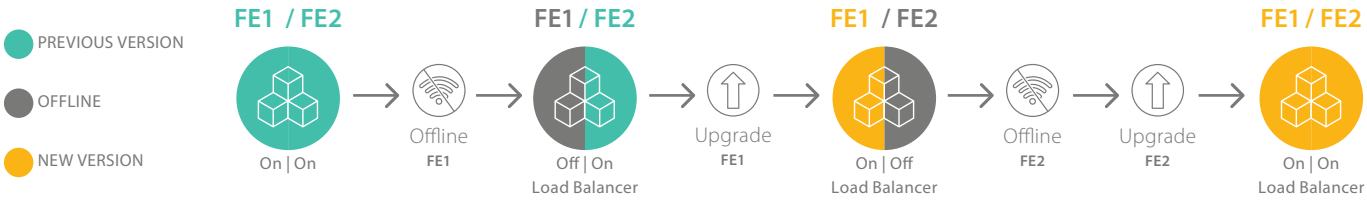
A Downtime approach is the safest way to perform an upgrade and advised when there are no business requirements for zero-downtime. By stopping the production environment and preventing end-users access, the Upgrade process consistency is guaranteed, and a rollback can happen with no data loss.
This strategy can be used both in a LifeTime Upgrade or a Solution-Based Upgrade.



Zero-Downtime Upgrade (Production)

A Zero-Downtime Upgrade is like changing a car engine without stopping the car itself. Which means that if you need to rollback, you will most certainly have data loss.
To perform a Zero-Downtime Upgrade, with success, your environment requires multiple Front Ends in each OutSystems Zone. **If you don't have this feature enabled you can't proceed with this solution.** The Upgrade process is done by:

1. Disabling the Front-end servers in Service Center.
2. Upgrade the platform in the controller node.
3. In the load balancer disconnect half of front-end servers in each zone.
4. Upgrade those servers and then enable them in Service Center.
5. Switch the active Front-End servers in the Load Balancer to the upgraded ones.
6. Upgrade the remaining Front-End servers.
7. Finally, reconnect all servers to the Load Balancer.



+1 404 719 5100
info@outsystems.com
www.outsystems.com