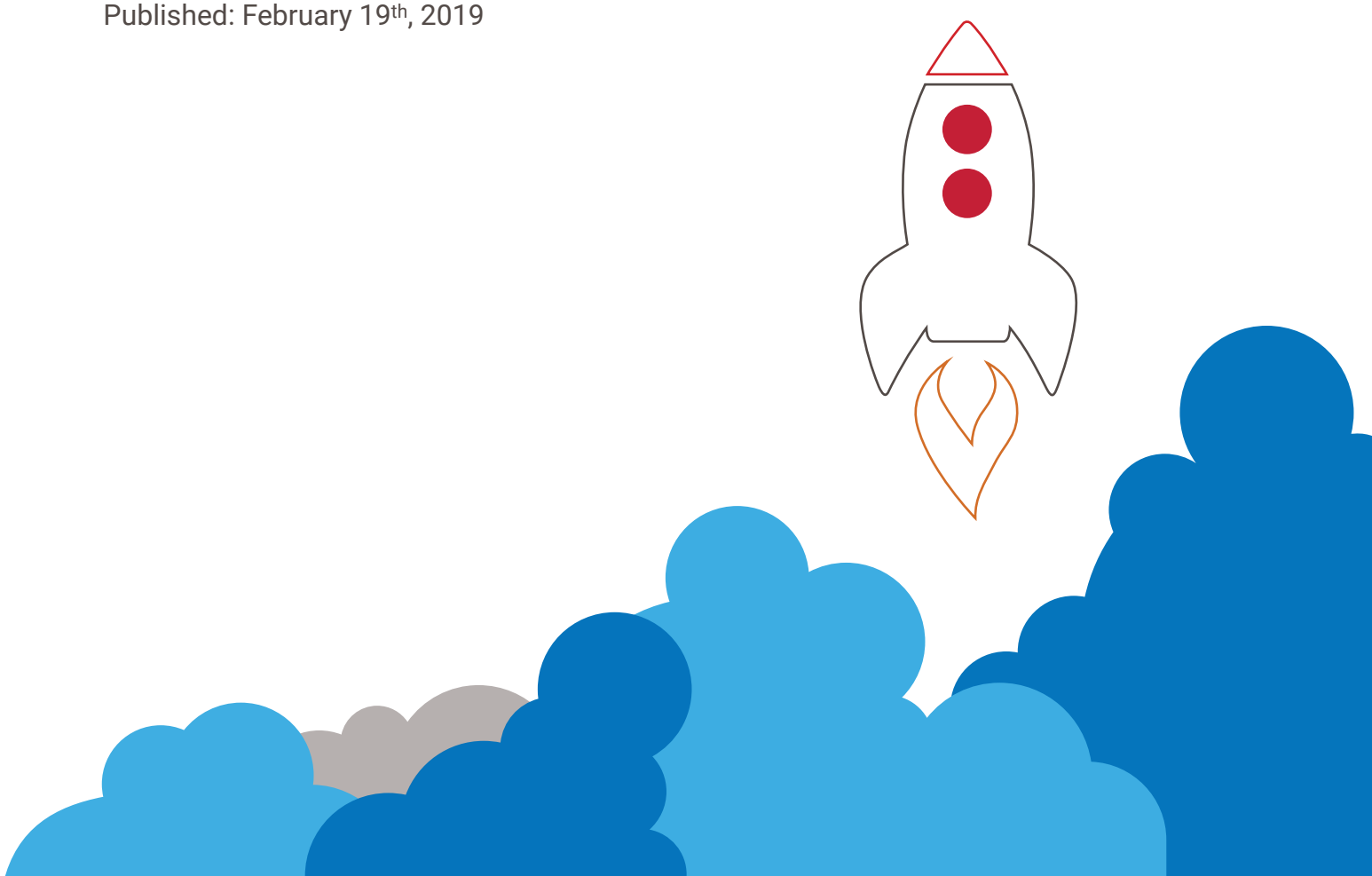
urban{code}

Upgrading to UrbanCode Deploy 7

Published: February 19th, 2019



{Contents}

{Introduction}

This white paper describes the general activities required to upgrade UrbanCode Deploy from version 6 (all releases) to version 7. Although you can use this paper as a template for upgrades, organizations vary in size, complexity, infrastructure, and other factors, and so a one-size-fits-all upgrade solution is not practical. Therefore, our objective is to guide customers in the creation and execution of upgrade plans suitable for their organizations.

The intended audience of this paper is the IT staff who owns or manages UrbanCode Deploy for their organization and who is primarily responsible for planning and executing the product upgrades. Other IT teams that play ancillary roles in product upgrades, such as DBAs and storage teams, might also find useful information in this paper.

We will describe the overall upgrade in four phases:

1. Planning
2. Creating a test environment
3. Upgrading the test environment
4. Upgrading the production environment

By completing these phases, you build confidence for the production upgrade and help minimize impact on production users. For some organizations, these phases might even be required parts of the production change-control process.

Here is a typical "happy path" you might follow based on the phases that this paper covers:

1. Clone your UrbanCode Deploy production environment to create a test environment.
2. Validate the test environment.
3. Upgrade the test environment.
4. Roll back the upgrade in the test environment.
5. Upgrade the test environment.
6. Validate the test environment.
7. Upgrade the production environment.
8. Upgrade and convert production agents to use the new WebSockets communication protocol.
9. Validate the production environment.

These procedures depend on the overall UrbanCode Deploy architecture or infrastructure remaining the same during the upgrade. If these items must change, for best results complete those changes outside the scope of the UrbanCode Deploy upgrade. For example, do not change the following aspects of your environment immediately prior or during the upgrade:

- UrbanCode Deploy high-availability configuration or lack thereof
- Database vendor and version
- Underlying OS vendor and version
- Rational License Key Servers

{Phase 1: Planning}

An upgrade plan is one of the most important tools available during upgrades. Creating a plan helps you answer the "who, what, when, where, why, and how" questions related to the upgrade, and it helps reduce unanticipated issues during or after the production upgrade.

1.1 Review help available from the UrbanCode team

The UrbanCode team offers various levels of upgrade assistance to customers. Decide what levels of help your organization needs and wants. Your account manager can provide contact information for these organizations.

Help level	Help Available
L2 Support	• Advise on upgrade plan • Primary contact for support ticket • Review APARs and patches for the old and new versions • Enable customer on the process to quickly engage L2 Support if product issues arise during the upgrade, allowing for real-time troubleshooting and timely resolution • Socialize planned upgrade activity with key stakeholders (for example, L3, development, and client advocates). This help ensures that the product team is prepared to engage with customers quickly, if a product issue arises during the upgrade
L3 Support	• Advise on upgrade plan • Assist with reviewing APARs and patches as needed • Resolve issues that are found during upgrade testing or after the upgrade
Client advocate	• Advise on upgrade plan • Extra communication channel for upgrade questions, concerns, and issues • Facilitate and document upgrade meetings • Communicate upgrade completions to management (client advocacy, product teams, and development)
Lab services (fee based)	• Assess and plan upgrades • Application deployment configuration • 3rd-party migration and integration • Staff augmentation and transitional support, including hands-on execution of upgrades

1.2 Open a preemptive support ticket

The ticket serves as a placeholder for unanticipated issues during the upgrade. The UrbanCode support team can also use the ticket to provide feedback on the customer's upgrade plan.

1.3 Review the product upgrade documentation

See the upgrade information in IBM Knowledge Center:
https://www.ibm.com/support/knowledgecenter/SS4GSP_7.0.0/com.ibm.udeploy.doc/topics/c_node_upgrading.html

See the information in "Getting Started – Upgrade Notes":
https://developer.ibm.com/urbancode/products/urbancode-deploy/whats-new/whats-new-urbancode-deploy-7-0/getting-started/

Pay special attention to the notes about upgrading from or to specific versions. Consider the version you are upgrading from and to. For example, consider these types of comments:

Starting in version	Comments
6.1.1.5	Some UrbanCode Deploy server files are reorganized into an appdata directory
6.2.2.0	UrbanCode Deploy servers and relays require Java version 8
6.2.3.0	UrbanCode Deploy servers and relays must be upgraded at the same time
6.2.7.1	API-breaking changes have been made to several REST endpoints
7.0.0.0	New server-agent communication protocols are introduced

1.4 Consider your organization's processes and requirements

Organizations vary in size, complexity, infrastructure, and other factors. Consider several questions when you start to plan to upgrade UrbanCode Deploy. Working these items into the upgrade plan early can help streamline the upgrade process.
Production change control

- Is a change request or evidence of testing required before upgrading production?
- What teams are involved? For example, are DBAs, Midrange and Linux, QA, security, and storage involved?

Infrastructure logistics

- Who owns or controls relay machines? Agent machines?
- Will agents be upgraded through the product UI or scripts on the agent computers?
- Are differences between test and prod environments unavoidable? For example, do the following situations exist?
  - Does the prod environment span multiple datacenters?
  - Does the prod UrbanCode Deploy instance cross firewalls or networks to deploy to agents in the test environment?

1.5 Choose a relay and agent upgrade strategy

Follow these guidelines for upgrading relays and agents:

- Upgrade relays at the same time (that is, in same change window) as the server.
- You do not have to upgrade agents at the same time (that is, in same change window) as the server, but follow the server's minimum recommended and minimum required agent version.
- Do not connect a new relay or agent version to an old server version. These connections are not supported

1.6 Decide whether to convert Java Message Service (JMS) agents to web agents

UrbanCode Deploy 7 introduces a new server-agent communication protocol that is based on WebSockets instead of JMS. The new communication protocol brings significant improvements in stability, scalability, and performance to the product. Therefore, we strongly recommend that customers take the opportunity to convert to web agents.

UrbanCode Deploy has an approximate capacity of 15,000 JMS agents for single-instance servers, or 8,000 JMS agents for highly-available server clusters. Organizations that must connect more than 15,000 or 8,000 agents to UrbanCode Deploy should convert JMS agents to web agents to take advantage of an approximate capacity of 100,000 agents.

See the following blog for more information:
https://developer.ibm.com/urbancode/2018/11/25/web-agents-bring-greater-scalability-and-stability-to-urbancode-deploy/

1.7 Slim down UrbanCode Deploy

Over time, the database and file-system storage footprints can grow considerably. Data growth can directly affect upgrade and rollback time. Therefore for best results, consider the following measures to reduce your stored data.

Set a version cleanup policy

Avoid retaining all component versions indefinitely, especially if you configure the product to store artifacts in Code Station. UrbanCode Deploy offers automatic version cleanup settings, and you can use the REST API or udclient CLI to clean up specific versions.

Clean up the audit log

The audit log feature records all interactions with the product and stores this data in the database. This data grows continuously with normal usage of the product, and the data growth can eventually affect upgrade and rollback time. Use the audit log cleanup settings to avoid this.

1.8 Create a validation plan

Define two levels of validation: smoke test and full validation.

During validation, consider known issues such as the number of offline agents and specific errors in the server log.

Scripting or otherwise automating validation steps where possible can help make validation consistent, efficient, and repeatable. Validation automation also offers the added benefit of being reusable during subsequent upgrades.

Smoke test

The purpose of a smoke test is simply to check whether UrbanCode Deploy is available and that basic functionality is working. It should be fast and shallow, or trivial. Consider these examples:

- Log into the UI and click through the tabs at the top of the page.
- Create a system property.
- Run a "hello world" generic process.

For HA installations, consider performing the smoke test on each server individually. The following example ensures that two HA servers have read/write access to the database and CodeStation file system:

1. Stop UrbanCode Deploy server B.
2. With server A running, create any component version, and push a file into it.
3. Start server B, and stop server A.
4. Log into the UI, download the file from the component version, and then delete the component version.
5. Start server A, and stop server B.
6. Log into the UI, and verify that the component version is gone.

Full validation

Full validation takes longer than the smoke test, because it more thoroughly exercises all applicable templates, processes, and so on. Consider running real "hello world" deployments of critical applications. Also cover inbound integrations from third-party tools, such as CMDBs and Jenkins, and custom applications, reports, scripts, and tools that invoke UrbanCode Deploy's REST API or udclient CLI.

As part of full validation, consider the agent upgrade strategy that you plan to use. If you intend to upgrade all agents at the same time as the server, then full validation does not require testing the old agent version. However, if agents are upgraded after the server upgrade, you might want to validate both the old and new agent versions.

Make sure that the definition of full validation includes the criteria for calling an upgrade successful versus deciding to roll it back.

{Phase 2: Create a test environment}

A test environment is the ideal place to practice upgrading and rolling back upgrades. Working in a test environment helps build confidence in the production upgrade without affecting real production users.

2.1 Create a test environment for upgrade validation

The goal of this step is to create a production-like UrbanCode Deploy test environment where the customer can validate and refine the upgrade and rollback process.
See the following technote for instructions:
http://www-01.ibm.com/support/docview.wss?uid=swg21694427

2.2 Validate the test environment

Before practicing an upgrade on the test UrbanCode Deploy environment, complete a full validation of the environment to ensure that everything works as expected. If any issues arise prior to upgrading the test environment, then upgrading might lead to confusion about the root cause of the issues and lower confidence for the production upgrade.

{Phase 3: Upgrade UrbanCode Deploy in the test environment}

3.1 Prepare a backup

For servers, follow these general backup steps:

- Cancel running processes and stop the UrbanCode Deploy server.
- Back up the database. For example, export the schema to a file.
- Back up the appdata directory. For example, copy the directory, or identify an NFS snapshot.
- Back up the local server directory.

For relays and agents, simply back up the directory where the software was installed.

3.2 Upgrade

- Run the upgrade script on the UrbanCode Deploy server.
- Run the upgrade script on the relays.
- Run a smoke test.
- Upgrade agents according to your agent upgrade strategy.

Consider recording the duration of the upgrade to refine the production upgrade plan. The more production-like your test environment is, the more useful this record is.

3.3 Roll back the upgrade

The purpose of this planned rollback is to build confidence that the production upgrade can be backed out in case of some unforeseen issue. Ideally, a rollback will not be required after the production upgrade. For relays and agents, simply restore the directory that you backed up earlier.

For servers, follow these general steps to roll back the upgrade:

- Stop the UrbanCode Deploy server.
- Restore the database. For example, drop the schema and import the schema from the backup file.
- Restore the appdata directory. For example, copy the NFS snapshot or manual backup into the directory.
- Restore the local server directory.
- Start the UrbanCode Deploy server, and run a smoke test.

3.4 Upgrade without the rollback

Follow the same procedure as in step 3.2. However, do not complete a planned rollback after this upgrade.

3.5 Full validation

Complete the full validation plan that you defined earlier. Record evidence of testing if required, such as screenshots and links to successful process executions. Report any issues to the UrbanCode team so you can obtain any patches, test fixes, or workarounds prior to the production upgrade.

{Phase 4: Upgrade UrbanCode Deploy in the production environment}

4.1 Notify UrbanCode support that the upgrade is approved and scheduled

Support should already be aware of your planned upgrade, but we recommend notifying them when the decision to upgrade is final. This can help minimize any delays in engaging the teams for support.

4.2 Prepare a backup

For servers, follow these general backup steps:

- Cancel running processes, and stop the UrbanCode Deploy server.
- Back up the database. For example, export the schema to a file.
- Back up the appdata directory. For example, copy the directory, or identify an NFS snapshot.
- Back up the local server directory.

For relays and agents, simply back up the directory where the software was installed.

4.3 Upgrade the server and relays

- Run the upgrade script on the UrbanCode Deploy server.
- Run the upgrade script on the relays.
- Run a smoke test.

4.4 Run the upgrade at the DR site, if applicable

If your production environment spans multiple datacenters, such as for disaster recovery, a single UrbanCode Deploy instance might include servers installed in multiple datacenters. In that scenario, any cold or disaster recovery servers that run UrbanCode Deploy must be upgraded.

The procedure for this might vary from organization to organization, but might include these items:

- Fail over the UrbanCode Deploy database, shared storage, DNS and load balancers to the disaster recovery site.
- Run the upgrade script on servers in the disaster recovery site.
- Start UrbanCode Deploy, run a smoke test, and stop the application.
- Fail over the UrbanCode Deploy database, shared storage, DNS and load balancers back to the primary site.

4.5 Complete a full validation

Repeat the same validation plan that you carried out on the test environment. Report any unanticipated issues to the UrbanCode team.

Decide whether the upgrade meets the criteria to be deemed successful; if not, inform the UrbanCode team and roll back the upgrade by using the same procedure you practiced on the test instance.

4.6 Upgrade agents

Upgrade agents according to your agent upgrade strategy.

{Summary}

To help ensure success, you can approach upgrading UrbanCode Deploy in four phases: plan, create a test environment, upgrade UrbanCode Deploy in the test environment, and upgrade UrbanCode Deploy in the production environment. The benefits and some key considerations for creating an upgrade plan are clear, and having a plan further enables success. The best practice of creating a test environment for practicing upgrades and rollbacks furthers positive upgrade results. A high-level approach to upgrading UrbanCode Deploy servers, relays, and agents, including some considerations related to more complicated organizations, such as high availability and disaster recovery installations, completes the approach to upgrading UrbanCode Deploy.

The authors hope you found this paper useful in approaching your upgrade. If you have questions or feedback, please contact your account manager, support representative, or client advocate.

©Copyright HCL Technologies Limited 2019.

This information was developed for products and services offered in the US.

HCL may not offer the products, services, or features discussed in this document in other countries. Consult your local HCL representative for information on the products and services currently available in your area. Any reference to an HCL product, program, or service is not intended to state or imply that only that HCL product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any HCL intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non- product, program, or service.

HCL may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
HCL
330 Potrero Ave. Sunnyvale, CA 94085 USA
Attention: Office of the General Counsel

For license inquiries regarding double-byte character set (DBCS) information, contact the Intellectual Property Department in your country or send inquiries, in writing, to:
HCL
330 Potrero Ave. Sunnyvale, CA 94085 USA
Attention: Office of the General Counsel

HCL TECHNOLOGIES LTD. PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. HCL may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-HCL websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this product and use of those websites is at your own risk.

HCL may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-HCL products was obtained from the suppliers of those products, their published announcements or other publicly available sources. has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-HCL products. Questions on the capabilities of non-products should be addressed to the suppliers of those products.

Statements regarding HCL's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Trademarks
HCL, the HCL logo, and hcl.com are trademarks or registered trademarks of HCL Technologies Ltd., registered in many jurisdictions worldwide. UrbanCode Deploy is a trademark of IBM Corporation, registered in many jurisdictions, and is used under license. Other product and service names might be trademarks of or other companies.

urban{code} | HCL