IBM uDeploy® Introduction

5.0.0

IBM uDeploy Introduction: 5.0.0

Publication date January 2013

Document Number: 5.0.0.1i

About This Book

This book describes how to use UrbanCode, an IBM Company's IBM uDeploy product and is intended for all users.

Product Documentation

IBM uDeploy's online Help is installed along with the product software and can be accessed from the product's web-based user interface. Product books are available in PDF format at UrbanCode, an IBM Company's Documentation portal: http://docs.urbancode.com/, and are included with the installation package.

In addition to this book, the books comprising the documentation suite include:

Table 1. Product Documentation Suite

Book	Description
IBM uDeploy User Guide	Describes how to use the product; contains chapters for core features, such as components, applications, and resources.
IBM uDeploy Administration Guide	Describes how to install and configure the product, and how to set up security.
IBM uDeploy Introduction	Provides an overview of the product's significant features and describes its architecture.

Product Support

The UrbanCode, an IBM Company Support portal, http://support.urbancode.com/, provides information that can address any of your questions about the product. The portal enables you to:

- review product FAQs
- download patches
- view release notes that contain last-minute product information
- review product availability and compatibility information
- access white papers and product demonstrations

Document Conventions

This book uses the following special conventions:.

- `Program listings, code fragments, and literal examples are presented in this typeface.`
- Product navigation instructions are provided like this:

  `Home > Components > [selected component] > Versions > [selected version] > Add a Status [button]`

This example, which explains how to add a status to a component version, means: from the IBM uDeploy home page click the Components tab (which displays the Components pane); select a component (which displays a pane with information for the selected component); click the Versions tab (which displays a pane with information about the selected version); and click the Add a Status button.

- User interface objects, such as field and button names, are displayed with initial Capital Letters.
- Variable text in path names or user interface objects is displayed in *italic* text.
- Information you are supposed to enter is `displayed in this format`.

Introduction

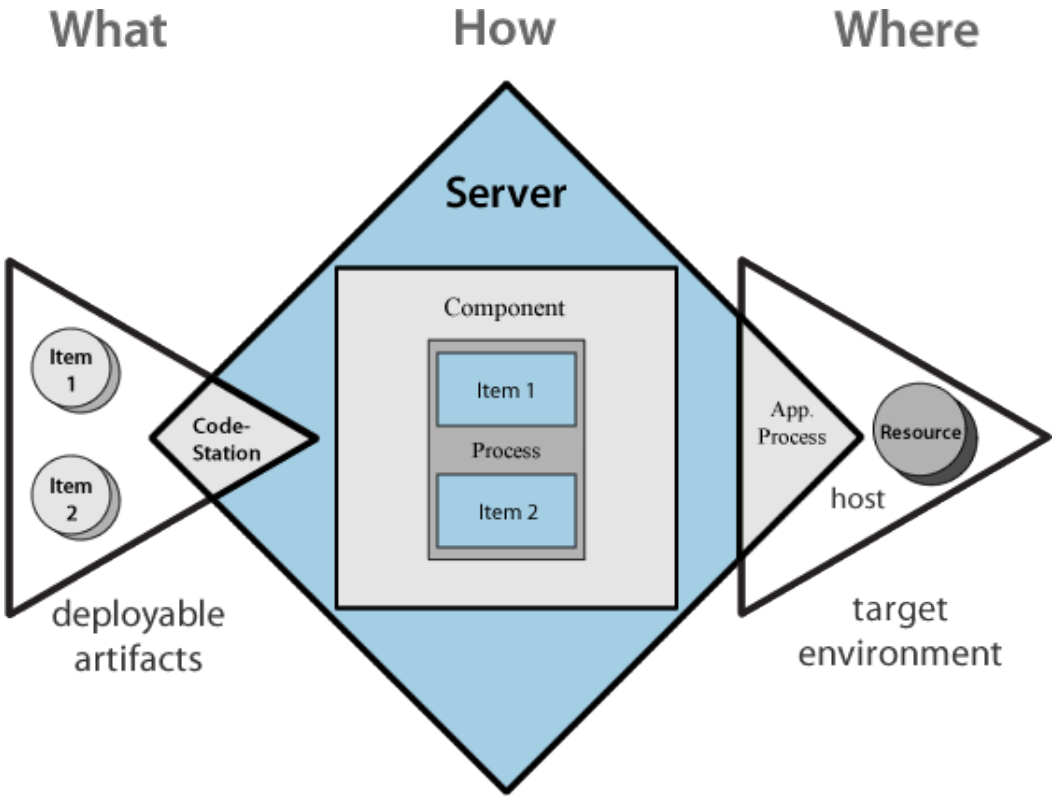
Overview

At its base, software deployment is a simple concept that sometimes gets obscured by jargon. A deployment is the process of moving software (broadly defined) through various preproduction stages to final production. Typically, each stage represents a step of higher criticality, such as quality assurance to production. Complexity arises from the sheer volume of things deployed, the number and variety of deployment targets, constantly-decreasing deployment cycles, and the ever-increasing rate of technological change. While virtualization provides some relief to the process, it also—perhaps paradoxically—increases the challenge with its exponential growth of deployment targets.

IBM uDeploy helps you meet the challenge by providing tools that improve deployment speeds while simultaneously improving their reliability. IBM uDeploy's release automation tools provide complete visibility into *n*-tiered deployments, enabling you to model processes that orchestrate complex deployments across every environment and approval gate. IBM uDeploy's drag-and-drop design tools decrease design-time by making it easy to visualize the end-to-end deployment process and develop the big picture—the *What*, *How*, and *Where* of the deployment workflow:

- **What**: the deployable items—binaries, static content, middleware updates, database changes and configurations, and anything else associated with the software—that IBM uDeploy delivers to target destinations.
- **How**: refers to combining deployable items with processes to create components, and designing applications that coordinate and orchestrate multi-component deployments.
- **Where**: the target destination's hosts and environments—IBM uDeploy can scale to any environment.

Figure 1. Deployment Process



In IBM uDeploy, deployable items are combined into logical groupings called components. Components are deployed by component processes which consist of user-configured steps, many taken from integrations with third-party tools called plug-ins. Multi-component deployments are handled by user-assembled *applications*.

IBM uDeploy represents deployment targets by what it calls *resources*. Resources—databases, servers, and so on—reside on hosts. Complex deployments can contain numerous components that target multiple hosts. Deployments are managed by agents residing on the hosts. Components can also remain independent of one another, which enables incremental or targeted deployments. Of course, you can model your components as you see fit—IBM uDeploy is flexible and works the way you work.

Server

The IBM uDeploy server is a standalone server that provides IBM uDeploy's core services such as the user interface, component and application configuration tools, workflow engine, and security services, among others. Many services are REST-based.

IBM uDeploy supports cross-network deployments with relay servers. Relay servers enable network-to-network communications.

Agents

An agent is a lightweight process that runs on a host and communicates with the IBM uDeploy server. Agents manage the resources that are the actual deployment targets. Each machine participating in a deployment usually has an agent installed on it. When not performing deployments, agents run in the background with minimal overhead. See the section called “Resources”.

Repository

The IBM uDeploy-supplied artifact repository, CodeStation, provides secure and tamper-proof storage. It tracks artifact versions as they change and maintains an archive for each artifact. Associations between repository files and components are built-in and automatic.

Security

In IBM uDeploy’s role-based security system, users are assigned roles, and role-permissions are assigned to things such as projects, build configurations, and other resources. For example, a developer may be permitted to build a project, but only view non-project related material. See ???.

Components

Understanding how IBM uDeploy uses the term component is critical to understanding IBM uDeploy. Components represent deployable items along with user-defined processes that operate on them, usually by deploying them. Deployable items--also called artifacts--can be files, images, databases, configuration materials, or anything else associated with a software project. Components have *versions* which are used to ensure that proper component instances get deployed.

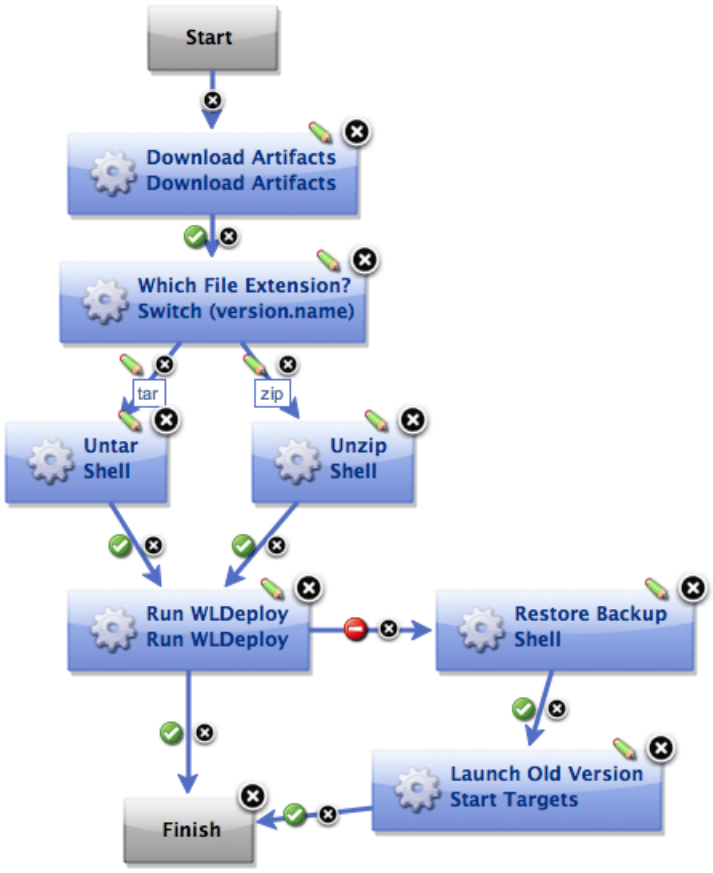
Artifacts can come from a number of sources: file systems, build servers such as AnthillPro, source version control systems, Maven repositories, as well as many others. When you create a component, you identify the source and define how the artifacts will be brought into IBM uDeploy. If the source is Subversion, for example, you specify the Subversion repository containing the artifacts. Each component represents artifacts from a single source.

Component Processes

A component process is a series of user-defined steps that operate on a component's artifacts. Each component has at least one process defined for it and can have several. A component process can be as

simple as a single step or contain numerous steps and relationships. The switch step, for instance, enables you to create conditional processes. You might, say, take artifacts from a source like an AnthillPro project and map the ones that get deployed to an HTTP server into one component; those that get deployed to a J2EE container to another; and those that get deployed to a database to yet another. Or, to take another example, a single-component deployment might consist of two processes: the first moves component files to a server on Friday night (a lengthy operation), while the second deploys the files Saturday morning.

Figure 2. Process Editor with a Component Process Containing a Switch Step



Component processes are created with IBM uDeploy's process editor. The process editor is visual drag-and-drop editor that enables you to drag process steps onto the design space and configure them as you go. As additional steps are placed, you visually define their relationships with one another. Process steps are selected from a menu of standard steps that replace typical deployment scripts and manual processes. IBM uDeploy provides steps for several utility processes, such as inventory management, and workflow control. Additional process steps are provided by plug-ins. Plug-ins provide integration with common deployment tools and application servers, such as WebSphere, Microsoft IIS, and many others. Out-of-the-box, IBM uDeploy provides plug-ins for many common processes, such as downloading and uploading artifacts, and retrieving environment information. A component process can have steps from more than one plug-in.

A component process is defined for a specific component. A component can have more than one process defined for it, but each component requires at least one process.

For example, deploying a J2EE EAR file to WebSphere server typically consists of the following operations:

1. transfer the EAR file to the target machine

2. stop the WebSphere server instance

3. invoke wsAdmin with deployment properties

4. start the WebSphere instance

5. verify that the deployment succeeded by accessing a specified URL

The WebSphere plug-in provides a configurable process step for each operation.

A frequently used component process can be saved as a template and applied later to new components.

Component processes are executed by IBM uDeploy agents running on hosts. One instance of a component process is invoked for each resource mapped to a component in the target environment, see the section called “Resources”.

Plug-ins

Plug-ins provide basic processing functions as well as integration with third-party tools. IBM uDeploy ships with plug-ins for several common deployment processes, and others are readily available for a wide variety of tools, such as middleware tools, databases, servers, and other deployment targets.

Third-party tools exhibit wide and varied functions, of course. Plug-in integration is achieved by breaking down a tool's functions into simple, discrete steps that invoke a specific behavior. A plug-in step might invoke a tool, or invoke different functions in a tool, such as extracting or inserting some type of data.

When you use plug-ins to create a component process, you can use steps from several plug-ins and configure the steps as you go. For example, you might create a process using a plug-in for a source control tool that deploys a component to a middleware server, and another plug-in to configure a step that removes the component from the server.

A component process that contains a plug-in step requires an agent. Unless the agent needs to interact with the host's file system or system processes, the agent does not have to be on the same host as the target resource.

IBM uDeploy enables you to download and install numerous component plug-ins. UrbanCode, an IBM Company does not charge any additional fees for plug-ins. The plug-in system is open and extensible--plug-ins can be written in any language.

Component Versions and the CodeStation Repository

After defining a component's source and processes, you import its artifacts into IBM uDeploy's artifact repository CodeStation. Artifacts can be imported automatically or manually. By default, a complete copy of an artifact's content is imported into CodeStation (the original artifacts are untouched). This provides several benefits, such as tamper-proof storage, and the ability to review and validate artifacts with IBM uDeploy's user interface. But if you have storage concerns or use a tool like Maven, you can limit CodeStation to using references to the artifacts instead of actually copying them.

Each time a component is imported, including the first time, it is versioned. Versions can be assigned automatically by IBM uDeploy, applied manually, or come from a build server. Every time a component's artifacts are modified and reimported, a new version of the component is created. So a component might have several versions in CodeStation and each version will be unique.

A version can be full or incremental. A full version contains all component artifacts; an incremental version only contains artifacts modified since the previous version was created.

Applications

An applications is the mechanism that initiate component deployments; they bring together components with their deployment targets, and orchestrate multi-component deployments.

Application Process

When you create an application, you identify the included components and define an application process. Application processes, like component processes, are created with the process editor. IBM uDeploy provides several common process steps, otherwise application processes are assembled from processes defined for their associated components.

Application processes can run manually, automatically on some trigger condition, or on a user-defined schedule. When a component has several processes defined for it, the application determines which ones are executed and in which order. For instance, an *n*-tiered application might have a web tier, a middleware tier, and a database tier. And, continuing the example, the database tier must be updated before the other two, which are then deployed concurrently. An application can orchestrate the entire process, including putting servers on- and off-line for load-balancing as required.

When an application process executes, it interacts with a specific environment. An environment is a collection of one or more resources. At least one environment must be associated with the application before the process can be executed. Application processes are environment agnostic; processes can be designed independently of any particular environment. This enables a single application to interact with separate environments, such as QA, or production. To use the same application process with multiple environments (a typical scenario), you associate each environment with the application and execute the process separately for each one.

In addition to deployments, several other common processes are available, such as rolling-back deployments. IBM uDeploy tracks the history of each component version, which enables application processes to restore environments to any desired point.

Environments

An environment is a user-defined collection of resources that host an application. Environments are typically modeled on some stage of the software project life cycle, such as development, QA, or production. A resource is a deployment target, such as a database or J2EE container. Resources are usually found on the same host where the agent that manages them is located. A host can be a physical machine, virtual machine, or be cloud-based.

Environments can have different topologies--for example: an environment can consist of a single machine; be spread over several machines; or spread over clusters of machines. Environments are application scoped. Although multi-tenant machines can be the target of multiple applications, experience has shown that most IT organizations use application-specific environments. Additionally, approvals are generally scoped to environments.

IBM uDeploy maintains an inventory of every artifact deployed to each environment and tracks the differences between them.

Snapshots

A snapshot is a collection of specific component versions, usually versions that are known to work together. Typically, a snapshot is generated in an uncontrolled environment--meaning one without— approvals. When a snapshot is created, a picture of the application's current state is captured. As an application moves through different environments, snapshots can ensure that proper component versions are used.

Snapshots help manage complex deployments—deployments with multiple tiers and development teams. For example, after testing and confirming that team A's component works with teams B's, a snapshot can be taken. Then, as development progresses, additional snapshots can be taken and used to model the effort and drive the entire deployment, coordinating versions, configurations, and processes.

Agents

An agent is a process that runs on target host and communicates with the IBM uDeploy server. Agents are integral to IBM uDeploy's client/server architecture. Agents perform the actual work of deploying components and so relieves the server from the task, making large deployments involving thousands of targets possible.

Typically, an agent runs on the same host where the resources it handles are located. A single agent can handle all resources on its host. If a host has several resources, an agent process is invoked separately for each resource. For example, a test environment might contain a single web server, a single middleware server, and a single database server all running on the same host (machine). A deployment to this environment might have one agent and three separate resources.

Depending on the number of hosts in an environment, a deployment might require a large number of agents. Agents are unobtrusive and secure. Agent communications use SSL encryption and mutual key-based authentication. For added security, agents do not listen to ports, but open direct connections to the server instead.

Resources

A resource is a user-defined construct based on IBM uDeploy's architectural model. Resources aid bookkeeping; inventory is tracked for resources. Resources are created and managed through the user interface.

A resource represents a deployment target--a physical machine, virtual machine, database, J2EE container, and so on. Components are deployed to resources by agents (which are physical processes). Resources generally reside on the same host where its managing agent runs. A host can have more than one resource. If an agent is configured to handle multiple resources, a separate agent process is invoked for each one.

A resource can represent a physical machine, which is the simplest configuration, or a specific target on a machine, such as a database or server. So a host (machine) can have several resources represented on it. In addition, a resource can represent a process distributed over several physical or virtual machines. Finally, environments consist of resources.

To perform a deployment, at least one resource must be defined and (usually) at least one agent. ("Usually" because trivial deployments can be done without an agent.) Typically, each host in a participating environment has an agent running on it to handle the resources located there.

A proxy resource is a resource effected by an agent on a host other than the one where the resource is located. If an agent does not require direct interaction with the file system or with process management on the host, a proxy resource can be used. When a deployment needs to interact with a service exposed on the network (a database or J2EE server, for instance), the interaction can happen from any machine that has access to the networked service.

Resource Groups

A resource group is a logical collection of resources. Resource groups enable collections of resources to be easily reused. Resource groups can manage multi-tenant scenarios, for example, in which several machines share the same resources.

Architecture Overview

IBM uDeploy architecture consists of a service tier and a data tier. The service tier has a central server that provides a web server front-end and core services, such as workflow, agent management, deployment, inventory, security, as well as others. A service can be thought of as a self-contained mechanism for hosting a piece of business logic. Services can be consumed by clients\agents or other services. Deployments are orchestrated by the server and performed by agents distributed throughout the network. Most clients use browsers to communicate with the web server via HTTP(S). Most server-agent communication is done via JMS (discussed below) but HTTP(S) is also used as required.

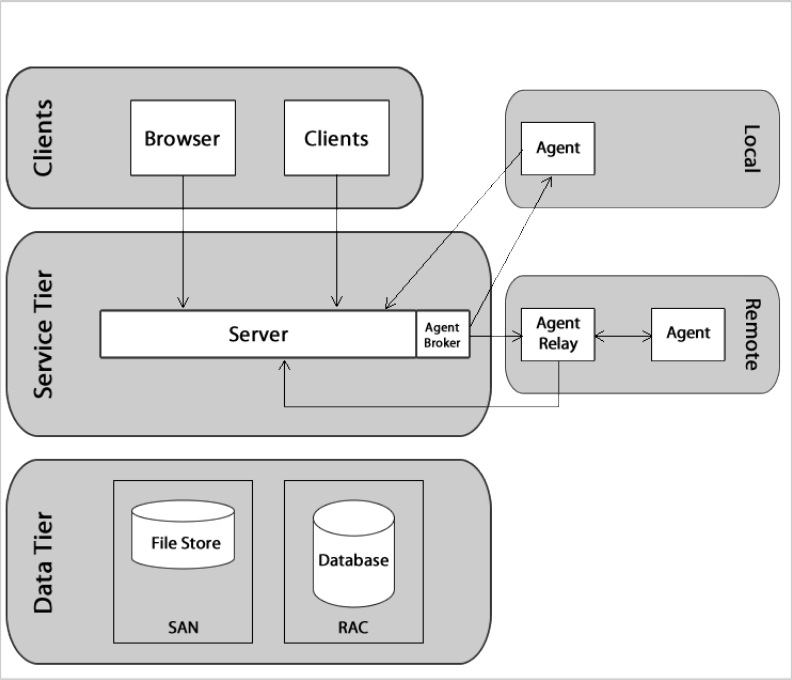
IBM uDeploy uses stateless communications for server-agent communications (JMS-based) as well as client-web service communications. Stateless, as used here, means the server retains little session information between requests and each request contains all the information required to handle it. The server sets-up listening sockets and listens for agents, relays, and users (clients). For added security, agents do not listen on ports. Agents send requests when they are ready to make the transition to a new state.

Server-agent communication is built around transferring—deploying—components. Components can contain any business-meaningful content, such as environment information, configuration data, source, static files, or anything else associated with a software project. Because JMS connections are persistent and not based on a request-response protocol, IBM uDeploy does not have to continually open and close ports, which enables the server to communicate with agents at any time while remaining secure and scalable.

Many IBM uDeploy services are REST-type (representational state transfer). REST-style services are web services that focus on transferring resources over HTTP. A resource can be any business-meaningful piece of data. Resources are transferred by a self-describing format such as XML or JSON (JavaScript Object Notation). The XML and JSON representations typically model resource states at the time of agent/client requests. REST-style services achieve statelessness by ensuring that requests include all the data needed by the server to make a coherent response.

The data tier's relational database stores configuration and run-time data. The data tier's file store—CodeStation—contains log files, artifacts, and other non-structured data objects. Reporting tools can connect directly to the relational database.

Figure 3. Overview



Service Tier

The IBM uDeploy server provides a variety of services, such as: the user interface, component and application configuration tools, workflow engine, and security services among others. The REST-based user interface provides the web-based front-end that is used to create components and fashion workflows; request processes, and manage security and resources, among other things.

When a workflow is requested, many services are used to fulfill the request, which are shown in the following illustration:

Figure 4. Services and Process Workflow

Workflow requests are initiated with the user interface, either the web-based application or the CLI (command line interface).

Table 2. Services

Service	Description
User Interface	Used to create components and fashion workflows, request processes and manage security and resources, among other things. REST-based.
Workflow Engine	Manages workflows—application and component processes. Calls the agent responsible for performing the workflow's current plug-in step. When the workflow is finished, alerts the notification and inventory services. Called by the deploy service. REST-based.
Agent	Tracks installed agents and routes plug-in commands to affected agents. Commands come from plug-in steps. Invoked by the workflow service. REST-based.

Service	Description
Work Item	Operates in tandem with the approval service; provides approver alerts and enables approvers to accept or reject workflows. If a scheduled workflow remains unapproved at run-time, the job fails automatically. REST-based.
Plug-in Manager	IBM uDeploy can interact with virtually any system through its extensible plug-in system; plug-ins provide functions by breaking-down tool features into automated steps. Plug-ins can be configured at design- and run-time. When a plug-in step executes, the controlling agent invokes its run-time process to execute the step. When a new component version is available, the agent compares the current component version and downloads and only new or changed artifacts.
Event	The event service is ubiquitous; it alerts other services as various trigger conditions occur.
Deployment Service	Manages deployments. When a deployment process is requested, invokes the workflow engine to perform the process. Works in tandem with the security service to ensure users have required permissions. REST-based.
Notification Manager	Notifies users about the status of deployments; notifications are sent to approvers if the system is configured with an email server and the user has an email address. Invoked by the workflow manager. REST-based.
Inventory Manager	When a workflow finishes, the inventory manager updates affected inventory records. IBM uDeploy maintains an inventory of every deployed artifact in every environment, which provides complete visibility across environments. REST-type service.
Approval Engine	Enables creation of approval-requiring jobs and approver roles. Works in tandem with the work item service to ensure required approvals are made before scheduled jobs. REST-based.
Security	Controls what users can do and see; maps to organizational structures by teams, roles, activities, etc. REST-based.
Calendar	Used to schedule processes to being at some future point; works in tandem with the approval and work item services. REST-based.
CodeStation	Handles versioning of artifacts; agents invoke it when downloading component versions. REST-based.

Clients

Web browsers are IBM uDeploy's most common client (agents are discussed in another topic, see the section called “Agents”) but other clients can be developed to access the web services. Clients are deployed locally (on the same LAN as the IBM uDeploy server) or remotely, and communicate with the server via HTTP or HTTPS. The IBM uDeploy browser-based GUI is a Rich Internet Application (RIA) that maintains much of its functionality in the browser. Clients interact with RESTful (representational state transfer) services on the server as needed. A command line client is available that provides most of features found in the browser-based GUI. The command line client is also built on top of RESTful services.

Data Tier

Relational Database

Your relational database is a critical element for performance and disaster recovery. The provided Derby database, while sufficient for proof-of-concept work, is generally insufficient for the enterprise. Full-featured databases like Oracle, MS SQL Server, or MySQL are better options. Ideally, the database —whichever is used—should be configured for high-availability, high-performance, and be backed-up regularly.

10-20 GB of database storage should be sufficient for most environments. For Oracle, an architecture based on Oracle RAC is recommended; for Microsoft SQL Server, a clustered configuration is preferred; for MySQL, utilize MySQL Cluster.

File Storage—CodeStation

The data tier also provides log file and Codestation artifact storage. Artifacts represent deployable items such as files, images, databases, configuration materials, or anything else associated with a software project. By default, these are stored in the `var` subdirectory in the IBM uDeploy server installation directory. In an enterprise environment, the default installation might not be ideal, see the section called “Relocating Codestation” for a discussion about enterprise options.

IBM uDeploy's secure and tamper-proof artifact repository ensures that deployed components are identical to those tested in preproduction environments. Without the repository, artifacts would have to be pulled from network shares or some other system, increasing both security risks and the potential for error.

The artifact repository uses content addressable storage to maximize efficiency while minimizing disk use. The repository tracks file versions and maintains a complete history for all components. Maximizing efficiency is important, since the artifact repository stores files that are much larger than source files. Association of files with Components is built into the system. Without any configuration, each Component gets its own area of the repository for its files. There is no chance of confusion or mix-up of files to Components. And, each Component Package is mapped to a specific set of files and versions corresponding to the Component.

The artifact repository comes with a client application that provides remote access to the repository. Using the client, the user can add/modify files, create Packages, retrieve files, as well as view the history of changes. The client application provides a file transfer capability that can be used to deliver files to target servers during deployments. This built-in transfer mechanism verifies the integrity of all transferred files against their expected cryptographic signatures, thus guaranteeing that files have not been corrupted during transmission or tampered with during storage. In addition to the client application, the artifact repository exposes REST-based web services. These services are used to build integrations between build systems such as AnthillPro and IBM uDeploy. Such integrations automatically place the artifacts produced by the build process in the artifact repository, thus making the artifacts available for deployment.

Relocating Codestation

By default, the data tier's log files and Codestation artifacts are stored in the `var` subdirectory within the IBM uDeploy server directory. Ideally, this data should be stored on robust network storage that is regularly synchronized with an off-site disaster recovery facility. In addition, the IBM uDeploy server should have a fast network connection to storage (agents do not need access to the storage location). In Unix environments, you can use *symbolic links* from the `var` subdirectory to network storage. On Windows platforms there are several options for redirecting logs and artifacts, including `mklink` (supported in Windows 7 and later).

If you want to relocate Codestation, relocate both the `var` directory as well as the `\logs\store` directory. A good rule-of-thumb for determining Codestation storage requirements is: *`average artifact size * number of versions imported per day * average number of days before cleanup`*

Distributed teams should also take advantage of IBM uDeploy location-specific Codestation proxies to improve performance and lower WAN usage.

Data Center Configuration

This section provides several installation recommendations.

Cold Standby

IBM uDeploy employs the cold standby HA strategy for the application tier. When the primary system fails, the cold standby is brought online and promoted to primary server. Once online, the standby reestablishes connections with all agents, performs recovery, and proceeds with any queued processes. Because the most intense work is handed-off to agents, a high performance configuration should not have an agent installed on the same hardware as the main server.

The IBM uDeploy server aggressively utilizes threading and takes advantage of any additional CPU cores assigned to it. A small to midrange server with 2-4 multi-core CPUs is ideal, but, because it is relatively easy to move an existing IBM uDeploy server installation to a new machine, starting small and scaling as needed is a very legitimate strategy. The memory available to the application tier should also be increased from the default 256 MB to something on the order of 1 GB.

Platform Considerations

IBM uDeploy agents are platform agnostic, and can be installed on anything that provides a Java 1.5 JDK. The server process is also platform agnostic. Our customer base includes large IBM uDeploy installations on Windows, Solaris, AIX, HP-UX, other Unix flavors and various Linux platforms, all running successfully.

We have seen somewhat better performance from Unix and Linux operating systems, but recommend installing on the platform with which you are most familiar and comfortable.

Recommended Server Installation

- **Two server-class machines**

  UrbanCode, an IBM Company recommends two machines for the server: a primary machine and a standby for fail-over. In addition, the database should be hosted on a separate machine.

- **Separate machine for the database**

- **Processor** 2-4 CPUs, 2+ cores for each.

- **RAM** 8 GB

- **Storage** Individual requirements depend on usage, retention policies, and application types. In general, the larger number of artifacts kept in IBM uDeploy's artifact repository (CodeStation), the more storage needed.

- **Network** Gigabit (1000) Ethernet with low-latency to the database.

Agent Minimum Requirements

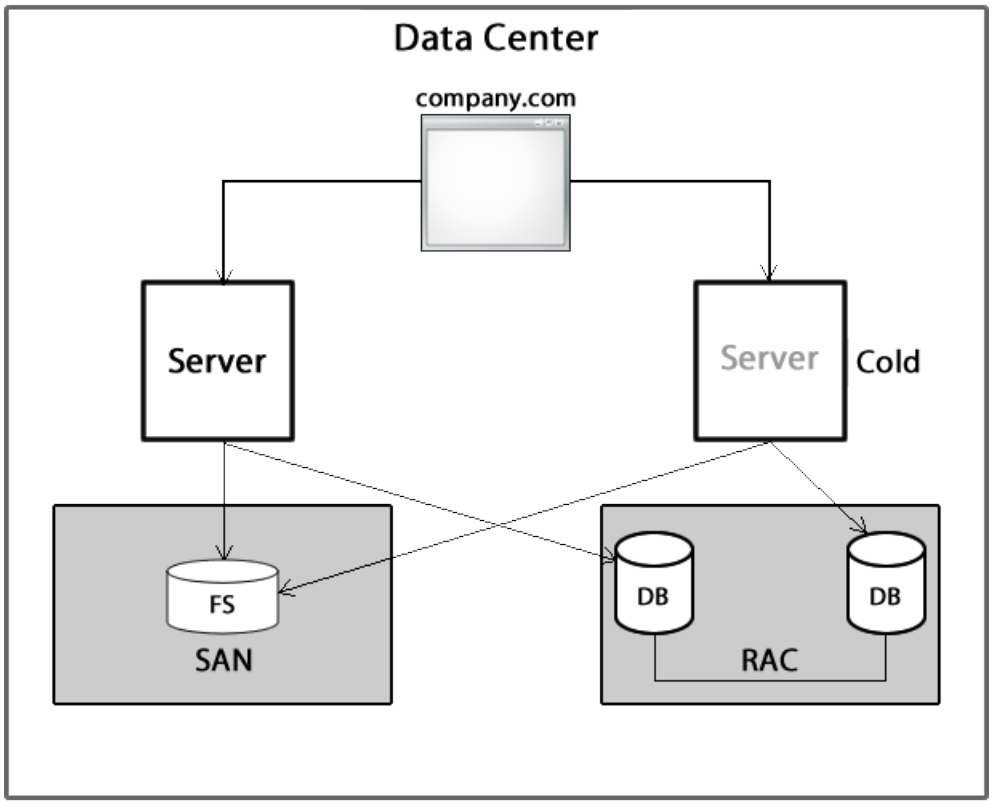
Designed to be minimally intrusive, agents require 64-256 MB of memory and 100 MB of disk space. Additional requirements are determined by the processes the agent will run. Agents should be installed on separate machines. For evaluation purposes, a good option is to install an agent on a virtual machine.

Typical Data Center Configurations

Most organizations configure the data tier with network storage and a clustered database. The service tier performs best when it's on a dedicated, stable, multi-core machine with a fast connection to the data tier. A standby machine should be maintained and kept ready in case the primary server goes down.

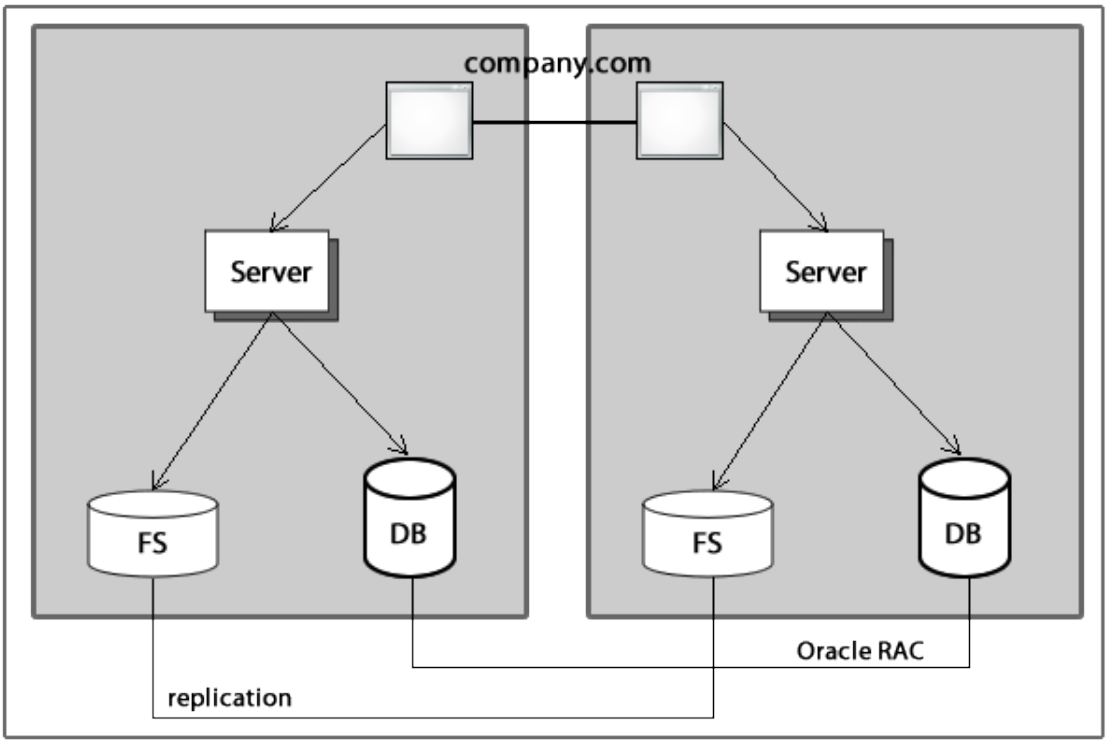
The following figures illustrate typical IBM uDeploy configurations.

Figure 5. Single Data Center Configuration



There are no remote agents or agent relays in this configuration.

Figure 6. Multiple Data Centers



Recovery Using a Database Back-up

To prepare a back-up of your IBM uDeploy installation, copy the database and server files. Back-up the server by copying the server directory along with all subdirectories. This ensures that you can revert to a server version that matches your configuration, while also preserving your artifact repository.

If you are not using the Derby database, copy the database too. If you are using Derby, it will be copied along with the server files.

Install the back-up using the original path, or some configuration files will need to be changed.

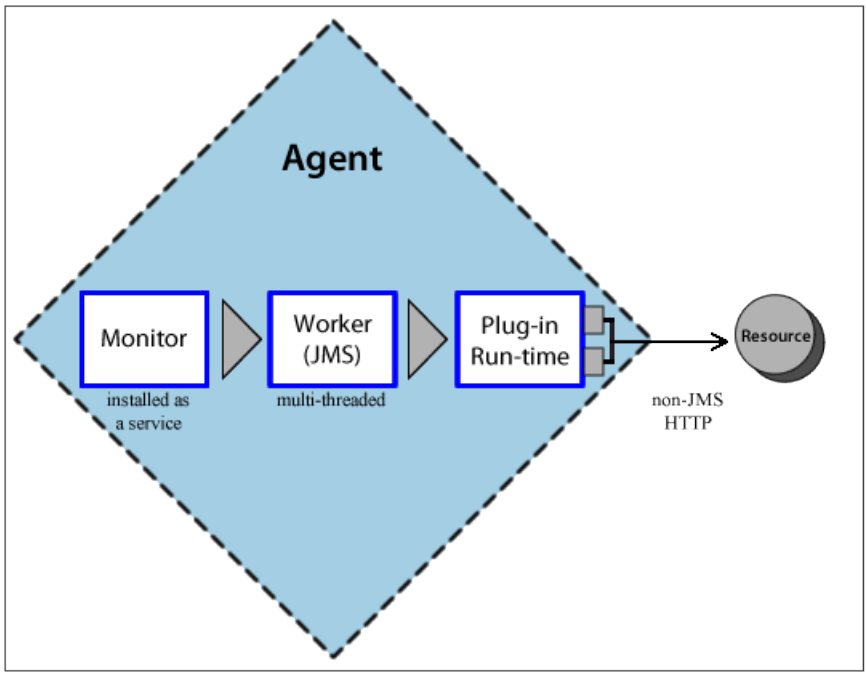
Agents

Agents play a central role in the IBM uDeploy architecture. An agent is a lightweight process that runs on a deployment-target host and communicates with the IBM uDeploy server. Agents perform the actual work of deployment which relieves the server from the task. All processes—packaging, configuration, deployments, and so on—requested by the IBM uDeploy server are executed on hardware assigned to agents. Once an installed agent has been started, the agent opens a socket connection to the IBM uDeploy server. Communication between server and agents uses a JMS-based (Java Message Service) protocol and can be secured using SSL, with optional mutual key-based authentication for each end-point. This communication protocol is stateless and resilient to network outages (the benefits of statelessness are discussed below).

While we characterize an agent as a single process, technically an agent consists of a *worker* process and a *monitor* process. The worker is a multi-threaded process that performs the actual deployment work after receiving commands from the server. Work commands come from plug-in steps which provide seamless

integration with many third-party tools. The monitor is a service that manages the worker process--starting and stopping, handling restarts, upgrades, and security, for example. Agents are rarely upgraded because their functionality is derived from plug-ins, which can be upgraded at will. Once an agent is installed, it can be managed from the IBM uDeploy web application.

Figure 7. Agent Processes



Agents are an important part of IBM uDeploy's scalability. By adding more agents, the throughput and capacity of the system increases almost exponentially and so can scale to fit even the largest enterprise.

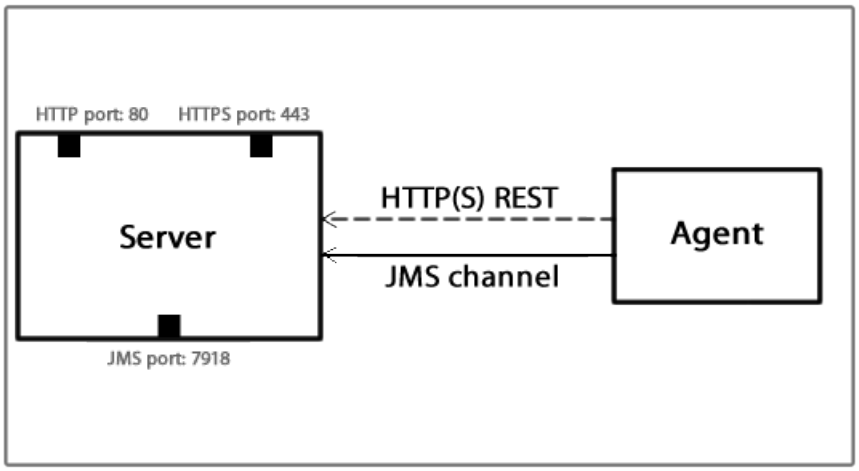
Server-Agent Communication

Most agent communication is done with JMS, but some agent activities—posting logs, transmitting test results, or posting files to CodeStation, for example—use the web tier via HTTP(s) as needed. The JMS channel is IBM uDeploy's primary control channel; it's the channel the server uses to send agent commands. By default the server listens on only three ports: port 7918 for JMS, 8080 for HTTP, 8443 for HTTPS.

The agent monitor service uses JMS for all server communications and for sending commands, such as "run step," to the worker process. The worker process uses JMS for system communications, and HTTP REST services when performing plug-in steps or retrieving information from the server.

Stateless server-agent communication provides significant benefits to performance, security, availability, and disaster recovery. Because each agent request is self-contained, a transaction consists of independent message which can be synchronized to secondary storage as it occurs. Either endpoint--server or agent—can be taken down and brought back up without repercussion (other than lost time). If communications fail mid-transaction, no messages are lost. Once reconnected, the server and agent automatically determine which messages got through and what work was successfully completed. After an outage, the system synchronizes the endpoints and recovers affected processes. The results of any work performed by an agent during the outage are communicated to the server.

Figure 8. Stateless Communication



In Figure 8, “Stateless Communication”, the arrow represent the direction in which communications was established, but the flow can be in both directions with JMS.

Remote Agents--Crossing Network Boundaries and Firewalls

IBM uDeploy supports remote agents—cross-network deployments. As long as there is at least a low bandwidth WAN connection between the server and remote agents, the IBM uDeploy server can send work to agents located in other geographic locations. To aid performance and ease maintenance, IBM uDeploy uses agent relays to communicate with remote agents. An agent relay requires that only a single machine in the remote network contact the server. Other remote agents communicate with the server by using the agent relay. All agent-server communication from the remote network goes through the relay. Agent relays can be configured as CodeStation proxies in order to optimize the transfer of large objects.

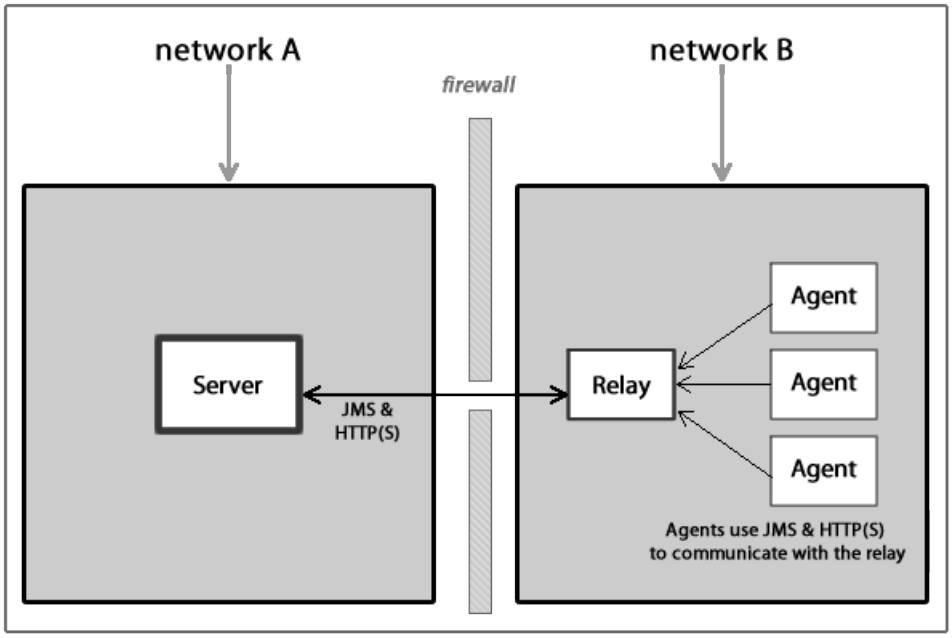
The following, a simple artifact move, illustrates the mechanics of remote communications:

1. A remote agent starts and establishes a connection to the agent relay via JMS, which, in turn, alerts the IBM uDeploy server via JMS that the remote agent is online.

2. The server sends, say, an artifact download command to the relay via JMS, and the relay delivers the message to the remote agent (also via JMS).

3. The server locates the artifacts, and then:
   - Uploads the artifacts to the relay over HTTP(s), which begins streaming them directly to the agent over the server-relay HTTP(s) connection.
   - Once the remote agent completes the work, it informs the server via JMS.

Figure 9. Crossing Network Boundaries



By default, agent relays open the connection to the IBM uDeploy server, but the direction can be reversed if your firewall requires it. Remote agents open connections to the agent relay.

In configurations with relay agents, agents local to the IBM uDeploy server continue to use direct communications.

Agent Security

IBM uDeploy agents employ user impersonation when required to perform tasks for which they would not otherwise have permission. To run a database update script, for example, an agent might need to be the "oracle" user; but to update the application, the agent might need to be the "websphere" user. By using impersonation, the same agent can run the script and update the application, which enables you to combine these steps into a single process. For information about user impersonation, see the section called “User Impersonation”

User Impersonation

IBM uDeploy can use user impersonation when an agent must execute a command for which it might not otherwise have permission, or when a specific user must be employed for a given process. On Unix/Linux systems the *su/sudo* commands are used to impersonate users; on Windows IBM uDeploy provides a utility program to handle impersonation. You implement impersonation when you configure a component's plug-in process step.

Using *su/sudo*

The `su` command (as used by IBM uDeploy) enables a user to start a shell as another user (process steps can be considered individual shells). When you configure a process step (see ???), you can tell IBM uDeploy to use impersonation for the step. By default, `su` is used but you can use `sudo` instead. To configure impersonation, you supply the user name required by the target host. When the impersonation-configured process step runs, the *su* or *sudo* command runs the step as the impersonated user. Each step that needs user impersonation must be configured independently.

Before *sudo* can be used, impersonation privileges must be defined in the */etc/sudoers* file. When you configure *sudoers*, ensure that the impersonating user does not have to supply a password. Typically, you would configure the /etc/sudoers file like this:

```
Defaults:X !requiretty

X ALL=(Y) NOPASSWD: ALL
```

where X and Y are user names. Configured this way, user X can run any command as user Y without supplying a password.

su and *sudo* maintain a record in the system logs of all of their activity. *su* can be used without configuring the *sudoers* file. For information about *su/sudo* see the Unix/Linux documentation.

Note

For Unix- or Linux-based agents the password option is ignored.

Impersonation on Windows Systems

For agents running on Windows platforms, IBM uDeploy provides a program that handles impersonation. You implement impersonation for Windows-based agents the same way you do for Unix- or Linux-based agents: when you configure a process step, you specify the *local user* credentials—user name and password—that will be used when the step is processed. For impersonation purposes, a local user is one whose user name and password are stored on the target computer and who is part of the administration group and has, at a minimum, the following privileges:

- *SE_INCREASE_QUOTA_NAME* (adjust memory quotas for a process)
- *SE_ASSIGNPRIMARYTOKEN_NAME* (replace a process-level token)
- *SE_INTERACTIVE_LOGON_NAME* (local logon)

You can also impersonate the Windows LocalSystem account. The LocalSystem account is installed on every Windows machine and is the equivalent of the root user on Unix/Linux. It is guaranteed to have the privileges listed above.

Note

For Windows-based agents the *sudo* option is ignored if selected.

SSL Mutual Key-based Authentication

SSL (Secure Socket Layer) technology enables clients and servers to communicate securely by encrypting all communications. Data are encrypted before being sent and decrypted by the recipient—communications cannot be deciphered or modified by third-parties.

SSL technology can be used in several modes. In *unauthenticated mode*, communication is encrypted/decrypted but users do not have to authenticate or verify their credentials. By default IBM uDeploy uses this mode for its JMS-based server/agent communication. By default, JMS-based communication uses port 7918.

SSL unauthenticated mode can also be used for HTTP communication. You can implement this mode for HTTP during server/agent installation, or activate it afterward. See ???.

In *mutual authentication mode*, communications are encrypted as usual, but users are also required to authenticate themselves by providing digital certificates. A digital certificate is a cryptographically signed document intended to assure others as to the identity of the certificate's owner. IBM uDeploy certificates are self-signed.

When mutual authentication mode is active, IBM uDeploy uses it for JMS-based server/agent communication. In this mode, the IBM uDeploy server provides a digital certificate to each agent, and each agent provides one to the server. This mode can be implemented during server/agent installation, or activated afterward. See ??? for information about activating this mode and exchanging certificates between the server and agents.

Unauthenticated mode for HTTP and mutual authentication mode for JMS are optional; you can implement one without implementing the other, or implement both.

Frequently Asked Questions—FAQ

Deployments

Can IBM uDeploy forecast download times for specific targets?

While IBM uDeploy does not currently forecast deployment times for specific hosts, it does calculate the average time for previous deployments. Using that information and several other factors, such as the number and size of files, and machine and network speed, estimates can closely mirror actual results. Reports are available for numerous deployment metrics, including average deployment time.

Can deployments be versioned?

Yes, all deployments are versioned. Every deployment tracks artifact (file) versions, process versions, and parameter and configuration versions. Every deployment is completely audit-able and reproducible.

What deployment mechanisms are supported?

While target machines/agents *pull* from the IBM uDeploy server, the server actually directs the work by instructing agents to pull. See the section called “Can an agent/target host check the IBM uDeploy server for outstanding work?”.

What happens when a target host is unexpectedly down during a deployment?

When the target host comes back online, an event is generated. If the deployed items do not match what should have been deployed, the machine is considered *non-compliant*. The event handler generates a new event that can be handled in a number of ways:.

- a notification can be sent to users
- a deployment can be automatically triggered to bring the host into compliance

Agents/Targets/Hosts

How are target hosts added to IBM uDeploy?

Target hosts typically have a IBM uDeploy agent installed on them. They can also be accessed through agent relays (also called proxy agents), see the section called “Remote Agents--Crossing Network Boundaries and Firewalls”. Agents automatically register themselves with IBM uDeploy when they come online.

Can an agent/target host check the IBM uDeploy server for outstanding work?

Yes. This is handled by the event system. For example, when a new agent comes online for the first time, the event engine automatically matches it to a resource group that might be associated with an application environment. The event engine can then determine if any outstanding work should be performed.

A schedule can also create events for agents.

Can changes to an individual target host be shown?

Yes. Each target can be individually viewed, along with its file versions and requests.

Can IBM uDeploy compare the image of an item to be downloaded to what is currently deployed?

Yes. IBM uDeploy knows file versions and the differences between versions, and can download only changed files if desired.

Rollbacks

How are deployment rollbacks handled?

Rollbacks are taken into consideration during the development of integrations with deployment targets, such as content systems, databases, and middleware. Deployment rollbacks typically involve making changes to non-transactional systems and are done with compensating actions—actions that undo prior actions.

For example, to rollback an incremental deployment (one in which files are added, deleted, and modified), added files need to be removed, deleted files need to be restored, and modified files changed back to their original state. IBM uDeploy provides built-in plug-in steps to perform incremental updates as well as rollbacks.

How are database rollbacks handled?

IBM uDeploy's database deployment toolkit provides forward scripts that can apply database changes, as well as corresponding rollback scripts to undo those changes. The database deployment toolkit can handle database deployments and rollbacks.

How are middleware rollbacks handled?

Most middleware rollbacks, such as WAR or EAR file deployments, are usually straightforward. In most situations, rolling back to a prior version of an EAR file is as simple as re-deploying an earlier version. Middleware configuration rollbacks might be more complex as they sometimes require service restarts.

Approvals/Notifications

What are the approval capabilities of IBM uDeploy?

IBM uDeploy's approval-type workflows provide great flexibility when designing approvals with multiple roles or paths. Approvals can be attached to applications and/or environment. Whenever a workflow executes, IBM uDeploy determines if an approval is required; if it is, notifications are sent to all users in approver roles. Only when the approval workflow completes successfully will the deployment be executed.

What are the alerting/notification capabilities of IBM uDeploy?

Notifications are triggered by events—anything that happens in IBM uDeploy can have a notification associated with it. Notification schemes enable you to configure who should be notified, as well as how and when notifications should be made. 'How' refers to the method of delivery, such as email or SMS, as well as the message template. 'When' identifies the triggering event. For example, IBM uDeploy can notify you if a process has failed and identify the effected machines.

Why are descriptions missing from notifications?

If a description is not supplied when a process is started, the triggered notification will not have a description.

Data Repository and File Storage

How are Applications Stored and Versioned?

Files associated with an application are versioned in our artifact repository, CodeStation. CodeStation ensures that files deployed to production environments are exactly the same as those deployed for testing and approved for production.

In addition to CodeStation, IBM uDeploy can integrate with external repositories (SCM, network share, Maven repositories, etc.) where versioned files can be stored. When an external repository is used, IBM uDeploy maintains meta data about the files along with references to the external repositories.

How are files transferred?

IBM uDeploy uses a highly optimized file transfer protocol that is secured using SSL encryption with optional mutual authentication.

Miscellaneous

What command line tools are available?

IBM uDeploy provides a command-line interface that has access to the IBM uDeploy server. It can be used to find or set properties, and perform numerous product functions. To install the tool, download the `udclient.zip` from the IBM uDeploy release page on Supportal (http://support.UrbanCode, an IBM Company.com).

How can IBM uDeploy scale to handle, say, 60K server instances?

The IBM uDeploy server supports active-active architecture which enables multiple server instances to share work and load balancing. Multiple active-active instances of IBM uDeploy can share workflow execution, which enables a large number of concurrent workflows to be executed.

In addition, agent relays act as agent proxies and work on behalf of the IBM uDeploy server. By providing a single avenue of communication, a relay can provide the logical configuration for a large number of targets. This reduces the overhead of agent management and artifact distribution.

What is the largest number of Unix-based data centers supported by a customer?

Our largest customer has multiple data centers with over 3,000 Unix servers in each one. The data centers are located in different geographic regions and time zones.

Can user-created scripts be incorporated in deployments?

Yes, scripts can be added to any process. IBM uDeploy supports scripting with PowerShell, groovy, Unix shell, JavaScript, windows batch files, and many others. In addition, any type of script interpreter can be

added to IBM uDeploy's shell step in workflow processes. For example, you can configure the network by running a PowerShell script on the target machine, then perform automated deployment steps, then, finally, run another script to verify that the deployment has been performed.

Glossary

A

Agent	Agents are light-weight Java processes that run on the agent machine. Agents allow the distribution of tasks for performance and multi-platform support. The agent contacts the server whenever the agent process is started. Since the agent communicates with the central server, it need not be on the same network as the central server. However, the agent must be able to open a socket connection to the server. By default, all communication between the central server and the agent is not secure. Communication may be secured using SSL. Agents are light-weight Java processes that run on the agent machine. Agents allow the distribution of tasks for performance and multi-platform support. The agent contacts the server whenever the agent process is started. Since the agent communicates with the central server, it need not be on the same network as the central server. However, the agent must be able to open a socket connection to the server. By default, all communication between the central server and the agent is not secure. Communication may be secured using SSL.
agent	An agent is a lightweight process that runs on a host and communicates with the IBM uDeploy server. Agents manage the resources that are the actual deployment targets.
Agent Filters	Agent Filters are used to select one or more agent(s) at run time and to monitor the quiet period.
agent pools	An agent pool helps you organize and manage agents installed in different environments.
Agent Relays	The new communication relay acts as a proxy for agents (which perform work on behalf of the AnthillPro server) that are located behind a firewall or in another network location. Available under separate download.
agent relays	An agent relay is used to communicate with remote agent. An agent relay requires that only a single machine in the remote network contact the server.
AHPTool	AHPTool is a command-line interface for AnthillPro that provides communication between agent-side commands, scripting and the AnthillPro central server. AHPTool is focused on setting and retrieving information from the AnthillPro server in the context of a running workflow: It can be used to look up or set properties at the system, step, request, job, Build Life and agent levels. AHPTool can upload and retrieve Test, Coverage, Analytics, or Issue data in the form of an XML document to AnthillPro -- making it an excellent integration point for writing your own plug-in or script.
Any Agent Filter	An any agent filter selects the first available agent. Also, it returns all online agents in the environment, ordered by a combination of current load and throughput metric.
applications	An application is the mechanism that initiates component deployment; they bring together components with their deployment target and orchestrates multi-componet deployments. An Application must have one component.

application process	An application process can run automatically, manually, or on a user-defined schedule. An application process can orchestrate the entire process including putting servers on-and-off line for load-balancing as required.
Application Security Report	The application security report provides information about user roles and privileges defined for IBM uDeploy-managed applications.
Artifact	For any artifact associated with this project, the files produced by a workflow published as the artifact.
artifacts	Artifacts are files, images, databases, configuration materials, or anything else associated with a software project.
Artifact Retention Policy	Manages how long you will keep the artifacts in order to save disk space.
Artifact Set	An Artifact Set is a label for a collection of build artifacts. Artifact Sets are used to define dependency relationships: Any project that another project depends on generates one or more collections of files used by the dependent project.
Authentication Realm	Authentication Realms are used to determine a user's identity within an Authorization Realm.
Authorization Realm	Authorization reals are used to associate Users with Roles and to determine user access to AnthillPro.

B

Bar Chart	Displays the data in a bar chart embedded in HTML. Data must all be numeric.
blackout	Blackouts are set per-environment, per-application. Once set, no deployments (nor Snapshots) can be scheduled to occur in that Environment. Any previously scheduled deployments to the Environment will fail if they fall within the blackout date you set.
Build-Farm	The default environment containing all agents used for pure-build processes.
Build Life	represents all the transformations a build has gone through, and the processes such as deployments and testing that the artifacts have undergone.
Build Life Links	Build Life Links are used to pass the URL of resources outside of AnthillPro to the Build Life.
Build Life Originating Workflow Request Property	When set as a workflow property, the value will get pushed to the Build Life originating workflow before the build begins.
Build Life Properties	typically used to hold variable data for builds or to create an audit trail. Once the property is set, and the build has run, the Build Life Property will be visible on the Build Life page.
Build Request Property	Build Life Tools- Build Life Tools are available through the UI to help you diagnose problems and mange Build Lives.
Build Request Property	The property will get pushed to the build request before the build begins.
Build Workflow	A build workflow defines the process for running jobs.

C

Caching Proxy	Could reduce bandwidth and improve AnthillPro’s response time. This is especially helpful in distributed development environments: the proxy can improve performance for users at off-site locations because commonly used pages are loaded from the locally stored cache.
Cleanup	Cleanup configures AnthillPro to periodically cleanup old Build Lives, build request and miscellaneous jobs generated by a project. Cleanup is on a per-project basis, so every project that uses the same Life-Cycle Model will have the same policy. To clean up the Build Life, the user can delete, inactivate, or archive the Build Life.
Cleanup Build Lives	determines when to cleanup build lives. The cleanup of build lives is base on the status the build life has achieved.
Cleanup Policy	Specifies when to delete information about old Build Lives and other task associated with the project.
Cleanup Schedule	A Cleanup Schedule kicks-off the Cleanup.
Cleanup Build Request	A Cleanup Build Request determines when build request are cleaned up.
Clone Instance	Cloning an instance of your AnthillPro server will allow you to change scripts, optimize processes, improve build performance, move the server, etc., without having to experiment on the production server.
CodeStation	CodeStation is the name of AnthillPro’s artifact repository management and access tool set. It provides support of dependencies of third-party tool kits and software libraries. CodeStation is also responsible for bringing the dependency management lookup and retrieval utilities to the individual developer.
CodeStation	CodeStation is IBM uDeploy's artifact repository. It provides secure and tamper-proof storage. It tracks artifact versions as they change and maintains an archive for each artifact.
CodeStation Artifact Time-to-Live	This feature determines how long unused artifacts remain in the cache.
components	A component represents deployable items along with user-defined processes that operate on it. Components are deployed to a resource by agents.
component inventory	A component inventory tells you what version of the component is running on a resource.
component process	A component process is a series of user-defined steps that operate on a components artifacts.
Component Security Report	A component security report provides information about user roles and privileges defined for components.
component template	Component templates enable you save and reuse component processes and properties. Components based on templates inherit the template's properties and process.

Conflict Strategy	A Conflict Strategy allows you to control how AnthillPro acts when a conflict occurs within the dependency tree.
Cron Schedule	A cron schedule may be configured to consider more complex criteria than a simple interval.

D

Defined-value property	Basic property type. Displays a secured field for the user (either the property name or default value) when running a build.
Dependencies	Specifies the specific artifacts, including version, from other projects that should be retrieved in order to support the workflow. Also specifies the artifacts from other projects and the directory in the checked-out source that those artifacts should be placed in.
deployment	A deployment is the process of moving software through various preproduction stages to final production.
Deployment Average Duration Report	A deployment average duration report provides the average deployment time for applications executed during a user-specified reporting period.
Deployment Count Report	A deployment count report provides information about the number of deployments executed during a user-specified reporting period.
Deployment Detail Report	The deployment detail report provides information about deployments executed during a user-specified reporting period.
Deployment Reports	A deployment report provides information about user roles and privileges defined with the IBM uDeploy security system.
Deployment Total Duration Report	A deployment total duration report provides total deployment for applications executed during a user-specified period.
design space	The process editor's work area, where plug-in steps are configured and process flows defined.
Dev-kit	The Dev-kit provides comprehensive documentation on AnthillPro scripting, remoting, using SOAP-based web services with AnthillPro, the AHPTool (command-line tool) and creating your own Plug-in for AnthillPro.
digital certificate	A digital certificate is a cryptographically signed document intended to assure others about the identity of the certificate's owner.
Distributed Servers	Distributed Servers is a complimentary product to AnthillPro designed to reduce the reliance on WAN connections by performing all the heavy lifting (builds, etc.) on local networks. There are three main components to Distributed Servers: Distributed Web Interface, Codestation 2.0, and Agent Relay.
Distributed Web Interface	The interface mirrors the AnthillPro server, providing the same information and functionality as the main server, for AnthillPro users.

E

Empty-value properties	enable users to set a standard for the available attributes a user can provide when executing a resource. Using an empty-value property gives users the option to

	define a required value or simply pass a blank value that AnthillPro will effectively ignore.
environment	An environment is a user-definded collection of one or more resources that host an application. At least one environment must be associated with the process before the process can be executed.
Environment	An environment is a partition grid of agents that is specific for different stages of a project life-cycle.
Environment Group	determines the set of environments that project workflows may be executed on. Each environment group must contain at least one environment.
environment inventory	An environment inventory tells you both what versions of any given component is running on a particular resource.
Environment Property	Environment Properties are used to set default values for a particular environment.
Environment Security Report	An environment security report provides information about user roles and privileges defined for environments.
Event Triggers	Event Triggers use scripts to create an Event Filter that listens to events passing through the AnthillPro Event Service. When the Event Script detects an event, it can then trigger another action by the AnthillPro server.

F

Fixed Agent Filter	Fixed Agent Filters always select a specific agent within the environment. If the requested agent is locked or can't receive more work, the request will be queued until the agent frees up.
full version	A full version contains all component artifacts.

G

Guest User Account	The Guest User Account gives anonymous access to AnthillPro, and does not require a user name or password at login.

I

IDE Plug-ins	The plug-ins enable developers to view the current activity and state of projects, start new builds, map their projects to projects in AnthillPro, and resolve the project's dependency artifacts.
incremental version	An incremental version contains only artifacts that have been modified since the previous version was created.
Independent Scheduling	Related projects do not need to know about each other for independent scheduling. A dependent project starts each build by gathering the most recent artifacts from its dependencies.
Internal Projects	Allows for dependency relationships between concurrently developed AnthillPro projects.

Interval Schedule	Regularly fires after a fixed interval of time.

J

Job	A job is a series of steps detailing how to get something done.
Job-execution Property	Use the Job-execution property type if the value, default value or options are to be generated by a job that executes on an agent.
Job Iteration Property	Job iteration properties are configured after iterating a job. They are typically used to set parameters for a single job that is run many times, with a slightly different parameter each time.
Job Iteration	Job Iterations are used to run the same job multiple times.
Job Wizard	A Job Wizard assists in the creation of build jobs. The Wizard takes you through the steps to configure the builder and the publisher required by the job, and is the best way to ensure your build job will be configured successfully. The Build Life is a map of (1) what occurred during the build; (2) the processes that were performed on the generated artifacts; (3) where the build artifacts ended up.

L

Library Jobs	A Library Job is a way for you to create template jobs. Once a Library Job has been configured, it can be used by multiple projects or can be turned into a project-specific job.
Library Workflow	The Library Workflow can only use library jobs as part of its definition.
Life-Cycle Build	The Life-Cycle Build provides a wealth of information and visibility into the build and release cycle. It is defined by its Life-Cycle Model, Environments, and Build workflow.
Life-Cycle Model	The Life-Cycle Model provides a template for managing the dependencies, artifacts, deployments, etc, associated with every build of the project. They are reusable, allowing you to apply the same standards to any similar project.
LDAP	A lightweight directory access protocol (LDAP) is a widely used protocol used to access distributed directory information over the internet protocol (I.P) networks.
lock	A lock is routinely used to ensure that processes do not interfere with one another.
Lockable Resource	A Lockable Resource specifies a resource that might be used by multiple workflows, gives it a name and forces arbitrary workflows to run one at a time or in small groups.

N

Network Settings	Network Settings are used to configure communication between the AnthillPro server and agents.
Notification	Notifications inform the recipient of the status of a CI build.
notifications	Notifications play a role in approving deployments: IBM uDeploy can be configured to send out an e-mail to either a single individual or to a group or people

	(based on their security role) notifying them that they need to approve a requested deployment.
Notification Scheme	Notification Schemes determine who to notify of build status, what conditions to notify them on, and what mechanism to notify them with. sets rules determining what groups of users are sent which kind of notification about specific events.
notification scheme	A notification scheme enables IBM uDeploy to send out notifications based on events. For example, the Default Notification scheme will send out an e-mail when an Application Deployment fails or succeeds.
Notification Template	Notification Templates are Velocity templates that take information about the build and produce a document.

O

Operational Project	Operational Projects provide a simplified interface that allows AnthillPro users to execute ancillary tasks not easily run during a build, deployment, etc., workflow.

P

Permission	Permissions associate the role, resource, and an action that may be performed on the resource. Typical actions include the ability to read or view the resource, the ability to write to or modify the resource, the ability to modify the security settings for a resource, or the ability to execute the resource.
plug-ins	A plug-in is the integration with third-party tools.
Preflight Build	Preflight Builds allow developers to run a test build in the authoritative build environment before committing their changes to source control.
process	Processes play a coordination role. They are authored using a visual drag-n-drop editor.
process editor	A process editor is a visual drag-and-drop editor that enables you to drag process steps onto the design space and configure them as you go.
processing property	A processing property is a way to add user-supplied information to a process.
Project Environment Property	Project Environment Properties are automatically placed as environment variables for all commands run in the target environment.
Project Property	Project Properties are used for all workflows regardless of the target environment.
Property	Properties identify various properties that must be specified when the workflow is executed. Properties are used to manage variables passed into commands, agent filters, and custom stamping algorithm templates.
proxy resource	A proxy resource is a resource effected by an agent on a host other than the one where the resource is located.
Pull Build	A Pull Build will build every dependency that is out of date prior to building the top level of a project.
Pure Build	Pure builds take the source as input and transforms it into artifacts. In AnthillPro, it generates the Build Life.

Push Build	Push Builds perform the minimum number of builds, in the correct order, to ensure that a change to a component does not break anything that depends on it directly or transitively. A push build can become resource intensive.
Push Scheduling	Push Scheduling are used when an originating workflow builds, the workflows that depend on it automatically build.

Q

Quiet Periods	Quiet Periods are configured on the project, and play an integral part in ensuring that the source code AnthillPro obtains from the SCM contains a consistent set of changes. The quiet period is a measure of time that must pass without anyone committing changes to the source.

R

Read Permission	Read permissions allow a user to resolve/download the artifacts associated with a Build Life.
relay servers	A relay server enables network-to-network communication.
remote agents	A remote agent is an agent that will communicate with the server via an agent relay.
Report	Reports provide information about system activity.
Repository Trigger	Repository Triggers are used for Continuous Integration builds. Once the trigger is active for a workflow, every commit of source changes will initiate a build.
resource	A resource is a user-defined construct based on IBM uDeploy's architectural model. A resource represents a deployment target.
resource group	A resource group is a group of resources used to help organize and manage the agent installed in a different environment.
Resource Security Report	A resource security report provides information about user roles and privileges defined for resources.
role	A role enables you to further refine how a resource is utilized, and is similar to subresources.

S

Schedule	A Schedules determines when events such as builds, cleanups, backups, etc., are automatically run by the system. Once a schedule has been created, it may be used by many different AnthillPro resources.
Schedule Trigger	A Scheduled Trigger runs on its own timer that pools the SCM for changes. If changes are detected, the build is registered with the schedule and kicked off when the schedule fires. Scheduled Triggers give the option to force a build regardless of whether source changes have occurred.
schema	A schema is a visual representation of the different parts of IBM uDeploy that may be secured. Each Schema interacts with users indirectly, through the role.

Script Library	the script library is used to create, organize, and provide security around often-used AnthillPro scripts. The Script Library is most helpful for large organizations, allowing them to ensure that only the appropriate team members can modify a script.
Scripted Property	Scripted value properties are properties where the value, default value or options are generated by a script that is executed before workflow execution.
Scripted Agent Filters	A scripted agent filter selects agent based on an agent filter script.
SSL	A secure socket layer (SSL) enables clients and servers to communicate securely by encrypting all communications.
Security Permission	Security Permissions allows users to determine who can set security for the artifact set.
Security Reports	A security report provides information about user roles and privileges.
Security Setting	A Security Setting is used to configure access to the Anthill Server setting for configuration and artifact management.
Server Proxy	Server proxies enable you to access servers/repositories that are on external networks.
snapshot	A snapshot is a collection of specific component versions, usually versions that are known to work together.
Source Configuration	Source Configuration identifies exactly which source code should be retrieved from a repository.
Stamp	A Stamp is a configurable name for the build number, build identifier, or version number used to identify a build. This allows you to mimic your strategies.
Stamp Context Script	Stamp Context Scripts generate values for variables in the stamp context when creating a stamp (for builds, etc.).
Stamp Mapping	For every stamp style in the project's stamp style group, one must specify what stamping strategy will be used. A typical stamp style group might specify for development builds to a version number and strategy for incrementing that number.
Stamp Style	Stamp Styles are used to apply different stamps to a single build, and allow you to help track a build throughout its life-cycle.
Stamp Style Group	A Stamp Style Group creates common names for types of stamps.
Status Group	A Status Group is a set of common names for statuses.
stateless	Stateless means the server retains little session information between requests, and each request contains all information needed to handle it.
subresource	A subresource enables you to apply logical identifiers or categories within any given group.
switch step	A switch step enables you to create conditional processes.
System Tray Monitor	The System Tray Monitor provides feedback directly to the desktop, without having to open or refresh a browser.

System Property	The System property is used to set default values for a particular property for all workflows and project system wide.

T

Template Reports	Template Reports are good for generating tabular data in one or more types of output.
Trigger	A Trigger is an automated mechanism for kicking off a workflow.

U

uncontrolled environment	A uncontrolled environment is an environment that does not contain approvals approval gates.
user impersonation	IBM uDeploy can use user impersonation when an agent must execute a command for which it might not otherwise have permission.

V

Velocity Report	Velocity Reports are good for customizing the AnthillPro UI.
version	A version is set each time a component changes. There are two types of versions a full version and an incremental version.

W

Workflow Definition	A Workflow Definition defines which job should be run, species the order of jobs, as well as the elements to be run in parallel.
Workflow Priorities	Workflow Priorities allow you to determine the order in which workflows run. Use workflow priorities to determine which workflow will run first.
Workflow Property	Workflow properties are used to send a build to a particular platform when writing native code for multiple platforms.
Workflow Request Context	A workflow request context is a collection of requests for workflows that are processed together.
Workflow Task	A Workflow Task allow you to set up manual gates, etc., that must be performed.
Workflow Tool	There are two major categories of work flow tools: priorities and requests.
Workflow Request	The Workflow Request is the first action taken by the server when executing an originating or secondary workflow.

Index