

Deployit Cookbook - Using the Plan Analyzer

Version 3.9

Table of Contents

| | |
|------------------------------------|---|
| Table of Contents | 2 |
| Cookbook - Using the Plan Analyzer | 3 |
| Setup | 3 |
| Analyzing the Plan | 3 |
| Updating the plan | 5 |
| Using orchestrators | 5 |
| Starting the Deployment | 7 |
| Conclusion | 7 |

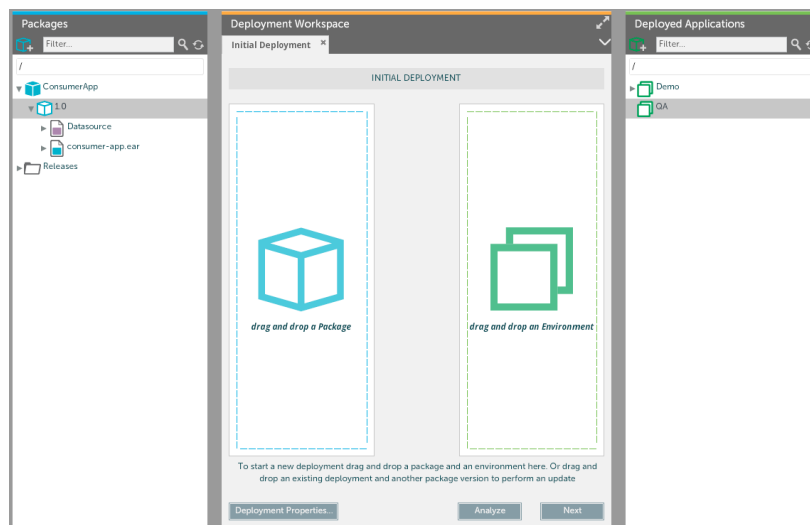
Cookbook - Using the Plan Analyzer

Ever wondered how Deployit comes up with a deployment plan? To give you more insight in Deployit's planning logic, Deployit 3.9 features a Plan Analyzer that allows you to inspect a deployment plan while it's being built up.

Setup

For this cookbook, we will use a simple Deployment Package containing an EAR file and a DataSource and deploy it to JBoss Application Server v5.

Here's the setup in Deployit:



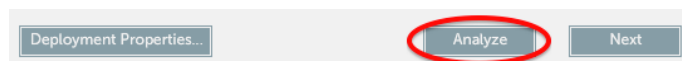
Our application is called **ConsumerApp/1.0** and we will deploy it to the **QA** environment. This is a simple environment with one JBoss server running on a unix host.

We start the deployment by dragging the **1.0** package from the Packages window to the Deployment Workspace, followed by the **QA** environment.

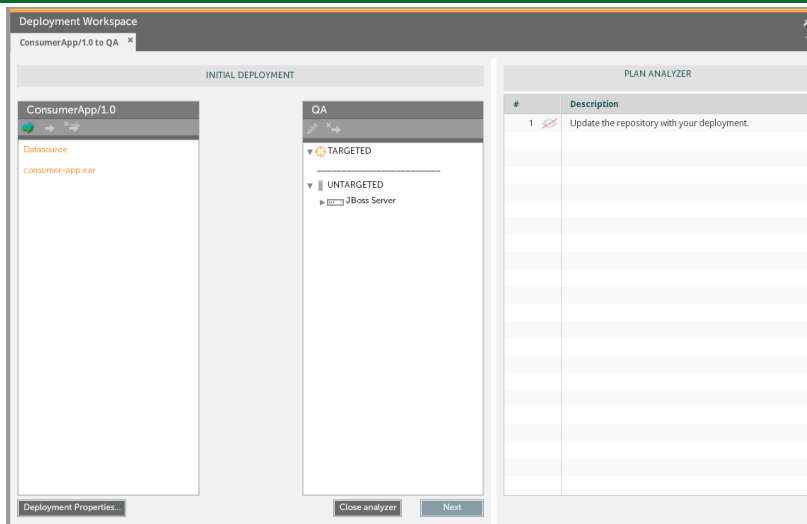
So far, so good, this is what we always do in Deployit.

Analyzing the Plan

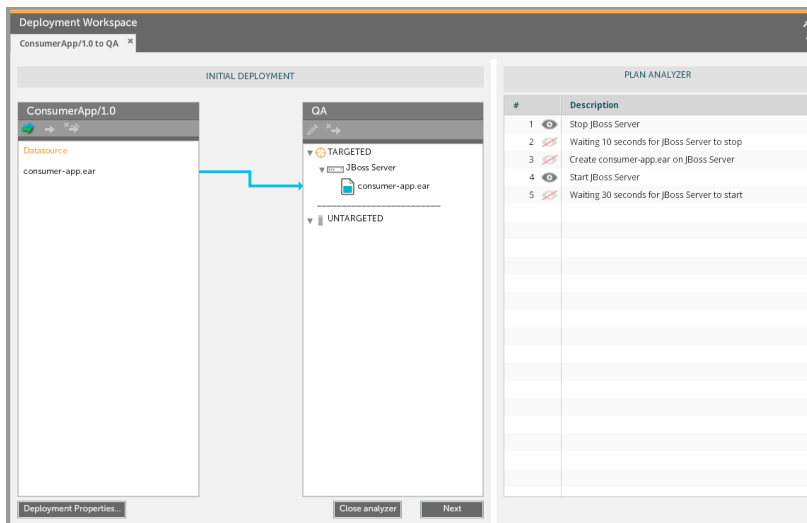
To inspect the deployment plan while it is being created, we click the Analyze button on the bottom of the screen.



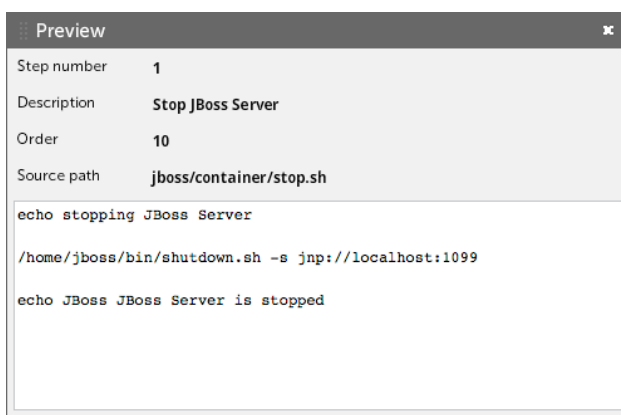
We now see our deployment specification on the left-hand side and the generated plan on the right. Currently it contains only one step, called "Update the repository with your Deployment". This is the default step that Deployit creates when there is an empty plan in order to do its housekeeping.



We now drag the **consumer-app.ear** deployable onto **JBoss Server** to see how the plan changes.



With the addition of a single deployed, five steps are added to the plan. Some steps have an preview icon in front of them. These steps have scripts attached to them. By double-clicking on the step, we can see the contents of the generated script. With this information, you will know in advance exactly what actions Deployit will execute on the remote server.



Apart from the contents of the script that will be executed, the Preview popup offers the following valuable information:

- Step number - The position of the step in the deployment plan
- Description - the name of the step
- Order - The order associated with the step. The order determines the sequence of steps in the plan, with lower order numbers coming before higher ones. The selected Orchestrator can influence this - see the section on the [Planning Stage](#) of the Deployit documentation.
- Source path - The location of the script template relative to Deployit's classpath. For example, relative to `SERVER_HOME/ext` or packaged in the relevant plugin.

Note that the step order for steps that don't have previews can also be found by hovering over the Step text and waiting for the tooltip.

| | | |
|---|--|--|
| 1 | | Stop JBoss Server |
| 2 | | Waiting 10 seconds for JBoss Server to stop |
| 3 | | Create consumer-app |
| 4 | | Start JBoss Server |
| 5 | | Waiting 30 seconds for JBoss Server to start |

Updating the plan

Now let's see what happens when we add more deployments to the mix. Let's add the **Datasource**. This time, we click on it and then click on the Map Single Deployable button - the single green arrow - above it. This will automatically map the datasource to the JBoss server in our environment, the only applicable target here. The plan is updated accordingly.

| | | |
|---|--|--|
| 1 | | Stop JBoss Server |
| 2 | | Waiting 10 seconds for JBoss Server to stop |
| 3 | | Create Datasource on JBoss Server |
| 4 | | Create consumer-app.ear on JBoss Server |
| 5 | | Start JBoss Server |
| 6 | | Waiting 30 seconds for JBoss Server to start |

The Create Datasource also has a Step Preview. Clicking on it will not reveal a shell script, but the datasource definition that will be added to the JBoss server.

Preview

Step number

3

Description

Create Datasource on JBoss Server 1

Order

50

Source path

jboss/datasource/template.xml.ftl

```
<?xml version="1.0" encoding="UTF-8"?>
<datasources>
  <local-tx-datasource>
    <check-valid-connection-sql>select from dual</check-valid-connection-sql>
    <connection-url>jdbc:oracle:thin:@//myhost:1521/orcl</connection-url>
    <driver-class>oracle.jdbc.driver.OracleDriver</driver-class>
    <jndi-name>/ds/consumer</jndi-name>
    <max-pool-size>10</max-pool-size>
    <min-pool-size>3</min-pool-size>
    <password>*****</password>
    <use-java-context>true</use-java-context>
    <user-name>jboss</user-name>
  </local-tx-datasource>
</datasources>
```

The values for the properties like 'check-valid-connection', 'jndi-name', etc are taken from the Datasource. Double click on the Datasource to edit the values of these properties.

Deployment Workspace

ConsumerApp/1.0 to QA

INITIAL DEPLOYMENT

ConsumerApp/1.0

Datasource

consumer-app.ear

PLAN ANALYZER

Datasource : JBoss Server

Name

Datasource

Type

JBossTransactionDataSource

Jndi Name

/ds/consumer

Use Java Context

☒

User Name

jboss

Password

Min Pool Size

3

Max Pool Size

10

Check Valid Connection Sql

select from dual

Apply

Cancel

Any changes made here are of course immediately updated in the Step Preview.

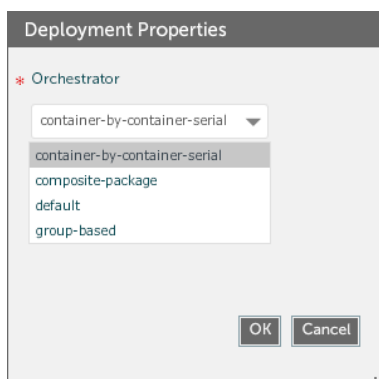
Using orchestrators

Another great way to use the Plan Analyzer is when using [Orchestrators](#). Orchestrators are used to control the sequence of the generated plan. They are used mainly when dealing with more than one server.

When we deploy to an environment with two JBoss servers, the plan will look like this:

| | |
|----|--|
| 1 | Stop JBoss Server 1 |
| 2 | Stop JBoss Server 2 |
| 3 | Waiting 10 seconds for JBoss Server 1 to stop |
| 4 | Waiting 10 seconds for JBoss Server 2 to stop |
| 5 | Create Datasource on JBoss Server 2 |
| 6 | Create Datasource on JBoss Server 1 |
| 7 | Create consumer-app.ear on JBoss Server 2 |
| 8 | Create consumer-app.ear on JBoss Server 1 |
| 9 | Start JBoss Server 1 |
| 10 | Start JBoss Server 2 |
| 11 | Waiting 30 seconds for JBoss Server 1 to start |
| 12 | Waiting 30 seconds for JBoss Server 2 to start |

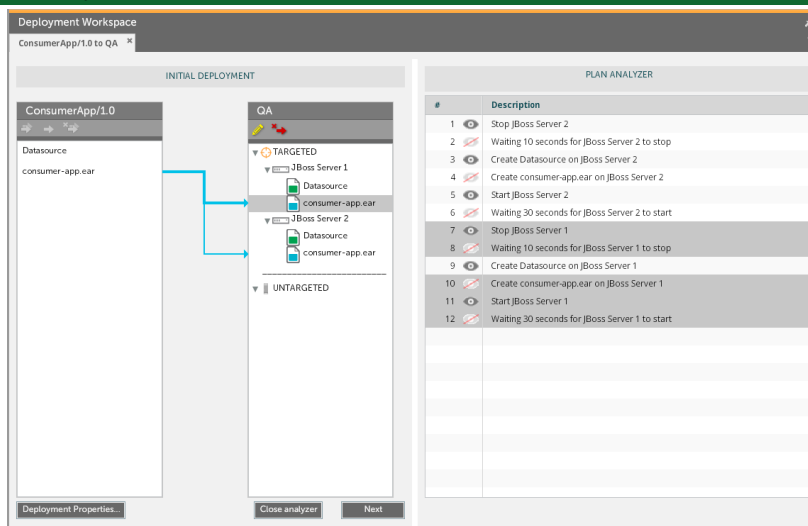
In this plan, both servers go down simultaneously. This happens because the default orchestrator treats all target middleware as one big "pool", so everything will be stopped, started, updated etc. together. We can change this by picking another orchestrator. We do this by clicking on the Deployment Properties button and selecting the container-by-container-serial orchestrator. This orchestrator treats each target server as it's own "deployment group", handling all changes to a server in one block and then moving on to the next server.



The plan is immediately updated, and now the servers are stopped one by one.

| | |
|----|--|
| 1 | Stop JBoss Server 2 |
| 2 | Waiting 10 seconds for JBoss Server 2 to stop |
| 3 | Create Datasource on JBoss Server 2 |
| 4 | Create consumer-app.ear on JBoss Server 2 |
| 5 | Start JBoss Server 2 |
| 6 | Waiting 30 seconds for JBoss Server 2 to start |
| 7 | Stop JBoss Server 1 |
| 8 | Waiting 10 seconds for JBoss Server 1 to stop |
| 9 | Create Datasource on JBoss Server 1 |
| 10 | Create consumer-app.ear on JBoss Server 1 |
| 11 | Start JBoss Server 1 |
| 12 | Waiting 30 seconds for JBoss Server 1 to start |

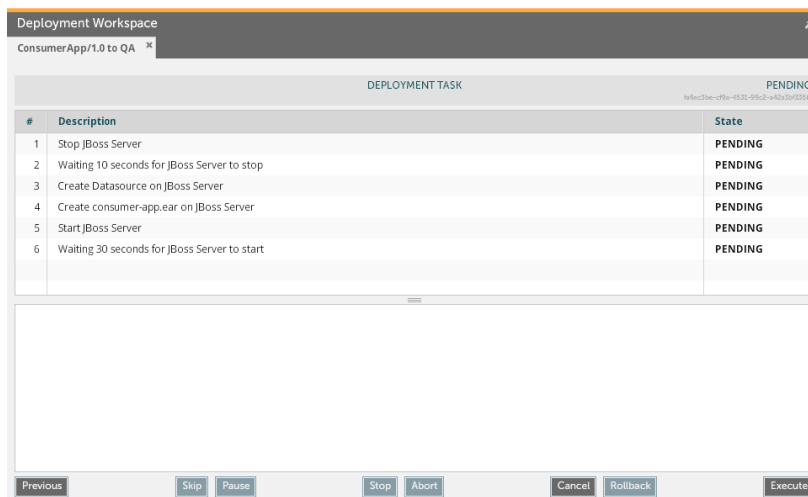
It is also useful to see which steps are related to which item in your deployment package. The Plan Analyzer also helps in this case: simply click on the deployed. For example when clicking on consumer-app.ear under JBoss Server 1, the associated steps are highlighted.



Vice-versa, when clicking on a step, the relevant deployed is highlighted.

Starting the Deployment

When you are satisfied with the generated plan, simply click Next to start the deployment.



This is the definite plan that will be executed to perform the deployment. Note that not until here, you are able to rearrange the steps of the plan by way of drag-and-drop. This is not possible in the Plan Analyzer, because the definite sequence of steps is not known yet at that stage.

Conclusion

The Plan Analyzer gives you insight in how a deployment plan is built up in Deployit, and shows a detailed view of generated scripts. It's a great tool to help you understand Deployit when deploying applications or developing your custom plugin.