https://lh5.googleusercontent.com/DRN4QVuC02ifom_NAMHupC-Z8jry1N9Bn1aTiJJIt9ovHe4Qfi3oNCtpnmX_hZEYhgdIWzxu2sniwwJtUoCWSoRWDSRAdMjO-Bc0bU1LIRW_AVCdJp0FKAFU44-pZFv2jn63r1pf8zVRjyPb7A

SFDC Automatic Deployment

Technical Guide

Version 1.0.0 - July 2016

# Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
| 27/07/2016 | 1.0.0 |  | Diogo Roxo |
|  |  |  |  |

Content

Document History 2

Content 3

1. OVERVIEW 4

2. Requirements 5

1.1. Installation 5

1.1.1. Programs and System Variables 5

1.1.2. Salesforce Auto Deploy Tool 5

1.2. Configuration 6

1.2.1. Build.properties 6

1.2.1.1. Remote Repository Configuration (Bitbucket) 6

1.2.1.2. Local Repository Configuration 7

1.2.1.3. Salesforce Org configuration 7

1.2.1.4. Input configuration 7

1.2.2. Build.xml 8

3. Usage 9

3.1. Branch Commit Log 9

3.2. Pull Request 10

3.3. Deploy Result Processing 10

3.3.1. Successful Deploy 11

3.3.2. Failed Deploy 11

4. Implementation Details 13

4.1. Configuration Files 13

4.2. Bash Scripts 13

4.2.1. runDeployFromPullRequest.sh 13

4.2.2. runDeployFromGitLog.sh 13

4.2.3. rerunLastFailedDeploy.sh 13

4.3. Python Scripts 13

5. Future Developments 14

# 1. OVERVIEW

This document describes a tool to perform an automatic deployment into a Salesforce Environment.

This tool allows to perform the deployment from two different origins:

1. From a Branch Commit Log in a remote repository (Bitbucket).
2. From a Pull Request in a remote repository (Bitbucket).

In both methods the deployment only considers the modified files for the Git diff being considered.

The examples below are presented using a Developer Sandbox.

# 2. Requirements

In order to use this tool you need to have a local GIT repository connected to BitBucket. In the examples below it was used a GIT repository and the Source Tree GUI.

## 1.1. Installation

All The components that need to be installed are present in this section.

### 1.1.1. Programs and System Variables

Programs that need to be installed and system variables to be created:

1. Install Python 3.5 (<https://www.python.org/downloads/>)

**System Variables**

**Variable Name**: PYTHONPATH

**Variable Value**:

C:/Users/Diogo/AppData/Local/Programs/Python/Python35-32;C:/Users/Diogo/AppData/Local/Programs/Python/Python35-32/DLLs;C:/Users/Diogo/AppData/Local/Programs/Python/Python35-32/Lib

1. Install Apache Ant (<http://ant.apache.org/bindownload.cgi>)

**System Variables**

**Variable Name**: ANT\_HOME

**Variable Value**: C:/Program Files/Ant/apache-ant-1.9.7

1. Install Salesforce Ant Migration Tool

(<https://developer.salesforce.com/page/Force.com_Migration_Tool>)

**System Variables**

**Variable Name**: SALESFORCE\_ANT\_HOME

**Variable Value**: C:/Program Files/Ant/salesforce\_ant\_37.0

**NOTE:** Alternatively to the download you can use the versions available in this document folder.

After this you need to add the Python and Ant installations to the System Variable PATH:

Variable Name: Path

Variable Value:

…;%ANT\_HOME%/bin; C:/Users/Diogo/AppData/Local/Programs/Python/Python35-32

## 1.1.2. Salesforce Auto Deploy Tool

Just extract this tool to somewhere in your PC. In my case I put it under:

C:/Users/Diogo/Documents/Software Projects

## 1.2. Configuration

All the User configurations are defined in the following files:

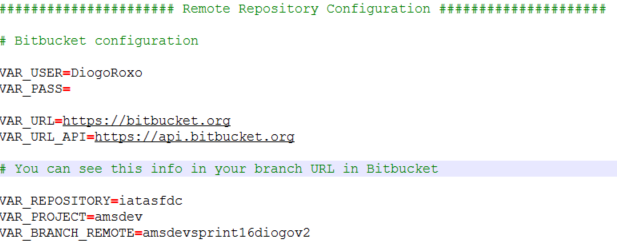
* Config/Build.properties
* Config/ Build.xml

### 1.2.1. Build.properties

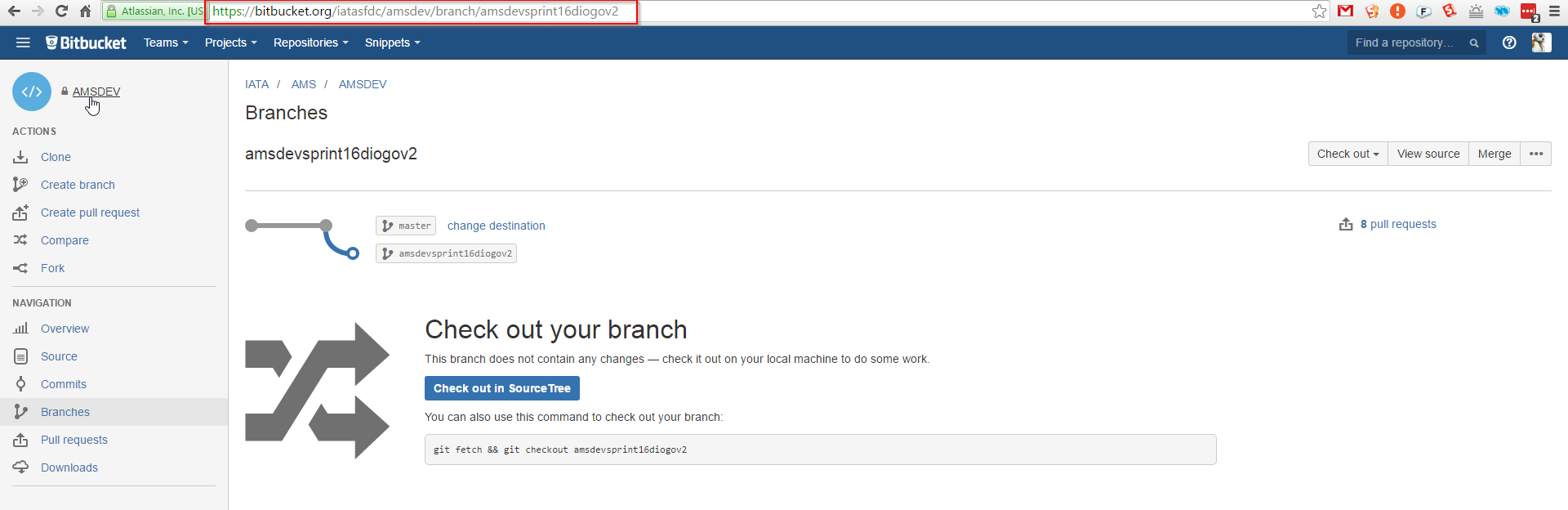
In this file you need to fill the info described below.

#### 1.2.1.1. Remote Repository Configuration (Bitbucket)

The Bitbucket configuration for your repository.

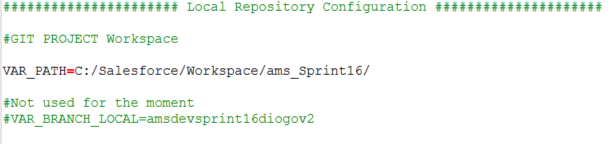


You can get this info in URL for your branch:



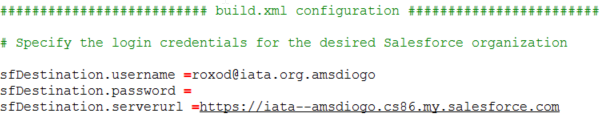
#### 1.2.1.2. Local Repository Configuration

The configuration for your local GIT repository.



#### 1.2.1.3. Salesforce Org configuration

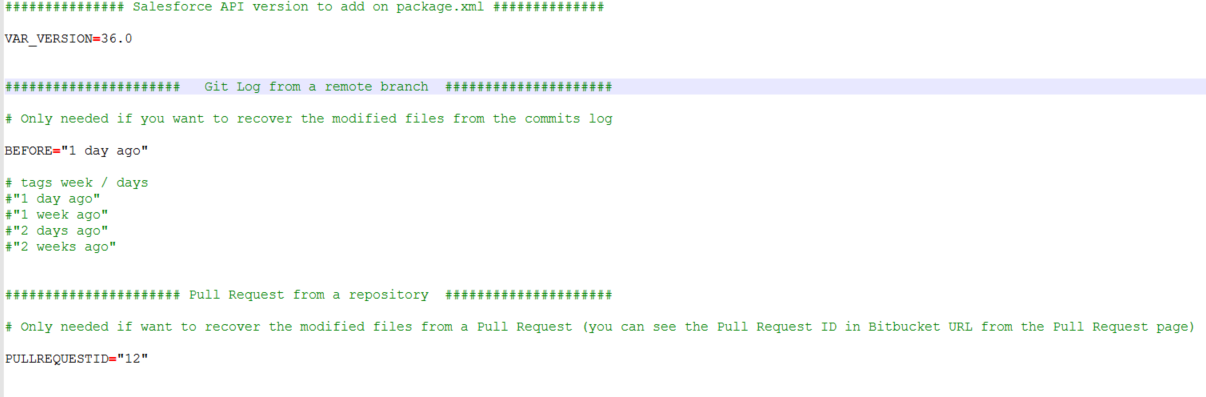
Salesforce login configuration (I used my sandbox as example).



#### 1.2.1.4. Input configuration

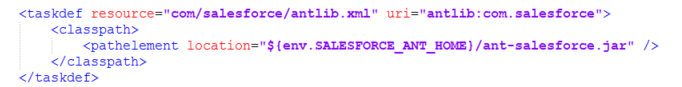
The Input configuration for the Deploy process:

1. **VAR\_VERSION** 🡪 The Salesforce API version that will be passed to the Package.xml
2. **BEFORE** 🡪 In case you want to use a **remote branch commit log** you need to specify this field as shown below.
3. **PULLREQUESTID** 🡪 In case you want to use a **remote branch Pull Request**.



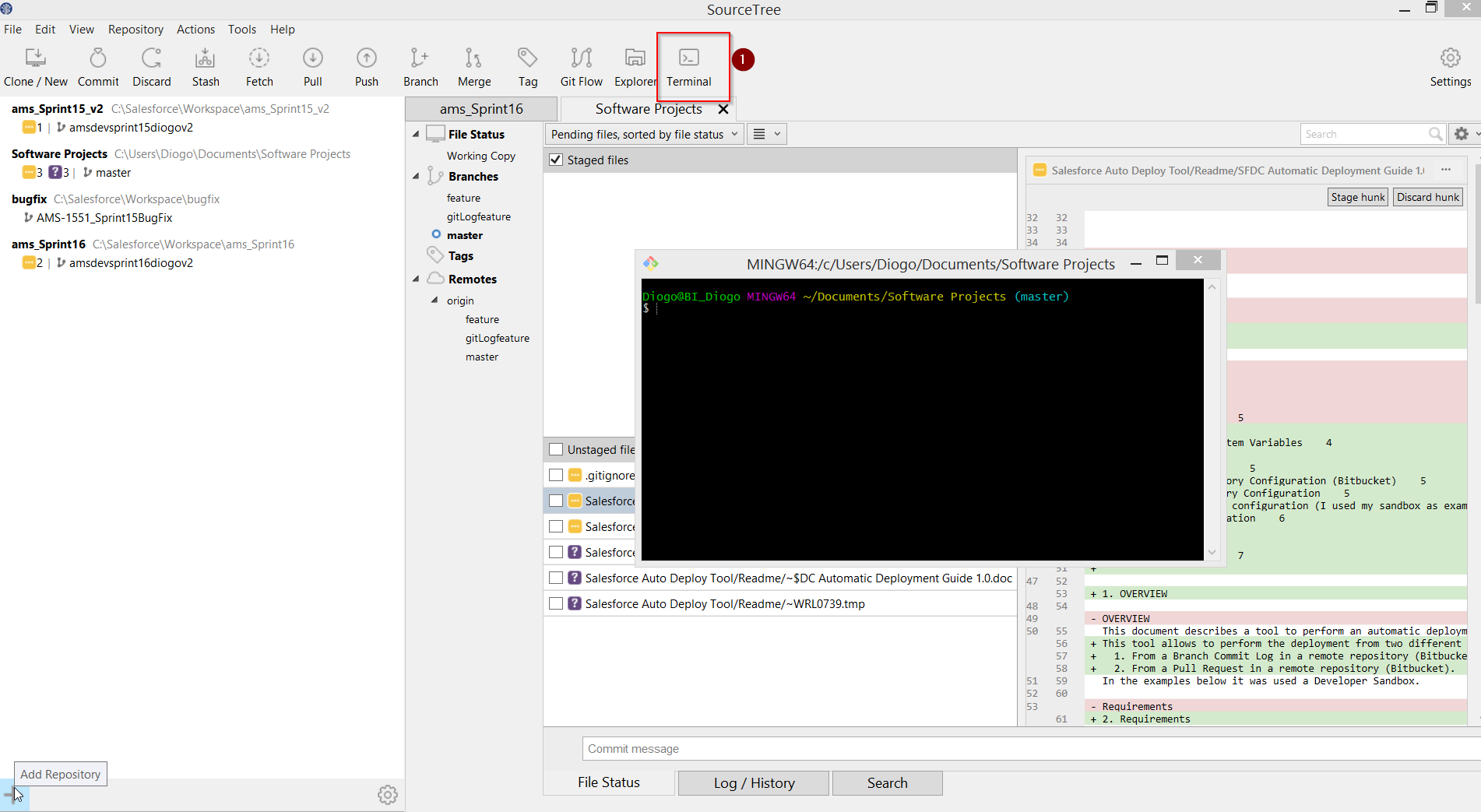
### 1.2.2. Build.xml

This file contains the Ant configuration regarding the Sandbox deployment. Just be sure you have correctly defined the **SALESFORCE\_ANT\_HOME** environment variable.



# 3. Usage

First you need to access to a command line terminal. The easiest way is to open Source Tree Terminal:



As it was explained above, the deployment can be performed from 2 different origins:

* **Branch Commit Log** in a remote repository (Bitbucket).
* **Pull Request** in a remote repository (Bitbucket).

First you have to navigate to the “Salesforce Auto Deploy Tool” folder. Just type in the terminal the command to the “Salesforce Auto Deploy Tool” folder:

**cd "C:/Users/Diogo/Documents/Software Projects/Salesforce Auto Deploy Tool"**

Then you follow the steps in **3.1** or **3.2** depending in what the deployment will be based.

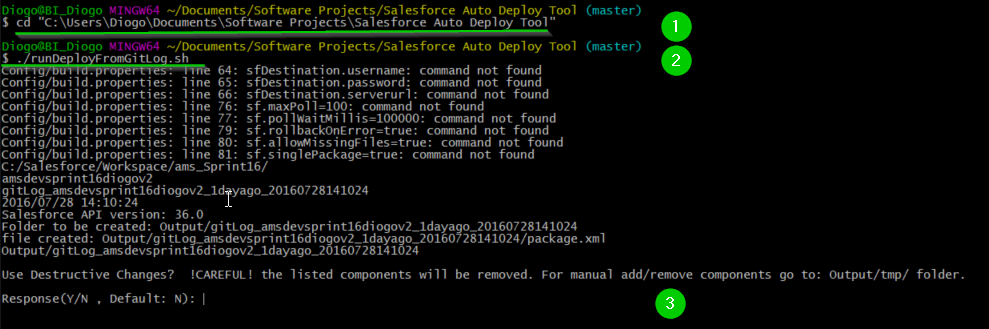
## 3.1. Branch Commit Log

After specifying the **BEFORE** variable in the input configuration you just have to run the bash command:

* **./runDeployFromGitLog.sh**

After this you will be asked for an Input:

* + **Use Destructive Changes (Y/N)**



The destructiveChanges.xml is always generated, however it will only be used if it’s explicitly specified (Y).

After this the deployment will run automatically.

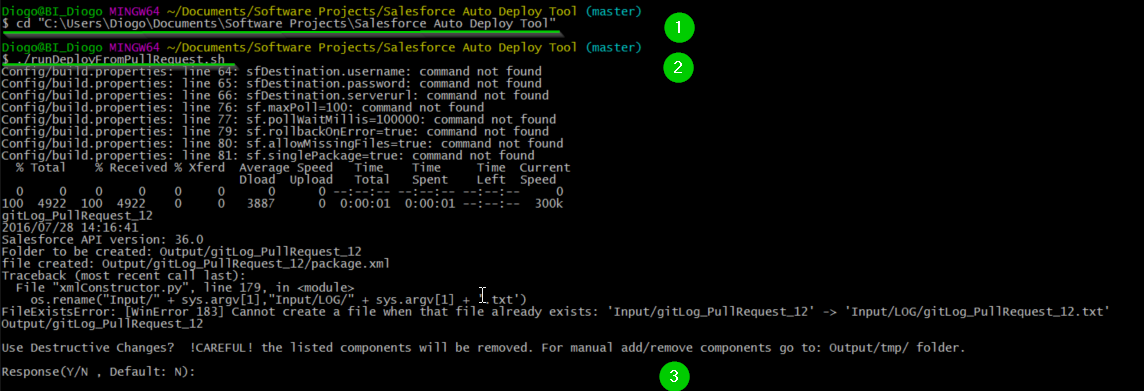
## 3.2. Pull Request

After specifying the **PULLREQUESTID** variable in the input configuration you just have to run the bash command:

* **./runDeployFromPullRequest.sh**

After this you will be asked for an Input:

* **Use Destructive Changes (Y/N)**

****

The destructiveChanges.xml is always generated, however it will only be used if it’s explicitly specified (Y).

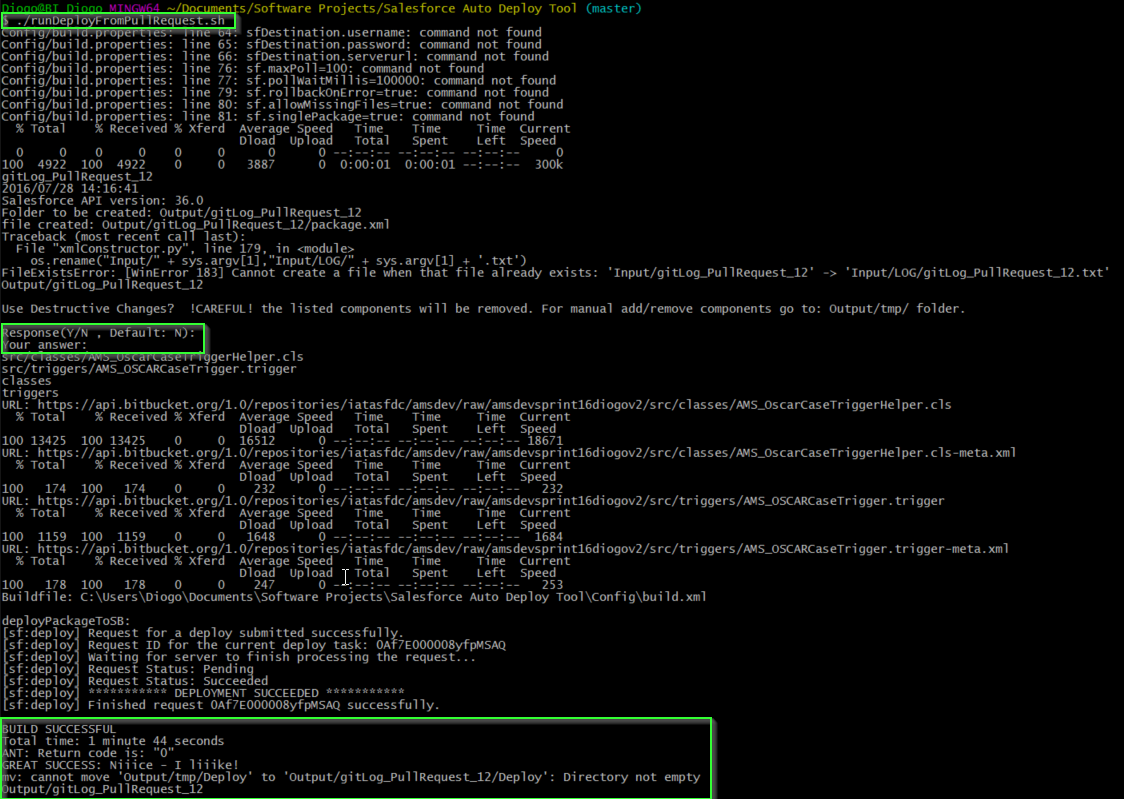
After this the deployment will run automatically.

## 3.3. Deploy Result Processing

There are two possible outcomes for the scripts in **3.1** and **3.2**: **Failed Deploy** and **Successful Deploy**.

### 3.3.1. Successful Deploy

When the deployment is successful it will be printed on the terminal:



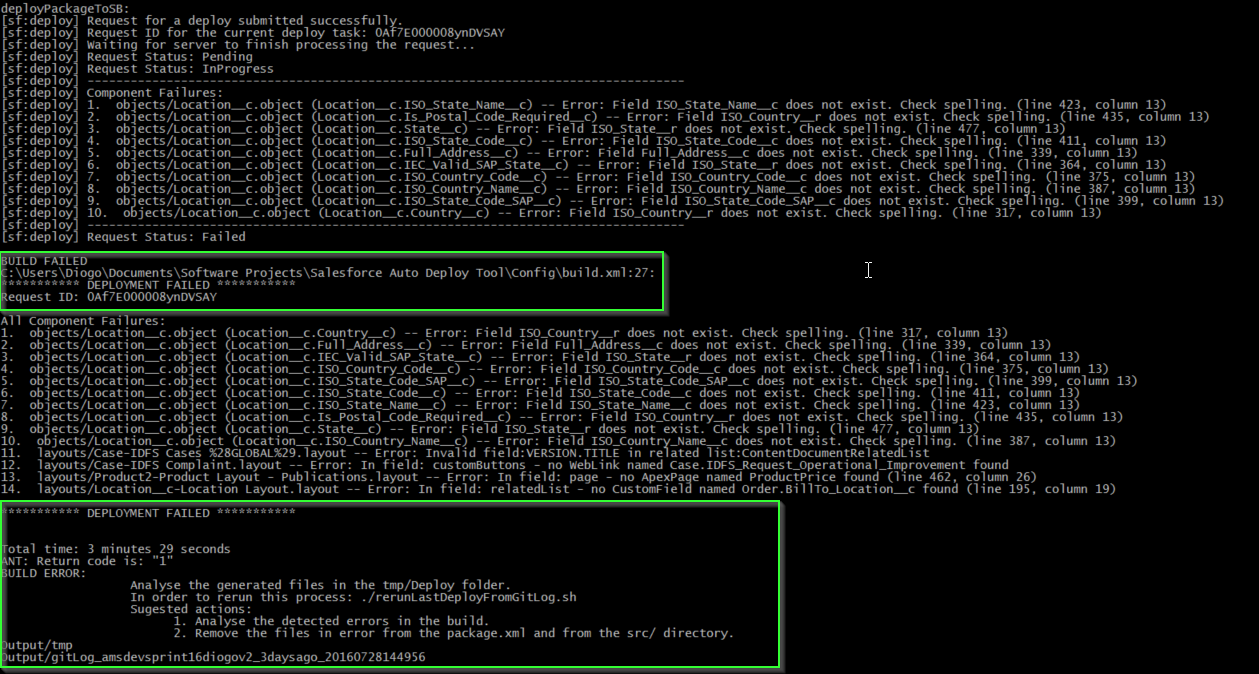
The generated result will be moved to:

**“/Salesforce Auto Deploy Tool/Output/gitLog\_[yourbranch]\_[currentDate]”**

In the result you can see all the entities recovered/generated for the deployment process.

### 3.3.2. Failed Deploy

In the case of a failed deploy (missing components, layouts, fields, objects, etc) for example:



The generated result will be moved to:

**“/Salesforce Auto Deploy Tool/Output/tmp”**

You should analyze the log to discover what caused the failure and then solve the errors accordingly.

If you need to manually modify the package.xml you have to navigate to the “**tmp/Deploy/src**” folder. In order to add/remove components you just add/remove them from the package.xml and from the files recovered.

After resolving all the issues you can rerun the deployment using command:

* **./rerunLastFailedDeploy.sh**

# 4. Implementation Details

Here it is described the implemented methods in order to improve/reuse/fix this process.

## 4.1. Configuration Files

Described above in the configuration section.

## 4.2. Bash Scripts

Three bash scripts were implemented.

### 4.2.1. runDeployFromPullRequest.sh

Steps:

1. Recover the modified files paths from the diff in the Pull Request.
2. Generates the Package.xml/destructiveChanges.xml using **xmlConstructor.py.**
3. From the Package.xml recovers the filepaths to be recovered from the Bitbucket specified branch.
4. Run the Ant Deploy for the recovered files.

### 4.2.2. runDeployFromGitLog.sh

Steps:

1. Recover the modified files paths from the Git Log.
2. Generates the Package.xml/destructiveChanges.xml using **xmlConstructor.py.**
3. From the Package.xml recovers the filepaths to be recovered from the Bitbucket specified branch.
4. Run the Ant Deploy for the recovered files.

### 4.2.3. rerunLastFailedDeploy.sh

Steps:

1. Run the Ant Deploy for the recovered files in the tmp folder. As stated above its only used in case of a failed deployment.

## 4.3. Python Scripts

Two python scripts were implemented:

* **xmlConstructor.py** 🡪 generates a package.xml based on the “git diff” provided.
* **xmlDecoder.py** 🡪 decodes the file paths based on a package.xml provided.

# 5. Future Developments

1. Improve python scripts. Create an object method to generalize the “Salesforce Object”. In this moment the Salesforce object is replicated in each method.
2. Generalize the bash scripts. Most of the steps are replicated.
3. … whatever you see fit…