

Installation Doc for Java, Ant, HomeBrew in Mac

HomeBrew Installation From <https://github.com/Homebrew/brew>

Paste the below command in mac terminal.

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```


Installation of Java in Mac via HomeBrew

Run the below mentioned commands in the terminal below:

```
brew tap adoptopenjdk/openjdk
```

```
brew install --cask adoptopenjdk11
```

You will need to add your password and then the installation will be completed! (refer screenshot)

```
manish.pal@manishp-ltmstw0 ~ % brew tap adoptopenjdk/openjdk
==> Tapping adoptopenjdk/openjdk
Cloning into '/usr/local/Homebrew/Library/Taps/adoptopenjdk/homebrew-openjdk'...
remote: Enumerating objects: 1989, done.
remote: Counting objects: 100% (53/53), done.
remote: Compressing objects: 100% (30/30), done.
remote: Total 1989 (delta 41), reused 25 (delta 23), pack-reused 1936
Receiving objects: 100% (1989/1989), 370.77 KiB | 443.00 KiB/s, done.
Resolving deltas: 100% (1421/1421), done.
Tapped 47 casks (85 files, 544.8KB).
manish.pal@manishp-ltmstw0 ~ % brew install --cask java
Error: Cask 'java' is unavailable: No Cask with this name exists.
manish.pal@manishp-ltmstw0 ~ % brew install --cask adoptopenjdk11
==> Downloading https://github.com/AdoptOpenJDK/openjdk11-binaries/releases/download
==> Downloading from https://github-releases.githubusercontent.com/140419044/156
##### 100.0%
==> Installing Cask adoptopenjdk11
==> Running installer for adoptopenjdk11; your password may be necessary.
Package installers may write to any location; options such as '--appdir' are ignored.
Password:
installer: Package name is AdoptOpenJDK
installer: Installing at base path /
installer: The install was successful.
package-id: net.adoptopenjdk.11.jdk
version: 11.0.11+9
volume: /
location:
install-time: 1619516653
 adoptopenjdk11 was successfully installed!
manish.pal@manishp-ltmstw0 ~ %
```

Download Ant See [Binary Distribution](#) for details.

1. Uncompress the downloaded file into a directory.
2. Add the bin directory to your path.

3. Set the ANT_HOME environment variable to the directory where you installed Ant.
[Note: Not able to install ANT via downloading and setting up the Path in bash_profile so going for the Brew Installation!]

cmd + shift + [.] ----> For seeing the hidden files:

cmd + shift + G ----> Enter the path location to access the folder/directory

Issues Faced during accessing the .bash_profile along with setting path variables:

```
manish.pal@manishp-ltmstw0 ~ % $HOME/.bash_profile
zsh: permission denied: /Users/manish.pal/.bash_profile
manish.pal@manishp-ltmstw0 ~ % ls -la ~ | grep bash
-rw-r--r--  1 manish.pal  staff   95 Apr 27 15:57 .bash_profile
manish.pal@manishp-ltmstw0 ~ % vi $HOME/.bash_profile
manish.pal@manishp-ltmstw0 ~ % vi $HOME/.bash_profile
manish.pal@manishp-ltmstw0 ~ % $HOME/.bash_profile
zsh: permission denied: /Users/manish.pal/.bash_profile
manish.pal@manishp-ltmstw0 ~ % echo $path
/usr/local/bin /usr/bin /bin /usr/sbin /sbin
manish.pal@manishp-ltmstw0 ~ % /etc/paths.d
zsh: permission denied: /etc/paths.d
manish.pal@manishp-ltmstw0 ~ % ANT --version
zsh: command not found: ANT
manish.pal@manishp-ltmstw0 ~ % ant
zsh: command not found: ant
manish.pal@manishp-ltmstw0 ~ % sudo open -e ~/.bash_profile
Password:
manish.pal@manishp-ltmstw0 ~ % █
```

Used below command as sudo user in order to open and access the file:

```
sudo open -e ~/.bash_profile
```

```
brew install ant. ---> Success!
```

```

[manish.pal@manishp-ltmstw0 ~ % brew reinstall ant
==> Downloading https://www.apache.org/dyn/closer.lua?path=ant/ivy/2.5.0/apache-
Already downloaded: /Users/manish.pal/Library/Caches/Homebrew/downloads/4db50de1
9f1cbf8e566630b0b94213336416c6f10533a1e84d094e2f1e2d054a--apache-ivy-2.5.0-bin.t
ar.gz
==> Downloading https://www.apache.org/dyn/closer.lua?path=commons/bcel/binaries
Already downloaded: /Users/manish.pal/Library/Caches/Homebrew/downloads/222d3991
dff4242f666efb012609c563ee961b6c39dbe96e26dcf156b030c8ba--bcel-6.5.0-bin.tar.gz
==> Downloading https://www.apache.org/dyn/closer.lua?path=ant/binaries/apache-a
Already downloaded: /Users/manish.pal/Library/Caches/Homebrew/downloads/0c2eccc
c09d1db9efa19107b5abe2aa27fbd9e5bb1ff5c7fb2c47120a37c19d--apache-ant-1.10.10-bin
.tar.xz
==> Reinstalling ant
📦 /usr/local/Cellar/ant/1.10.10: 1,671 files, 42MB, built in 6 seconds
[manish.pal@manishp-ltmstw0 ~ % ant
Buildfile: build.xml does not exist!
Build failed
[manish.pal@manishp-ltmstw0 ~ % ant -version
Apache Ant(TM) version 1.10.10 compiled on April 12 2021
manish.pal@manishp-ltmstw0 ~ % █

```

Download the ant Migration tool zip file from below link:

https://developer.salesforce.com/docs/atlas.en-us.daas.meta/daas/forcemigrationtool_install.htm

If you want to check the path where HomeBrew installed your packages:

The `/usr/local/Cellar` directory is the default location on OS X. You'll see sub-directories in there for all your installed formulae.

Add the Jar file `ant-salesforce.jar` into the directory of the ant installation (The path should look something like this see below)

`/usr/local/Cellar/ant/1.10.10/libexec`

Now let's see how this can be used with Salesforce:

When you give Ant a command, it will read the `build.xml` configuration file, searching for the target matching that name and once it finds it, it will execute it.

We may wonder how ANT connects to our target Salesforce instance, that's where `build.properties` file comes into the picture

i.e. `build.properties` where you can specify the Salesforce instance details such as test | login, username and password with security token.

```

Users > manish.pal > Downloads > Salesforce ant migration tool > salesforce_ant_51.0 > sample > build.properties
1  # build.properties
2  #
3
4  # Specify the login credentials for the desired Salesforce organization
5  sf.username = <Insert your Salesforce username here>
6  sf.password = <Insert your Salesforce password here>
7  sf.sessionId = <Insert your Salesforce session id here. Use this or username/password above. Cannot use both>
8  sf.pkgName = <Insert comma separated package names to be retrieved>
9  sf.zipFile = <Insert path of the zipfile to be retrieved>
10 sf.metadataType = <Insert metadata type name for which listMetadata or bulkRetrieve operations are to be performed>
11
12 # Use 'https://login.salesforce.com' for production or developer edition (the default if not specified).
13 # Use 'https://test.salesforce.com' for sandbox.
14 sf.serverurl = https://login.salesforce.com
15
16 sf.maxPoll = 20
17 # If your network requires an HTTP proxy, see http://ant.apache.org/manual/proxy.html for configuration.
18 #

```

build.xml File

```

1 <project name="Sample usage of Salesforce Ant tasks" default="test" basedir="." xmlns:sf="antlib:com.salesforce">
2
3   <property file="build.properties"/>
4   <property environment="env"/>
5
6   <!-- Setting default value for username, password and session id properties to empty string
7    so unset values are treated as empty. Without this, ant expressions such as ${sf.username}
8    will be treated literally. -->
9
10  <condition property="sf.username" value=""><not><isset property="sf.username"/></not></condition>
11  <condition property="sf.password" value=""><not><isset property="sf.password"/></not></condition>
12  <condition property="sf.sessionId" value=""><not><isset property="sf.sessionId"/></not></condition>
13
14  <taskdef resource="com/salesforce/antlib.xml" uri="antlib:com.salesforce">
15    <classpath>
16      <pathelement location="..ant-salesforce.jar" />
17      <!-- <pathelement location="..Users/manish.pal/Downloads/Salesforce_ant_migration_tool/ant-salesforce.jar" /> -->
18    </classpath>
19  </taskdef>
20
21  <!-- Test out deploy and retrieve verbs for package 'mypkg' -->
22  <target name="test">
23    <!-- Upload the contents of the "mypkg" package -->
24    <sf:deploy username="${sf.username}" password="${sf.password}" sessionId="${sf.sessionId}" serverurl="${sf.serverurl}" maxPoll="${sf.maxPoll}" deployRoot="mypkg" rollbackOnFailure="true">
25      <mkdir dir="retrieveOutput"/>
26      <!-- Retrieve the contents into another directory -->
27      <sf:retrieve username="${sf.username}" password="${sf.password}" sessionId="${sf.sessionId}" serverurl="${sf.serverurl}" maxPoll="${sf.maxPoll}" retrieveTarget="retrieveOutput">
28      </sf:retrieve>
29    </sf:deploy>

```

Points and loads the build properties file

Referring to your Org's credentials

setting Classpath for salesforce's ant extension library

Once the components are identified, we can make a list and put it into the package.xml file when we run ANT it will check for the package.xml file and list of components.

See below screenshot for reference, you will get to know that we've to specify the components in element you can either specify individual component names or specify * for all components for that type like (ApexClass, ApexTrigger, Pages, and so on).

```

Users > manish.pal > Downloads > Salesforce_ant_migration_tool > salesforce_ant_51.0 > sample > codepkg > package.xml > ...
1 <?xml version="1.0" encoding="UTF-8"?>
2 <Package xmlns="http://soap.sforce.com/2006/04/metadata">
3   <types>
4     <members>Employee</members>
5     <members>EmployeeTestClass</members>
6     <name>ApexClass</name>
7   </types>
8   <version>51.0</version>
9 </Package>
10

```

For more info on component types (https://developer.salesforce.com/docs/atlas.en-us.api_meta.meta/api_meta/manifest_samples.htm)

Now we need to retrieve the data from 1st org and will be deploying it to another org.

retrieve data

Now the first step is to first retrieve Code from my DevOrg1. to do this, open the command prompt and navigate to the path/directory where you want to retrieve the code.

Let's see how that works:-Let's observe and understand build.xml tags, in this case we first understand the structure of retrieveCode tag.

```
<!-- Shows retrieving code; only succeeds if done after deployCode -->
<target name="retrieveCode">
  <!-- Retrieve the contents listed in the file codepkg/package.xml into the codepkg directory -->
  <sf:retrieve username="${sf.username}" password="${sf.password}" sessionId="${sf.sessionId}" serverurl="${sf.serverurl}" maxPoll="${sf.maxPoll}" retrieveTarget="codepkg" unpackaged="codepkg/package.xml"/>
</target>
```

Two Important Attributes are

retrieveTarget → where exactly (directory name) you want to retrieve your code.

unpackaged → this attribute is to specify the package.xml itself (I am specifying it inside the codepkg folder itself)

Now we can execute the ant command to retrieve all components, ant retrieveCode

once you hit enter it will, it will fetch all the components mentioned in the package.xml and put those into the target folder.

```
manish.pal@manishp-ltmstw0 sample % ant retrieveCode
Buildfile: /Users/manish.pal/Downloads/Salesforce_ant_migration_tool/salesforce_ant_51.0/sample/build.xml

retrieveCode:
[sf:retrieve] Request for a retrieve submitted successfully.
[sf:retrieve] Request ID for the current retrieve task: 09S2w0000043potEAA
[sf:retrieve] Waiting for server to finish processing the request...
[sf:retrieve] Request Status: Succeeded
[sf:retrieve] Retrieve warnings (3):
[sf:retrieve] package.xml - Entity of type 'ApexClass' named 'SampleDeployClass' cannot be found
[sf:retrieve] package.xml - Entity of type 'ApexClass' named 'SampleFailingTestClass' cannot be found
[sf:retrieve] package.xml - Entity of type 'ApexTrigger' named 'SampleAccountTrigger' cannot be found
[sf:retrieve] Finished request 09S2w0000043potEAA successfully.

BUILD SUCCESSFUL
Total time: 3 seconds
manish.pal@manishp-ltmstw0 sample % ant retrieveCode
Buildfile: /Users/manish.pal/Downloads/Salesforce_ant_migration_tool/salesforce_ant_51.0/sample/build.xml

retrieveCode:
[sf:retrieve] Request for a retrieve submitted successfully.
[sf:retrieve] Request ID for the current retrieve task: 09S2w0000043puIEAQ
[sf:retrieve] Waiting for server to finish processing the request...
[sf:retrieve] Request Status: Succeeded
[sf:retrieve] Finished request 09S2w0000043puIEAQ successfully.

BUILD SUCCESSFUL
Total time: 4 seconds
manish.pal@manishp-ltmstw0 sample %
```

Since we fetched all the related components in a folder now, we want to copy the same code to my second org. In order to that follow the below steps:

Open the Build.properties file and replace the org credentials, with the 2nd Org credentials.
Goto build.xml to see the configuration for the deployCode tag. <Screenshot below>

```
<!-- Shows deploying code & running tests for code in directory -->
<target name="deployCode">
  <!-- Upload the contents of the "codepkg" directory, running the tests for just 1 class -->
  <sf:deploy username="${sf.username}" password="${sf.password}" sessionId="${sf.sessionId}" serverurl="${sf.serverurl}" maxPoll="${sf.maxPoll}" deployRoot="codepkg" testLevel="RunSpecifiedTests" rollbackOnError="
    <!-- <runTest>SampleDeployClass</runTest> -->
    <runTest>EmployeeTestClass</runTest>
  </sf:deploy>
</target>
```

Now let's deploy the code to the other org.

Open the command prompt and navigate until the you source code folder (where we saved our code).