

# The Apache Ant Project

A Dive Into A Powerful Scripting Tool



# Apache Ant

**THE APACHE ANT PROJECT**

Home | **Project**

- Apache Ant
  - **Welcome**
  - License
  - News
  - Security Reports
- Documentation
  - Manual 1.10.x
  - Manual 1.9.x
  - Related Projects
  - External Tools and Tasks
  - Resources
  - Frequently Asked Questions
  - Wiki
  - Having Problems?
- Download
  - Binary Distributions
  - Source Distributions
  - Ant Manual
- Contributing
  - Mailing Lists
  - Git Repositories
  - Subversion Repositories
  - Nightly-Continuous Builds
  - Bug Database
  - Security
- Sponsorship
  - Thanks
  - Sponsorship
- Project Management
  - Contributors
  - Apache Ant Mailing
  - Project Bylaws
  - Legal
  - Processes
  - Archive

**Welcome**

**Apache Ant™**

Apache Ant is a Java library and command-line tool whose mission is to drive processes described in one XML file. The main known usage of Ant is the build of Java applications. Ant supplies a rich set of Java applications. Ant can also be used effectively to build non-Java applications, for instance type of process which can be described in terms of targets and tasks.

Ant is written in Java. Users of Ant can develop their own "antlibs" containing Ant tasks and tasks. Ant is extremely flexible and does not impose coding conventions or directory layouts to the user.

Software development projects looking for a solution combining build tool and dependency management.

The Apache Ant project is part of the [Apache Software Foundation](#).

**Apache Ant 1.9.11 and 1.10.3**

**Mar 27, 2018 - Apache Ant 1.9.11 and 1.10.3 Released**

Apache Ant 1.9.11 and 1.10.3 are now available for download as source or binary from <https://ant.apache.org>.

The Apache Ant team currently maintains two lines of development. The 1.9.x releases require Java 1.5 and the 1.10.x releases are mostly bug fix releases while additional releases are required to use versions of Java prior to Java8 during the build process.

Ant 1.10.3 contains a superset of 1.9.11 - with the exception of a few tasks and features that were removed.

Both releases are mostly bug fix releases with a few new features being added. Ant 1.10.2 introduced support for JUnit5 in the form of the `JUnit5Runner` task. The new `JUnit5Runner` task will be added in upcoming releases.

**EasyAnt retired**

**Dec 13, 2016 - EasyAnt retired**

The Ant PMC [voted](#) to archive the EasyAnt subproject and all its modules. This means that all development will be done in the main Ant project. It also means that, if a community grows, the subproject could be [reactivated](#).

**Compress Ant Library 1.5**

**June 13, 2017 - Apache Compress Ant Library 1.5 Available**

Apache Compress Ant Library 1.5 is now available for download as [binary](#) or [source](#) releases.

This release adapts to the 1.14 release of Apache Commons Compress and now adds read-only support for Snappy and LZMA in addition to the read-only support offered by version 1.4.

**Apache Ivy 2.4.0**

- Ant is a java library and command line tools focused on building and deploying files.
- It is extensible and has grown to far more than a build tool.
- It is an active project that was originally released in 2000.



# Our Focus

- Installation/Getting Started
- General format and properties
- Copying files
- Tar/Zip files
- Variables
- Secure copy
- Executing SQL Scripts
- Using Git
- Firing off Javadoc
- Executing Tests



# Installation

- Download the latest version at: <https://ant.apache.org/bindownload.cgi>
- It is a Java application so you need a minimum JVM as well. Looks for that in the requirements. Most current ones need at least Java 5.
- Note where you install and configure the environment variable `ANT_HOME` to make things easier.



# General Format

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="Develpreneur" default="all" basedir=".">
  <target name="Basic" description="This is just a very simple task to display
a message">
    <echo message="About to run the most basic script."/>
    <tstamp/>
    <echo message="Finished build at ${TSTAMP}"/>
  </target>
  <target name="clean" description="Remove any non xml files just to be safe">
    <echo message="Removing extra files (not *.xml) ..."/>
    <delete>
      <fileset dir=".">
        <include name="*.backup"/>
      </fileset>
    </delete>
  </target>
  <target name="all" description="run everything" depends="Basic,clean">
    <echo message="Finished build at ${TSTAMP}"/>
  </target>
</project>
```



# An XML File

- Project: Name, Default, Basedir
- Echo sends a message to the screen
- Target descriptions to help plus main vs other tasks
- Depends to link tasks together (runs left to right)



# Copying Files

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="Develpreneur" default="all" basedir=".">
  <target name="Basic" description="This is just a very simple task to display a message">
    <echo message="About to run the most basic script."/>
    <tstamp/>
    <echo message="Finished build at ${TSTAMP}"/>
  </target>
  <target name="clean" description="Remove any non xml files just to be safe">
    <echo message="Removing extra files (not *.xml) ..."/>
    <delete>
      <fileset dir=".">
        <include name="*.backup"/>
      </fileset>
    </delete>
  </target>
  <target name="all" description="run everything" depends="Basic,clean">
    <echo message="Finished build at ${TSTAMP}"/>
  </target>
  <target name="backup" description="copy the basic.xml file to backup" depends="clean">
    <copy file="basic.xml" tofile="basic.backup"/>
  </target>
  <target name="backupchange" description="copy the basic.xml file to backup if it is newer">
    <copy file="basic.xml" tofile="basic.backup"/>
  </target>
  <target name="copynewdir" description="Copy all xml files to a new folder">
    <copy failonerror="false" todir="newfolder">
      <fileset dir=".">
        <include name="**/*.xml"/>
        <exclude name=".backup"/>
      </fileset>
    </copy>
  </target>
</project>
```



# Group/Compress

```
<target name="pkg" description="Package files into tarball"
depends="Basic,clean">
  <echo message="Packaging files..."/>
  <delete file="./pkg.tar"/>
  <tar destfile="./pkg.tar" basedir="."/>
</target>
<target name="zip" description="Package files into zipfile"
depends="Basic,clean">
  <echo message="Zipping files..."/>
  <delete file="./pkg.zip"/>
  <zip destfile="./pkg.zip" basedir="."/>
</target>
```





# Script Variables

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="Develpreneur" default="all" basedir=".">
  <property name="src.dir" value="."/>
  <property name="target.dir" value="newfolder"/>
  <property name="server" value="rob@rbprod:/home/rob"/>
  <target name="Basic" description="This is just a very simple task to
display a message">
    <echo message="About to run the most basic script."/>
    <tstamp/>
    <echo message="Finished build at ${TSTAMP}"/>
  </target>
  <target name="clean" description="Remove any non xml files just to be
safe">
    <echo message="Removing extra files (not *.xml) ..."/>
    <delete>
      <fileset dir="${src.dir}">
        <include name="*.backup"/>
      </fileset>
    </delete>
  </target>
</project>
```



# Ant Libraries

- SCP requires the ant-jsch.jar file.
- One location is <http://www.java2s.com/Code/Jar/a/Downloadantjsch181jar.htm>
- Copy the jar file into the ant/lib folder
- This is where you put database drivers as well



# An SCP Example

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="TimeDemo">
  <!-- Loading application directory properties -->
  <property name="deploy.dir" value="/Library/WebServer/Documents/TimeDemo"/>
  <property name="src.dir" value="/Users/robbroadhead/Coding/TimeDemo/WebMain"/>
  <property name="root.dir" value="/Users/robbroadhead/Coding/TimeDemo"/>
  <property name="repo" value="https://robbroadhead@bitbucket.org/robbroadhead/timematters.git"/>
  <property name="server" value="rob@mytime:/home/rob"/>
  <target name="all" depends="clean,deploy">
    <echo message="Ran full build."/>
    <tstamp/>
    <echo message="Finished build at ${TSTAMP}"/>
  </target>
  <target name="deploy" description="Copy the changed files to the target">
    <copy failonerror="false" todir="${deploy.dir}">
      <fileset dir="${src.dir}">
        <include name="**/*"/>
        <exclude name=".*"/>
      </fileset>
    </copy>
  </target>
  <target name="pkg" description="Package files into tarball" depends="deploy">
    <echo message="Packaging files..."/>
    <delete file="${root.dir}/timematters.tar"/>
    <tar destfile="${root.dir}/timematters.tar" basedir="${deploy.dir}">
    <echo message="Sending files..."/>
    <scp file="${root.dir}/timematters.tar" todir="${server}" keyfile="/Users/robbroadhead/.ssh/id_rsa"
trust="true"/>
  </target>
  <target name="clean" description="Remove the files just to be safe">
    <echo message="Removing old files..."/>
    <delete>
      <fileset dir="${deploy.dir}">
        <exclude name="images/*"/>
      </fileset>
    </delete>
  </target>
</project>
```



## Description

Copies a file or FileSet to or from a (remote) machine running an SSH daemon. FileSet *only* works for copying files from the local machine to a remote machine.

**Note:** This task depends on external libraries not included in the Ant distribution. See [Library Dependencies](#) for more information. This task has been tested with jsch-0.1.2 and later.

See also the [sshexec task](#)

## Parameters

Attribute	Description	Required
<i>file</i>	The file to copy. This can be a local path or a remote path of the form <code>user[:password]@host:/directory/path</code> . <code>password</code> can be omitted if you use key based authentication or specify the <code>password</code> attribute. The way remote path is recognized is whether it contains "@" character or not. This will not work if your <code>localPath</code> contains "@" character.	Yes, unless a nested <code>&lt;fileset&gt;</code> element is used
<i>localFile</i>	This is an alternative to the <code>file</code> attribute. But this must always point to a local file. The reason this was added was that when you give <code>file</code> attribute it is treated as remote if it contains "@" character. This character can exist also in local paths. <i>since Ant 1.6.2</i>	Alternative to <code>file</code> attribute
<i>remoteFile</i>	This is an alternative to the <code>file</code> attribute. But this must always point to a remote file. <i>since Ant 1.6.2</i>	Alternative to <code>file</code> attribute
<i>toDir</i>	The directory to copy to. This can be a local path or a remote path of the form <code>user[:password]@host:/directory/path</code> . <code>password</code> can be omitted if you use key based authentication or specify the <code>password</code> attribute. The way remote path is recognized is whether it contains "@" character or not. This will not work if your <code>localPath</code> contains "@" character.	Yes
<i>localToDir</i>	This is an alternative to the <code>toDir</code> attribute. But this must always point to a local directory. The reason this was added was that when you give <code>toDir</code> attribute it is treated as remote if it contains "@" character. This character can exist also in local paths. <i>since Ant 1.6.2</i>	Alternative to <code>toDir</code> attribute
<i>localToFile</i>	Changes the file name to the given name while receiving it, only useful if receiving a single file. <i>since Ant 1.6.2</i>	Alternative to <code>toDir</code> attribute
<i>remoteToDir</i>	This is an alternative to the <code>toDir</code> attribute. But this must always point to a remote directory. <i>since Ant 1.6.2</i>	Alternative to <code>toDir</code> attribute
<i>remoteToFile</i>	Changes the file name to the given name while sending it, only useful if sending a single file. <i>since Ant 1.6.2</i>	Alternative to <code>toDir</code> attribute
<i>port</i>	The port to connect to on the remote host.	No; defaults to "22"
<i>trust</i>	This trusts all unknown hosts if set to "yes" or "true". <b>Note:</b> If you set this to "false" (the default), the host you connect to must be listed in your <code>knownhosts</code> file, this also implies that the file exists.	No; defaults to "no"
<i>knownhosts</i>	This sets the known hosts file to use to validate the identity of the remote host. This must be a SSH2 format file. SSH1 format is not supported.	No; defaults to <code>\$ {user.home} /.ssh/known_hosts</code>
<i>failonerror</i>	Whether to halt the build if the transfer fails.	No; defaults to "true"
<i>password</i>	The password.	Yes, unless you are using key based authentication or the password has been given in the <code>file</code> or <code>toDir</code> attribute
<i>keyfile</i>	Location of the file holding the private key.	Yes, if you are using key based authentication
<i>passphrase</i>	Passphrase for your private key.	No; defaults to an empty string
<i>verbose</i>	Determines whether SCP outputs verbosely to the user. Currently this means outputting dots/stars showing the progress of a file transfer. <i>since Ant 1.6.2</i>	No; defaults to "false"
<i>sftp</i>	Determines whether SCP uses the sftp protocol. The sftp protocol is the file transfer protocol of SSH2. It is recommended that this be set to "true" if you are copying to/from a server that doesn't support scp1. <i>since Ant 1.7</i>	No; defaults to "false"
<i>preserveLastModified</i>	Determines whether the last modification timestamp of downloaded files is preserved. It only works when transferring from a remote to a local system and probably doesn't work with a server that doesn't support SSH2. <i>since Ant 1.8.0</i>	No; defaults to "false"
<i>filemode</i>	A 3 digit octal string, specify the user, group and other modes in the standard Unix fashion. Only applies to uploaded files. Note the actual permissions of the remote file will be governed by this setting and the <code>umask</code> on the remote server. <i>since Ant 1.9.5</i>	No; default is "644"
<i>dirmode</i>	A 3 digit octal string, specify the user, group and other modes in the standard Unix fashion. Only applies to uploaded dirs. Note the actual permissions of the remote dir will be governed by this setting and the <code>umask</code> on the remote server. <i>since Ant 1.9.5</i>	No; default is "755"
<i>serverAliveIntervalSeconds</i>	Sets a timeout interval in seconds after which if no data has been received from the server, the task will send a message through the encrypted channel to request a response from the server. <i>since Ant 1.9.7</i>	No, the default is "0", indicating that these messages will not be sent to the server
<i>serverAliveCountMax</i>	The number of server alive messages which may be sent without receiving any messages back from the server. Only used if <code>serverAliveIntervalSeconds</code> is not "0". <i>since Ant 1.9.7</i>	No; defaults to "3"
<i>compressed</i>	Whether to enable compression during transfer. <i>since Ant 1.9.8</i>	No; defaults to "false"



***“When in doubt, do a search on ‘[task name] ant task’ to find the documentation and examples quickly. Most commands are very similar in properties and options”***



# A SQL Example

```
<target name="dbreset" description="Drop and rebuild the database
from the script">
  <echo message="Dropping and rebuilding the database"/>
  <sql driver="com.mysql.jdbc.Driver" url="jdbc:mysql://127.0.0.1/
mydb" userid="dbuser" password="password123" src="${src.dir}/sql/
MyDBSource.sql"/>
  <sql driver="com.mysql.jdbc.Driver" url="jdbc:mysql://127.0.0.1/
mydb" userid="dbuser" password="password123"><![CDATA[
update some_table set column1 = column1 + 1 where column2 < 42;
]]></sql>
</target>
```



# A Git Example

```
<!-- Git Marcos-->
<macrodef name="git">
  <attribute name="command"/>
  <attribute name="dir" default=""/>
  <element name="args" optional="true"/>
  <sequential>
    <echo message="git @{{command}}"/>
    <exec executable="git" dir="@{{dir}}">
      <arg value="@{{command}}"/>
      <args/>
    </exec>
  </sequential>
</macrodef>
<macrodef name="git-clone-pull">
  <attribute name="repository"/>
  <attribute name="dest"/>
  <sequential>
    <git command="clone">
      <args>
        <arg value="@{{repository}}"/>
        <arg value="@{{dest}}"/>
      </args>
    </git>
    <git command="pull" dir="@{{dest}}"/>
  </sequential>
</macrodef>
<target name="grabsource" description="Pull files from bitbucket">
  <echo message="Getting files from repository ${repo}"/>
  <git-clone-pull repository="${repo}" dest="source"/>
  <echo message="Finished pull at ${TSTAMP}"/>
</target>
```



# Javadoc Example

```
<target name="doc" description="generate documentation">
  <javadoc packagenames="com.*" sourcepath="${java.dir}"
  destdir="${doc.dir}" author="true" version="true" use="true"
  windowtitle="Test API">
    <doctitle><![CDATA[<h1>Ant Tutorial Docs</h1>]]></
  doctitle>
    <bottom><![CDATA[<i>Copyright &#169; 2018 My Corp. All
  Rights Reserved.</i>]]></bottom>
    <tag name="todo" scope="all" description="To do:"/>
    <link href="https://docs.oracle.com/javase/8/docs/api/" /
  >
    </javadoc>
  </target>
```





# JUnit Example

```
<junit printsummary="yes" fork="yes" haltonfailure="yes">  
  <formatter type="plain"/>  
  <test name="my.app.TestCase1"/>  
  <test name="my.app.TestCase2"/>  
</junit>
```



# Masking Your Scripts



- Take it step by step, add a task here or there. No need to go crazy.
- Get comfortable with some core tasks.
- Expand as needed.
- The documentation and examples are your friends.



# Excellent Uses

- Backup scripts
- Database updates, imports and exports
- Automating common admin tasks
- Easier than shell scripts and can be as powerful
- Cross platform so perfect when developing and deploying on different ones



# Thank You!

I appreciate your time and would love to discuss any of this further. You can send questions, comments and suggestions through any of these methods.

- [info@develpreneur.com](mailto:info@develpreneur.com)
- <https://develpreneur.com/contact-us>
- @develpreneur
- <https://www.facebook.com/Develpreneur>

Our goal is building better developers from where they are today for a better tomorrow.

