Zero downtime deployment to Tomcat 7 with Maven

This is just one way of deploying to Tomcat, not the only one
Similar to the "sister guide" (Developing with Tomcat and Eclipse), there's more than one way to skin the deployment cat. With Maven, good options are the Tomcat plugin and the Cargo plugin. Modern PaaS platforms also typically have their own cluster deployment tools. However, this guide is specifically for Tomcat 7 using its parallel deployment feature. - Kalle Korhonen / tynamo.org

Tomcat 7 added a superb but rarely used feature, called parallel deployment. With enough memory on the system, you could approach poor man's high availability with parallel deployment but where it really shines is continuous integration and continuous deployments to your alpha/QA systems. It completely sucks if your development team/CI system deploys a new version to an alpha server 60 or more times a day and you need to reload the web application every time (or worse yet, restart the whole server) while at the same time, your QA is desperately trying to use the same system to verify features and test user experience. With parallel deployment, existing users can keep using the same version of the web application until their session expires while others can be simultaneously using newer versions.

However, parallel deployment and tools for it are surprisingly rough on the edges. I haven't found any Maven plugins that would automatically take advantage of the feature so I hacked one together, using Ant so those disliking Maven can also enjoy the feature 🙂.

```
<profile>
            <!-- to be used with jenkins continuous builds only -->
            <id>continuous-delivery</id>
            <build>
                <plugins>
                    <plugin>
                        <artifactId>maven-antrun-plugin</artifactId>
                        <version>1.7</version>

                        <dependencies>
                            <dependency>
                                <groupId>ant-contrib</groupId>
                                <artifactId>ant-contrib</artifactId>
                                <version>1.0b3</version>
                                <exclusions>
                                    <exclusion>
                                        <groupId>ant</groupId>
                                        <artifactId>ant</artifactId>
                                    </exclusion>
                                </exclusions>
                            </dependency>
                            <dependency>
                                <groupId>org.apache.ant</groupId>
                                <artifactId>ant-jsch</artifactId>
                                <version>1.8.3</version>
                            </dependency>
                            <!--  we are getting random
                            An Ant BuildException has occured:
com.jcraft.jsch.JSchException: Auth fail around Ant part
                            ...<scp todir exceptions when scping to war to alpha. Try
out a newer JSch fork from
                            https://github.com/vngx/vngx-jsch as a remedy or at least
to provide better error messages <dependency>
                                <groupId>com.jcraft</groupId>
                                <artifactId>jsch</artifactId>
                                <version>0.1.48</version>
                            </dependency>
                            -->
                            <dependency>
                                <groupId>org.vngx</groupId>
                                <artifactId>vngx-jsch</artifactId>
                                <version>0.9</version>
```

```
                            </dependency>
                        </dependencies>
                        <executions>
                            <execution>
                                <id>prepend-war-with-build-number</id>
                                <phase>validate</phase>
                                <configuration>
                                    <exportAntProperties>true</exportAntProperties>
                                    <target>
                                        <taskdef
resource="net/sf/antcontrib/antcontrib.properties" />
                                        <!-- strategy: add lots of zeroes, then
truncate to fixed width. Replace doesn't seem to allow full regex so do
                                            it in two parts -->
                                        <propertyregex property="prefixedBuildNumber"
input="${env.BUILD_NUMBER}" regexp="^([0-9])+?" replace="00000\1" override="true" />
                                        <propertyregex property="formattedBuildNumber"
input="${prefixedBuildNumber}" regexp="(\d{5}$)" select="\1" override="true" />
                                    </target>
                                </configuration>
                                <goals>
                                    <goal>run</goal>
                                </goals>
                            </execution>
                            <execution>
                                <id>rename-war-for-parallel-deployment</id>
                                <phase>install</phase>
                                <configuration>
                                    <target>
                                        <fail unless="formattedBuildNumber"
message="Environment variable BUILD_NUMBER is not available. Is this target running on
CI system?" />
                                        <!-- The line below just for local testing on
ci machine, we want to deploy remotely via ssh to alpha environment
                                            <copy
file="${project.build.directory}/myapp.war"
tofile="/usr/share/tomcat/webapps/myapp##${formattedBuildNumber}.war" /> -->
                                        <scp keyfile="${user.home}/.ssh/id_rsa"
file="${project.build.directory}/${project.build.finalName}.war"

todir="tomcat@alpha.myapp.com:/opt/tomcat/webapps/myapp##${formattedBuildNumber}.war.n
ew" />
                                        <sshexec host="alpha.myapp.com"
username="tomcat" keyfile="${user.home}/.ssh/id_rsa"
                                        command="mv
/opt/tomcat/webapps/myapp##${formattedBuildNumber}.war.new
/opt/tomcat/webapps/ROOT##${formattedBuildNumber}.war" />
                                        <!--  While waiting for
https://issues.apache.org/bugzilla/show_bug.cgi?id=52777 to be resolved, manually
delete older ones -->
                                        <sshexec host="alpha.myapp.com"
username="tomcat" keyfile="${user.home}/.ssh/id_rsa"
                                        command="ls /opt/tomcat/webapps/ROOT*.war |
sort -r | tail -n+2 | xargs rm" />
                                    </target>
                                </configuration>
                                <goals>
                                    <goal>run</goal>
                                </goals>
```

```
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
```

```
        </profile>
```

Little convoluted perhaps readable and straightforwarded. The important thing to note is that Tomcat's parallel deployment feature relies on special naming convention of the war file, with double hash (#) character separating the version from the context/war file name. Above, I'm using the BUILD_NUMBER variable as generated by (Jenkins) CI system but you are of course free to use any other version identifier, however Tomcat only understands alphabetical ordering of version identifiers. Another thing to note is that currently the previous versions are not unloaded by default but just passivated. This causes the memory to fairly quickly run out which is why I'm simply manually deleting the previous war files that causes Tomcat to undeploy and remove their exploded webapp folders. For continuous delivery, you most likely want to set **Host attribute undeployOldVersions** for removing previous instances, i.e. have something like this in your Tomcat's server.xml configuration file:

```
<Host name="localhost"  appBase="webapps" unpackWARs="true" autoDeploy="true"
undeployOldVersions="true">
```

Finally one more note about the parallel deployment is that if loading the new version of a webapp fails, it'll never become active, so you always have a version of the webapp in a runnable state.