How To Contribute

This page describes the mechanics of *how* to contribute software to Apache Whirr.

Getting the source code

First of all, you need the Whirr source code. Checkout git "trunk" using:

```
git clone git://git.apache.org/whirr.git
```

Setting up Eclipse

Generate the Eclipse project and classpath files:

```
mvn eclipse:eclipse -DdownloadSources=true -DdownloadJavadocs
```

(**NOTE:** If this **fails** because whirr jars aren't found in a remote repo, run `mvn install` first as described in the next section then retry.)

In Eclipse, set the `M2_REPO` classpath variable:

1. File, Preferences, Java, Build Path, Classpath Variables
2. New

- Name: M2_REPO
- Path: /path/to/.m2/repository

Then import all projects:

1. File, Import, General, Maven, Existing Maven Projects
2. Choose the *whirr-trunk* directory
3. Choose all modules

It's also convenient to add the top-level directory as an Eclipse project too.

Building Whirr

Build and install the JARs from the top-level with the following

```
mvn install
```

To check the code adheres to the style guidelines, which are Sun's conventions except 2 spaces for tabs, http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html, you can use

```
mvn checkstyle:checkstyle
```

Normally you don't need to deploy to remote repositories (this is covered in How To Release), but you can test local deployment with

```
mvn deploy -Ppackage -Pdeploy -Pjavadoc -DaltDeploymentRepository=id::default::file:target/deploy
```

Running Tests

Unit Tests

You can run the unit tests, which run locally without using any cloud providers, by typing:

```
mvn test
```

Integration Tests

To run integration tests you need to have an account with a cloud provider, and you need to set the necessary credentials. In theory, Whirr runs against any provider supported by jclouds, but this hasn't been tested yet.

When running in the cloud, you can avoid problems if RAM resources are slim by setting env var:

```
export MAVEN_OPTS=-Xmx200m
```

and by running the mvn commands below like

```
nice -n 19 mvn ...rest of args
```

; when tests are run with no specified heap limit, it can result in swapping and then ssh lockout.

To only run the tests for a given service, change into the services/<service> directory and type the following. Alternatively, run this from the top level to run the tests for all services.

```
mvn verify -Pintegration \
  -DargLine="-Dwhirr.test.provider=<cloud-provider> -Dwhirr.test.identity=<cloud-provider-user> -Dwhirr.test.
credential=<cloud-provider-secret-key>"
```

For Amazon EC2, `whirr.test.provider` should be set to `aws-ec2` (the default). The identity is the access key ID, and the credential is the secret access key.

The tests also rely on having an SSH keypair. By default they use *.ssh/id_rsa* and *.ssh/id_rsa.pub* in the user's home directory, but you can override this by setting by providing an extra config file as follows. Here we specify a file called *.whirr-test.properties* in our home directory.

```
mvn verify -Pintegration \
  -DargLine="-Dwhirr.test.provider=<cloud-provider> -Dwhirr.test.identity=<cloud-provider-user> -Dwhirr.test.
credential=<cloud-provider-secret-key> -Dconfig=.whirr-test.properties"
```

The property whirr.test.provider must be set on the command line as specified; it does not work to set it in the config= file.

Integration tests test the correctness of services on different providers. To test performance have a look at Running Benchmarks.

Contributing your work

Patches should be "attached" to an issue report in JIRA via the "'Attach File"' link on the issue's Jira. Please note that the attachment should be granted license to ASF for inclusion in ASF works (as per the Apache License).

You can also use the Apache Review Board instance at https://reviews.apache.org/ for review, although this is optional, and you still need to attach the patch to JIRA.

When you believe that your patch is ready to be committed, select the "'Submit Patch"' link on the issue's Jira.

You should run

```
mvn clean checkstyle:checkstyle apache-rat:check test
```

before selecting "'Submit Patch"'. Tests should all pass. You should also run integration tests if your changes may affect them. Submitting patches that fail tests is frowned on (unless the failure is not actually due to the patch).

If your patch involves performance optimizations, they should be validated by benchmarks that demonstrate an improvement.

If your patch creates an incompatibility with the latest major release, then you must set the '"Incompatible change"' flag on the issue's Jira 'and' fill in the '"Release Note"' field with an explanation of the impact of the incompatibility and the necessary steps users must take.

If your patch implements a major feature or improvement, then you must fill in the '"Release Note"' field on the issue's Jira with an explanation of the feature that will be comprehensible by the end user.

Once you have submitted your patch, it will go into the Whirr Review Queue, then a committer should evaluate it within a few days and either: commit it; or reject it with an explanation.

Please be patient. Committers are busy people too. If no one responds to your patch after a few days, please make friendly reminders. Please incorporate other's suggestions into your patch if you think they're reasonable. Finally, remember that even a patch that is not committed is useful to the community.

Should your patch be rejected, select the '"Resume Progress"' on the issue's Jira, upload a new patch with necessary fixes, and then select the '"Submit Patch"' link again.

In many cases a patch may need to be updated based on review comments. In this case the updated patch should be re-attached to the Jira with the name name. Jira will archive the older version of the patch and make the new patch the active patch. This will enable a history of patches on the Jira. Patch naming is generally WHIRR-#.patch where WHIRR-# is the id of the Jira issue.

Committers: for non-trivial changes, it is best to get another committer to review your patches before commit. Use **Submit Patch** link like other contributors, and then wait for a "+1" from another committer before committing. Please also try to frequently review things in the patch queue.

Post commit

After a patch has been committed, Hudson will run the unit tests.

Committing Guidelines for committers

Apply the patch uploaded by the contributor. Edit the `CHANGES.txt` file, adding a description of the change, the bug number it fixes, and the contributor's name. Add it to the appropriate section - INCOMPATIBLE CHANGES, NEW FEATURES, IMPROVEMENTS, BUG FIXES. Please follow the format in `CHANGES.txt` file. When adding an entry please add it to the end of a section. Use the same entry for the svn commit message.