Extend Jenkins

General Contribution
Want to help but don’t know what to do? You can help to extend and enrich the Jenkins community even if you don’t necessarily write code. Here is the Beginners Guide to Contributing. The latest snapshots of Jenkins and its plugins can be found at the official Jenkins CI server.

Developing Plugins
Jenkins supports plugins, which allow Jenkins to be extended to meet specific needs of individual projects. Read on to learn how to write a plugin, especially make sure to read how to set up your environment.

- Where do you want to start?
  - Start a new plugin
  - Contributing to existing plugin
- Plugin tutorial
- Architecture
- Extension points
  - Writing an SCM plugin
  - Action and its family of subtypes
  - Defining a new extension point
- Plugin cookbook (common starting points and recipes)
- Dependencies among plugins
- Hosting plugins at jenkins-ci.org
- FindBugs in plugins
- Checking out existing plugins
- Writing code
  - Using IntelliJ IDEA plugin
  - Structured Form Submission
  - Making your plugin behave in distributed Jenkins
  - Making your plugin behave in secured Jenkins
  - Marking a new plugin version as incompatible with older versions
  - Hints for plugin-development newbies
  - Exposing data to the remote API
  - Writing CLI commands
  - Participating to the initialization
  - Adding tool auto-installer
  - Hyperlinks in HTML
  - Dependency Injection
- Handling HTTP requests
  - Figuring out URL binding of Stapler
  - Web Method
- Writing Views (Jelly/Groovy)
  - Basic guide to Jelly usage in Jenkins
  - Understanding Jelly Tags
  - Writing Jelly views with IDE assistance
  - Adding tooltips
  - Writing a foldable section controlled by a checkbox
  - Jelly form controls
  - Jelly and XSS prevention
  - Unique IDs for Repeatable Content
  - AJAX with JavaScript proxy
  - Groovy View instead of Jelly
  - Hierarchical projects support
  - Form Validation
- XStream Persistence
  - Retaining backward compatibility
  - XStream Tips
- Writing unit tests
- Site Creation and Deployment
- Tips
  - Debugging native Maven jobs
  - Internationalization
  - Choosing Jenkins version to build against
  - Developing with JRebel
- OEM
• Bundling plugins with Jenkins
• Configuring Jenkins upon start up
• Making your plugin work with blue ocean

Additional references
• Plugin Structure
• Jenkins Source code cross-reference
• Jenkins Javadoc
• Documentation of Maven Jenkins Plugin
• Stapler URL mapping rules
• Jelly taglib reference core define, stapler, and taglibs defined in Jenkins core
• GitHub commit messages
• Moving from Subversion (svn) to Github
• UI Samples Plugin

Developing Plugins in Ruby
• Jenkins plugin development in Ruby — help wanted!
• Embedding Rack Application in your model
• Check out the source code for existing Ruby plugins

Developing Plugins in Python
• Jenkins plugin development in Python

Developing Jenkins
If you are interested in hacking Jenkins itself, read on to learn how to do so.
• About Jenkins project
• Building Jenkins
• Internationalization
• Jenkins core uses icons from the Tango Desktop Project.
• Jenkins CI for Jenkins Core
• Jenkins CI for Plugins
• Copyright on Source Code
• Contributing to Jenkins
• Pending Pull Requests
• Developing with JRebel
• Backporting toolkit for LTS

Jenkins Development Community Help
• Changing your Jenkins Contributor password on the Wiki, Jira and Fisheye

Developing Integrations with Jenkins
If you are developing other tools that interface with Jenkins, here are the relevant information.
• Auto-discovering Jenkins on the network
• Remote access API
• Execute arbitrary Java code on Jenkins master from remote systems
• Instance Identity

Need help?
• Where to find us