

Adding a container

Before you start you might be interested in reading the [Project Structure](#) tutorial which shows the directory organization of the Cargo sources. The [Building](#) tutorial explains how to build Cargo from sources and the [Contributing](#) tutorial explains what rules to follow when contributing code.

Here are some quick steps to follow if you wish to add support for a new container in Cargo:

- Subscribe to the cargo dev mailing list and ask as many question you'd like there! 😊
- Create a JIRA issue on <https://jira.codehaus.org> (you'll need to register). We'll then add you to the cargo-developers group in JIRA and assign the issue to you
- Checkout Cargo from [SVN](#) trunk
- Understand the Cargo project's [directory structure](#). Container implementations are located in `trunk/core/containers/ContainerName`.
- Have a look at existing container implementations
 - Some containers are simple to read and understand; for example **jo** or **glassfish**
 - Some other containers are much more feature-complete (remote deployers, datasources, etc.); for example **tomcat** or **jonas**.
- Create your container's Maven module, with its package inside.
- Create the following classes:
 - A [container](#) implementation class named `_ServerNameNxContainerType_Container` where `ServerName` is the name of the container, `Nx` the version and `ContainerType` the type of container (InstalledLocal or Remote). For example: `JBoss3xLocalContainer`.
 - One or several [configuration](#) implementation classes named `_ServerNameConfigurationType_Configuration` where `ConfigurationType` can be `StandaloneLocal`, `ExistingLocal` or `Runtime`. For example `JBossStandaloneLocalConfiguration`.
 - One or several [deployer](#) implementation classes named `_ServerNameDeployerType_Deployer` where `DeployerType` can be `InstalledLocal` or `Remote`. For example: `JBossInstalledLocalDeployer`.
 - Cargo has an SPI that you should use and that should make it easy for you. Your container class should extend `org.codehaus.cargo.container.spi.Abstract_ContainerType_Container` and your configuration class should extend `org.codehaus.cargo.container.spi.configuration.Abstract_ConfigurationType_Configuration`.
 - Finally, implement the `FactoryRegistry` that will register your container to CARGO and make sure you've defined a link to your container's factory registry in `src/main/resources/META-INF/services/org.codehaus.cargo.generic.AbstractFactoryRegistry`.
- Run the Cargo [build](#) to ensure everything is working. You'll probably find that you haven't followed the Cargo project's coding conventions... Fix those and build again until it passes! 😊 Please note that when you run the build it'll automatically run the samples test suites in your container (provided you've added your container to the generic API as described in the previous step and provided you've defined the right capabilities for your container). See the [Building](#) page for more details on the build.
- Once built, add your new container to the `uberpom` and check that the `uberjar` is still looking fine.
- Once added to the `uberpom`, add your container's download URL to the `samples` and add profiles for it.
 - As soon as you add your container to the `samples`, the Maven build will automatically attempt to configure, start, deploy some test applications, test them and stop your container.
- Register on [Codehaus' confluence](#). Once this is done we'll add you to the `cargo-developers` user group so that you have the right to edit yourself the Cargo web site pages
- Document the new container on the [Cargo web site](#)
- Create a SVN patch and attach it to the JIRA issue you have created above
- Once the patch has been accepted:
 - Add the container to the [Containers list](#) on the main web site
 - Add the container to the [Navigation page](#) (left side of the CARGO Web site)

- Add the container to the [Downloads page](#)
- Add the container to the [Continuous integration](#) so that its integrity can be checked at each build
 - Note that each container is built daily, each of them separated by 45 minutes. Just put your container as last, so it is tested daily without blocking the whole CI environment.

Thanks and happy coding!