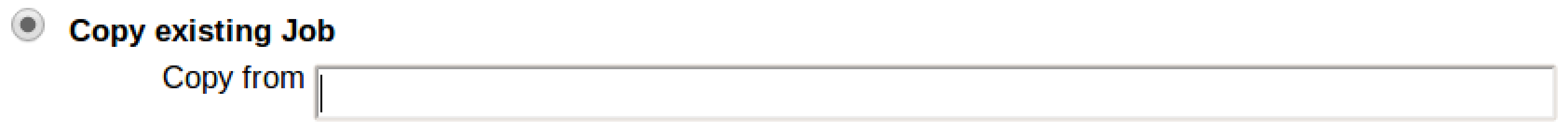
Jenkins

Advanced Builds

Section Contents

- Copying Projects
- Parameterized Builds
- Parameterized Triggers
- Multi-configuration Jobs
- Parallel Builds and Joining
- Build Pipelines

Copy Project



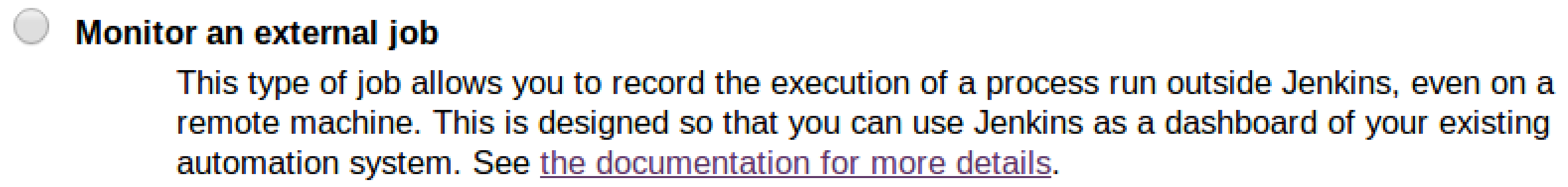
[● Copy existing Job]

Copy from

Copy Project

- Simple way to duplicate a preexisting job
- Will not fill the entire project, but will give you a head start

Monitoring External Jobs


○ **Monitor an external job**

This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system. See the documentation for more details.

Monitoring External Jobs

- Adds the ability to monitor the result of externally executed jobs.
  - Cron jobs, procmail, inetd-launched processes
  - Using Jenkins enables you to monitor a large number of such tasks with little overhead
- Handy to monitor servers and any external operations

Monitoring External Jobs (Part 1)

- Either
  - Retreive jenkins-core.jar from either the install
  - Run `sudo apt-get install jenkins-external-tool-monitor` (Linux, MacOSX)
- Either
  - `export JENKINS_HOME` (Linux, MacOSX)
  - `set JENKINS_HOME` (Windows)
- Create an external job in Jenkins

Monitoring External Jobs (Part 2)

- Set Jenkins Home to the URL:
  - `http://user:password@xyzcorp.com/path/to/jenkins/`
- Run
  - `java -jar /path/to/WEB-INF/lib/jenkins-core-*.jar ”job name” <program arg1 arg2...>`

Lab: Monitor External Job

- Create a monitor external job called "`windows dir`"
- Run a command prompt job that should fail, by calling `dir` on a directory or drive that doesn't exist
- Run a command prompt job that should succeed, by calling `dir` on a directory or drive that does exist, or just by calling `dir`
- Review the results in Jenkins

Parameterized Builds

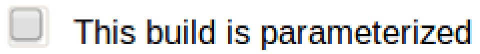
Parameterized Builds

- The Parameterized Build let you configure parameters based on:
  - String
  - Boolean
  - Choice
  - File
  - Any build (run)

Why Parameterized Builds

- Specify a target environment
- Specify quality of test
- Specify version of application or dependency to use
- Specify a browser for web tests

Parameterized Builds



☐ This build is parameterized

Parameterized Builds

- Boolean – True or False
- Choice – Multiple Choice you enter your own values
- File Parameter – Upload a file as a parameter
- Subversion Tag – Subversion release tag
- Password – A masked password field
- Run Parameter – A stable build URL of another project
- String Parameter – Non multiline string
- Text Parameter – Multiline string

Processing the Parameter

- The parameter given can be used as an environment variable when the job is run
- Standard is to make them upper case, e.g. `PACKAGE_NAME`
- Refer to the environment variable you can use either `${name}` or `$name` for Linux of Mac
- Refer to the environment variable you can use either `%name%` for Linux of Mac.

Processing the Parameter in Ant/NAnt

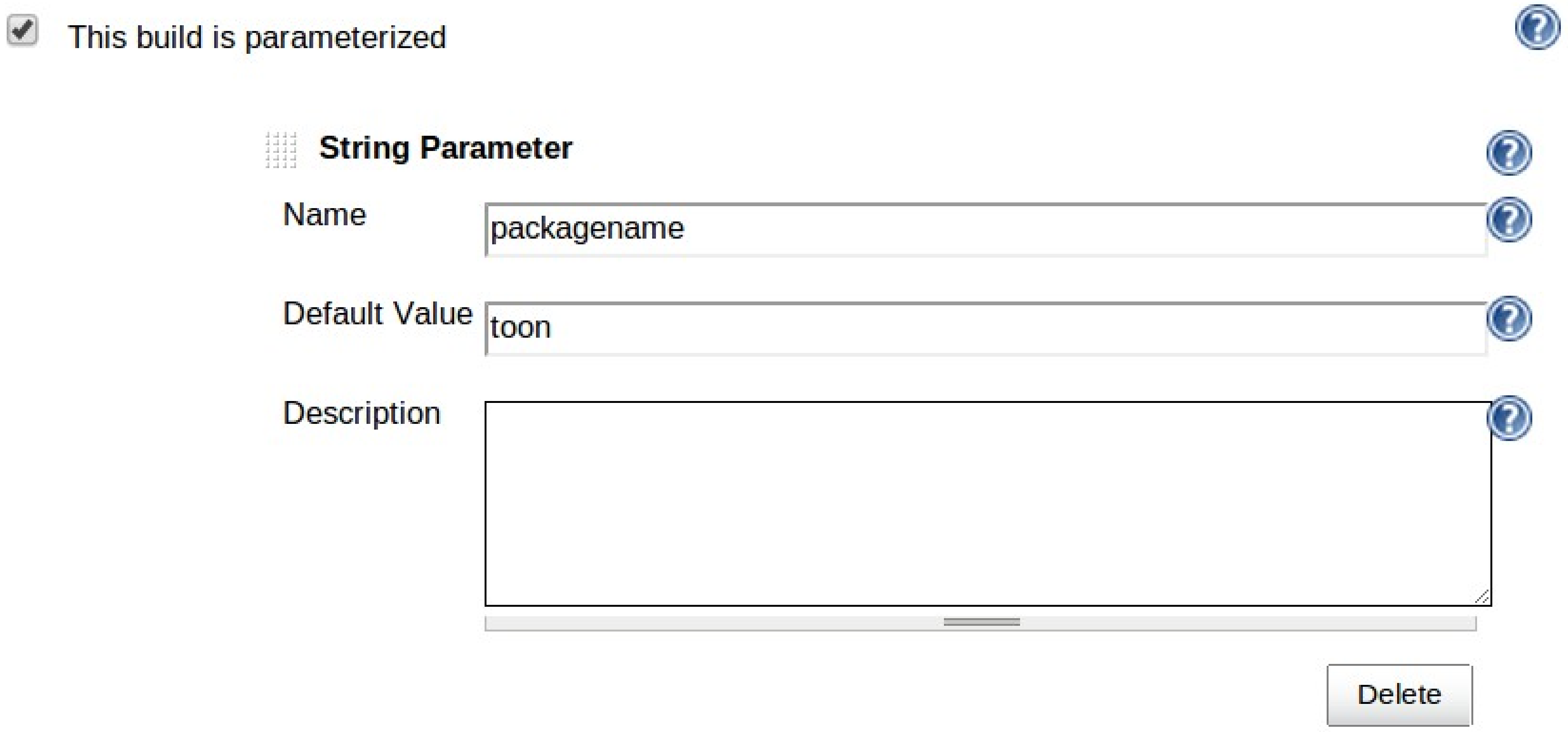
- Ant -

```
<property environment="env"/>
<property name="package_name" value="${env.PACKAGE_NAME}"
```

- Nant-

```
<property name="package_name"
value="${environment::get-variable('PACKAGE_NAME')}" />
```

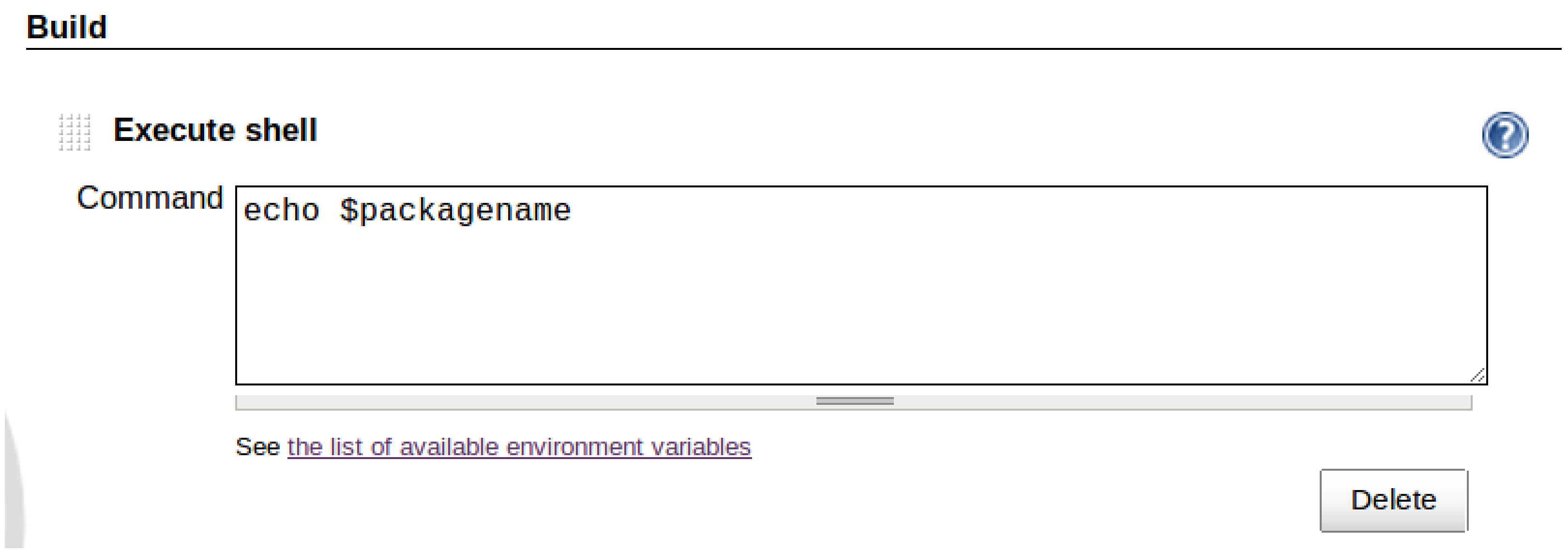
Process the Parameter


☑ This build is parameterized

String Parameter

Name	packagename
Default Value	toon
Description	

Process the Parameter



Build

Execute shell

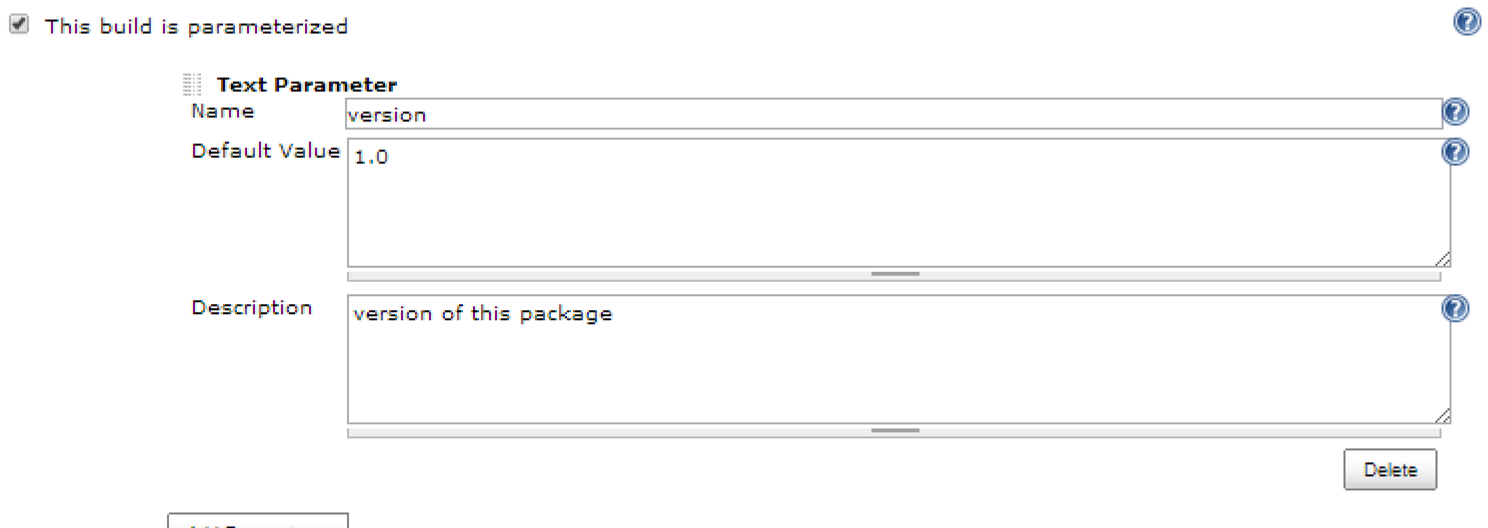
Command

```
echo $packagename
```

See the list of available environment variables

Lab: Create a parameterized job

- Create a free-style software project, called `parameterized-build`
- Select "This build is parameterized", and add a text parameter
- Include a parameter with name of `version`, default value of `1.0`, and a description of what version is.



[✓] This build is parameterized

Text Parameter

Name	version
Default Value	1.0
Description	version of this package

Lab: Create different parameters

- On the same `parameterized-build` project add some other parameters.
- Add a choice parameter of "mssqlserver", "postgres", "mysql", "mongodb"
- Add a password parameter of "password"
- A boolean parameter of "release-candidate"
- And a run parameter selecting a job's successful builds
- Output each of the parameters on either a Window batch (Windows only) or Shell (Linux, Mac)

Creating a token to run the job

- Tokens are used to ensure that the user processing the job is legitimate
- Set up a token in "Build Triggers" of your job
- Select a one word token to process the job

Starting a parameterized job from URL

- To start a job remotely via URL append `/buildwithparameters` to the end of the URL
- Add a query string `?parameter=value` where the parameter is the parameter name and the value is the value
- Might be more complicated with the file upload or run parameters
- Ne sure to add a token at the end of the URL like `?parameter=value&token=pennies`

Lab: Call a job via a URL with a token

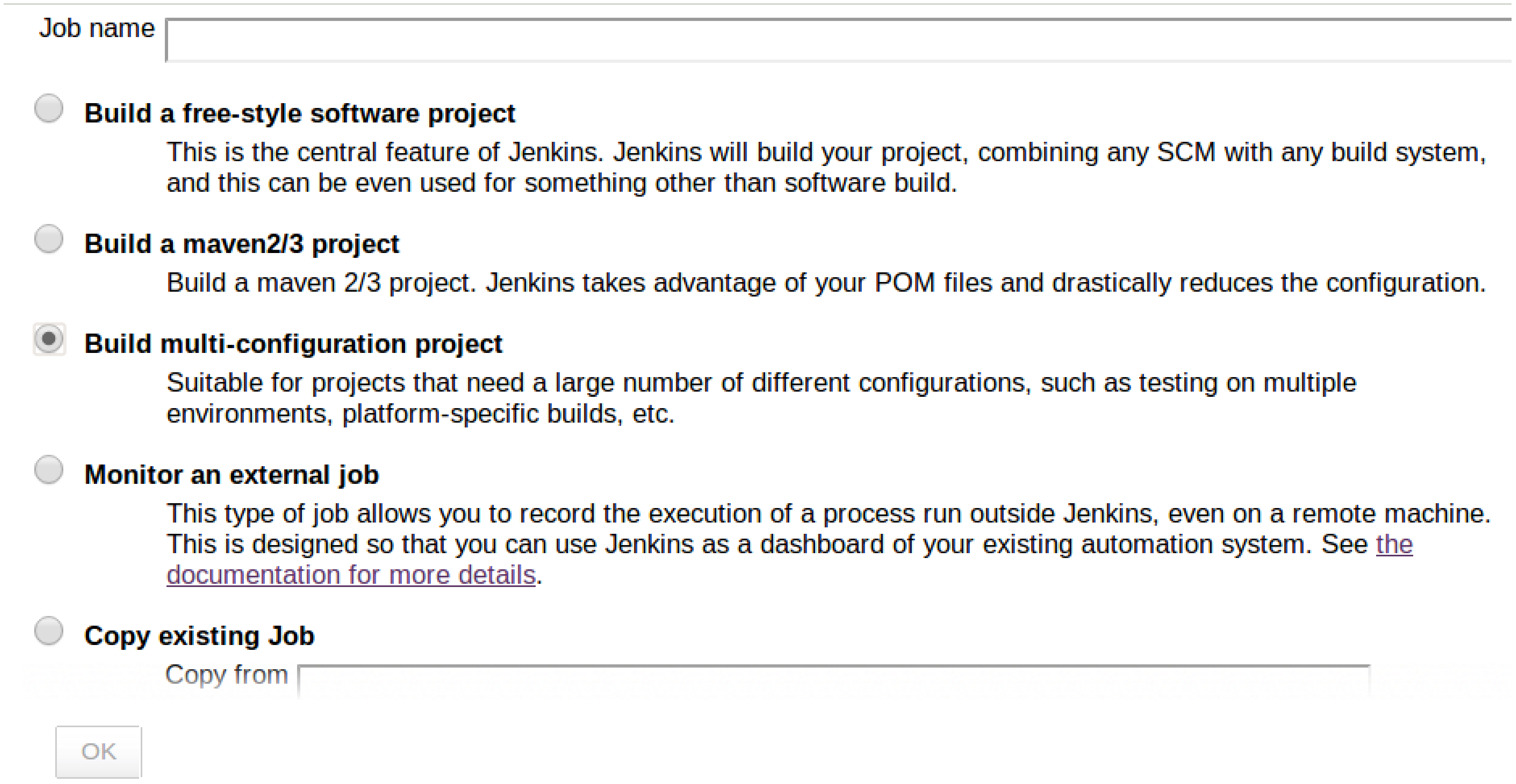
- Establish a token on the `parameterized-build` job
- Call the parameterized job via the URL on the web browser using the token
- Note: You may not be able to run remotely with a run parameter, even by URL encoding.

Multiconfiguration Projects

Multiconfiguration Project

- Multiconfiguration Jobs are jobs that can run with all possible jobs
- Useful for jobs, tests, browsers, databases, etc.
- Created using a configuration matrix
- The axis can be created based on remote nodes, JDKs, or custom accesses

Creating a Multiconfiguration Project



Job name

( ) **Build a free-style software project**

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

( ) **Build a maven2/3 project**

Build a maven 2/3 project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

[✓ **Build multi-configuration project**]

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

( ) **Monitor an external job**

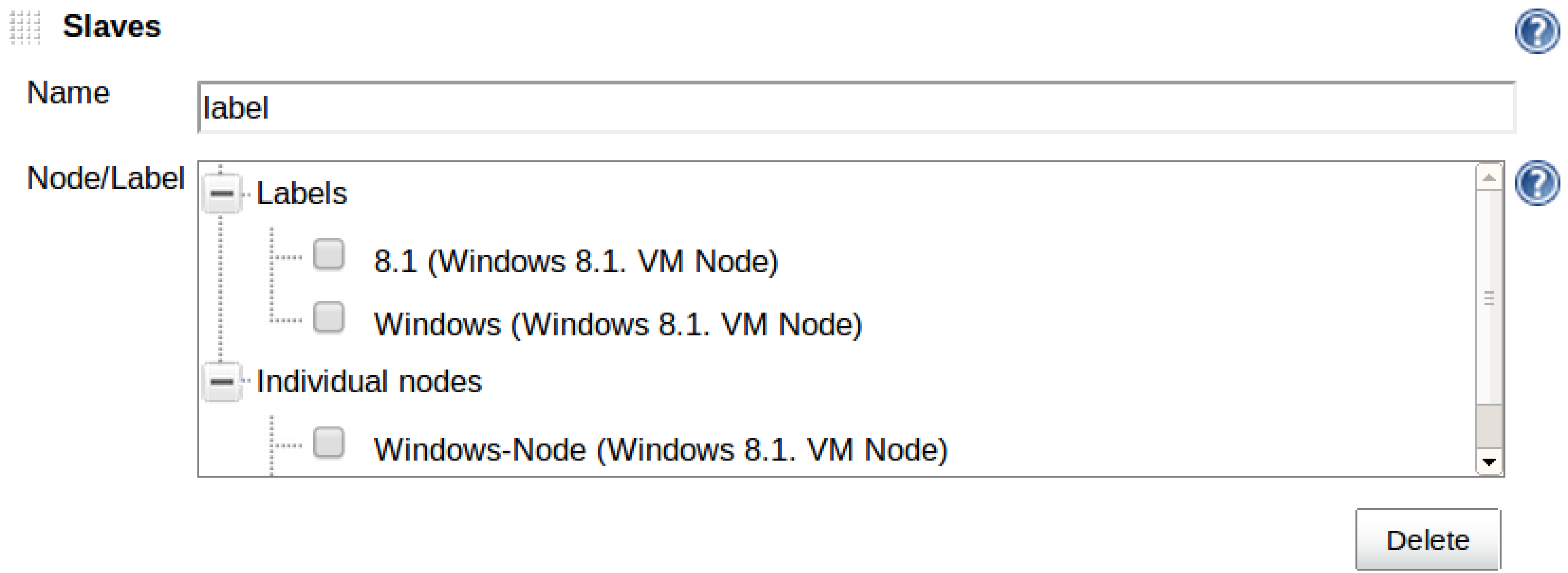
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system. See the documentation for more details.

( ) **Copy existing Job**

Copy from

Remote Node Axis

- Remote (Slave) Node can be chosen as a group in case you want to differentiate builds based on operating system
- Can be done based on Labels (more flexible)
- Can also be done based on explicit Individual Nodes



Slaves

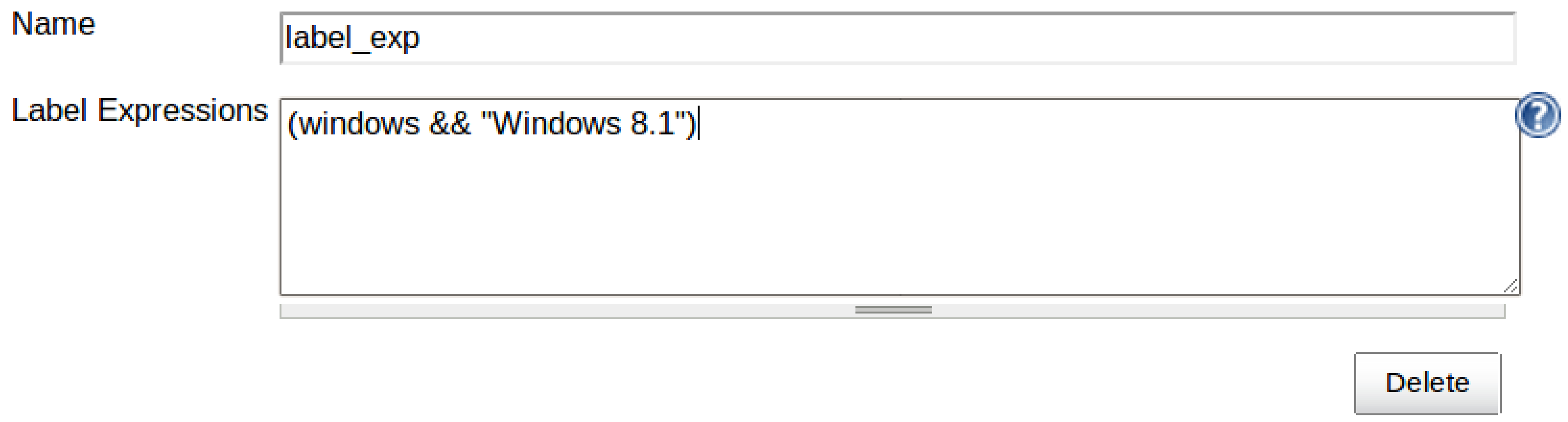
Name: label

Node/Label:

- Labels
  - [ ] 8.1 (Windows 8.1. VM Node)
  - [ ] Windows (Windows 8.1. VM Node)
- Individual nodes
  - [ ] Windows-Node (Windows 8.1. VM Node)

Label Axis

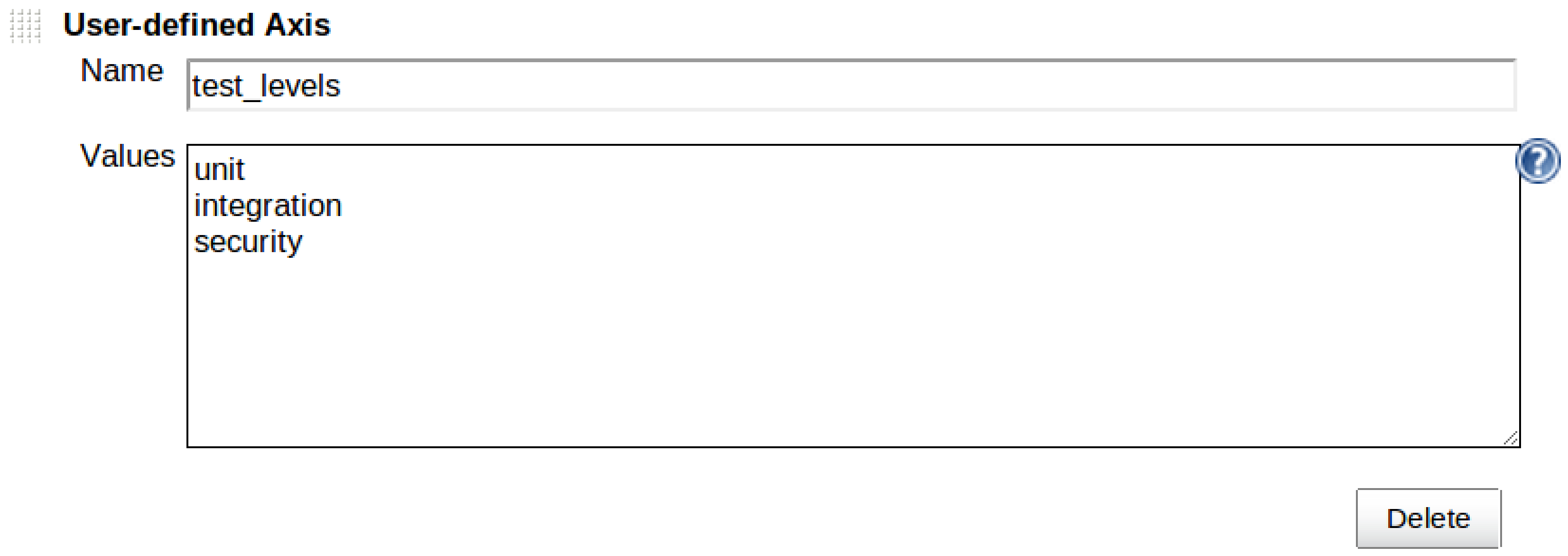
- Run the same build on multiple slaves (much like the "slaves" axis)
- This is more advanced in that it lets you use boolean expressions (such as `foo&&bar`)



Name	label_exp
Label Expressions	(windows && "Windows 8.1")

Custom Axis

- If neither of the axises meet your requirements
- You can create a custom list of items that can be used for the build



User-defined Axis

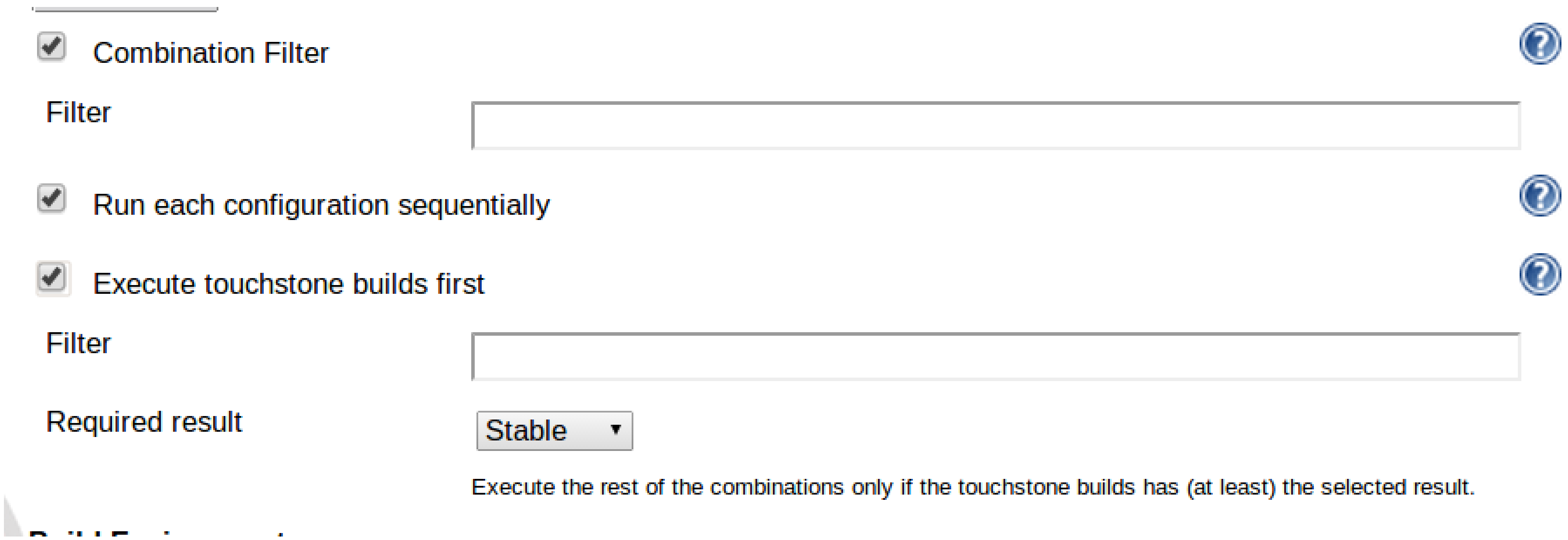
Name: test_levels

Values:
```
unit
integration
security
```

Specialized Configuration

- Combination Filter –
  - By default Jenkins will run every combination, but when not needed you can enter a boolean expression.
  - Applicable builds will be blue/yellow/red.
  - Non-Applicable builds will be gray
- “Execute touchstone builds first” -
  - Boolean statement that selects the "you go first" build
- "Run each configuration sequentially" – Good choice if you want to avoid resource contention

Specialized Configuration



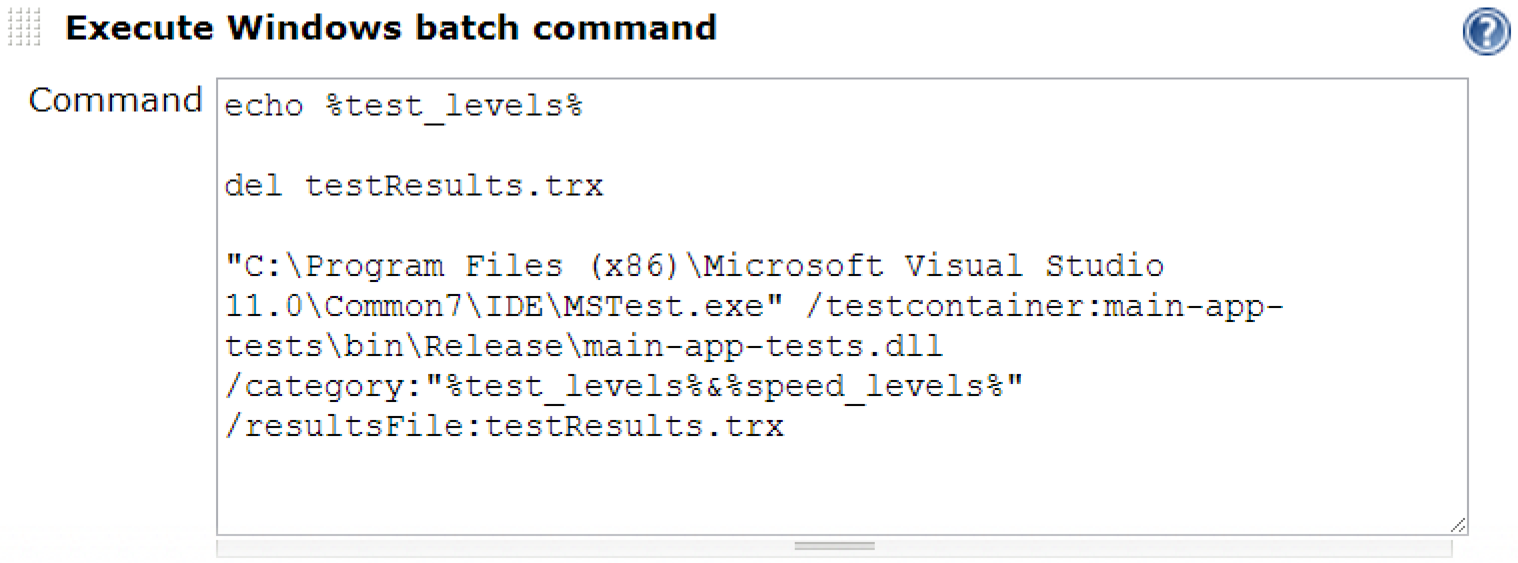
✓ Combination Filter	
Filter	
✓ Run each configuration sequentially	
✓ Execute touchstone builds first	
Filter	
Required result	Stable

Execute the rest of the combinations only if the touchstone builds has (at least) the selected result.

Lab: Create a Multiconfiguration Build

- Create a multiconfiguration job called multi-msbuild-project
- Use the same VCS setting from the `simple-msbuild-project`
- Create two user defined axises, `test_levels` and `speed levels`
- Use the same MSBuild settings, but do not use the MSTestRunner, instead use Windows Batch
- Add testResults.trx for Publish MSTest Results

Lab: Create a Multiconfiguration Build



```
echo %test_levels%

del testResults.trx

"C:\Program Files (x86)\Microsoft Visual Studio
11.0\Common7\IDE\MSTest.exe" /testcontainer:main-app-
tests\bin\Release\main-app-tests.dll
/category:"%test_levels%&%speed_levels%"
/resultsFile:testResults.trx
```

Lab: Combination Filters & Touchstone Builds

- Create a combination filter with that will select fast and slow builds but only for unit tests
- Run the build, observe
- Create a touchstone build that will only select the fast and unit build
- Run the build, observe

Fingerprinting

- Fingerprinting tracks build artifacts
- Backed by an MD5 checksum which maintained by Jenkins
- In each job go to and create a post-build action: "Record fingerprints to track usage"
- MD5s are stored in `$JENKINS_HOME/fingerprints`

Lab: Fingerprinting

Parallel Builds

Parallel Builds

- Used in conjunction with remote nodes
- Works especially well multi-configuration jobs
- To set off parallel builds merely add the jobs that will run in remotes in "Build other projects` in the Post Build Actions of the the Job.

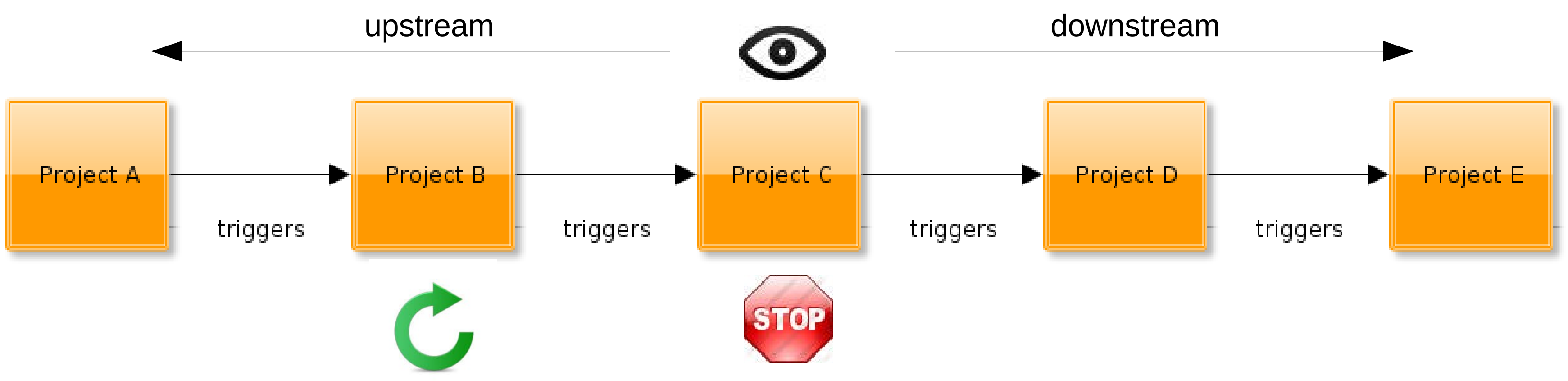
Lab: Create Parallel Jobs

- Create Integration Test Job that would be used on other people's machines
- Run one instance of an integration test on a Remote Node of another machine
-

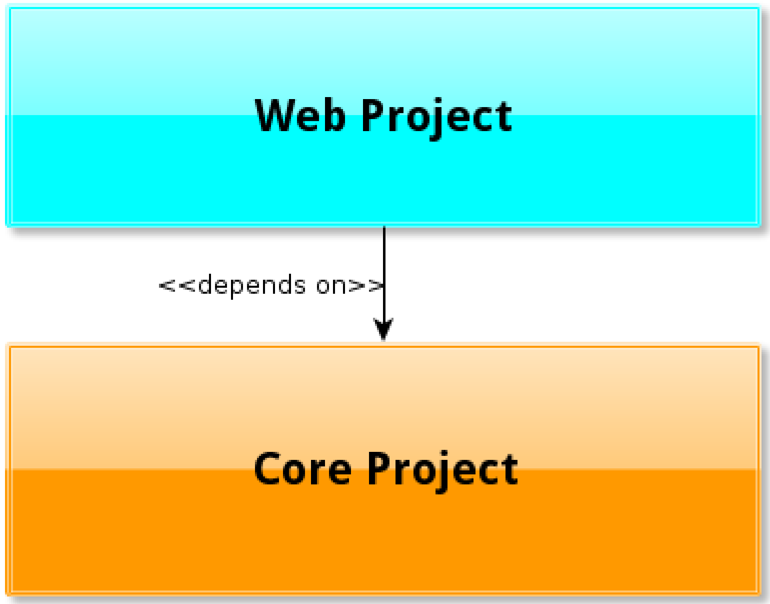
Blocking Builds Upstream

- Jenkins will prevent the project from building when a dependency of this project is in the queue, or building.
- The dependencies include the direct as well as the transitive dependencies.

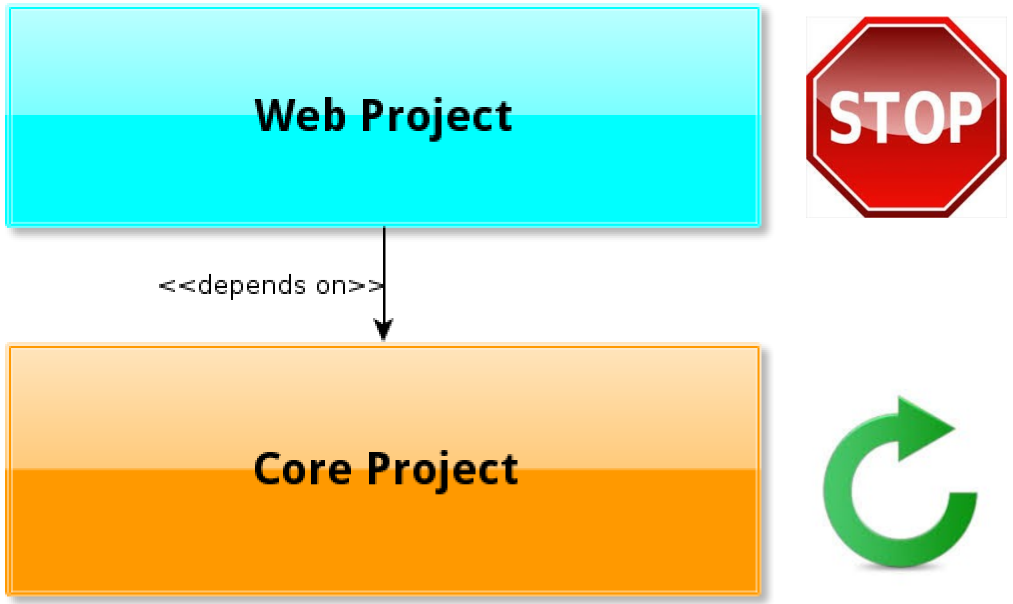
Upstream? Downstream?



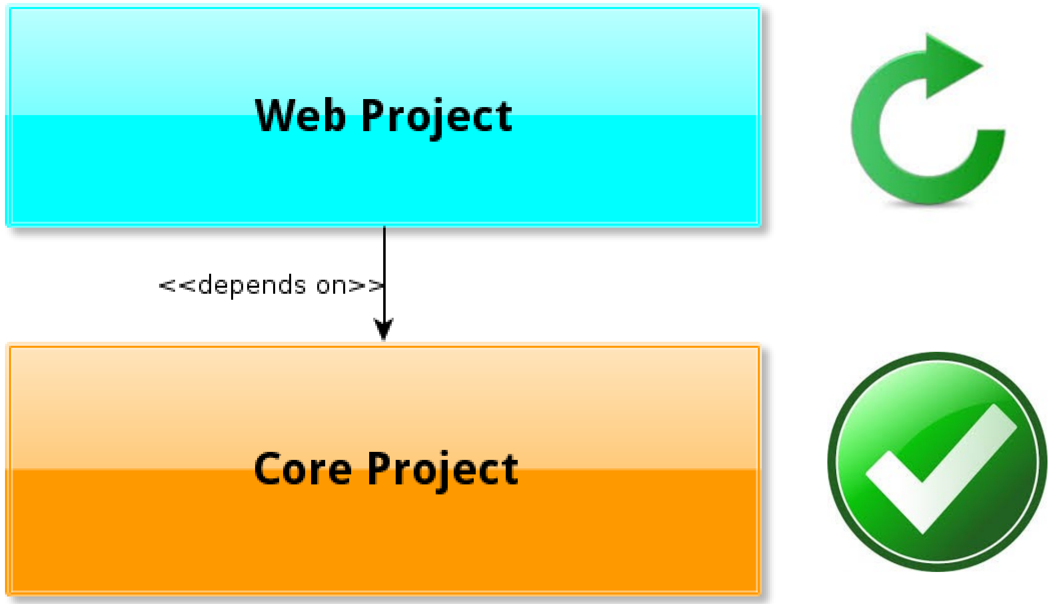
Blocking Builds Upstream



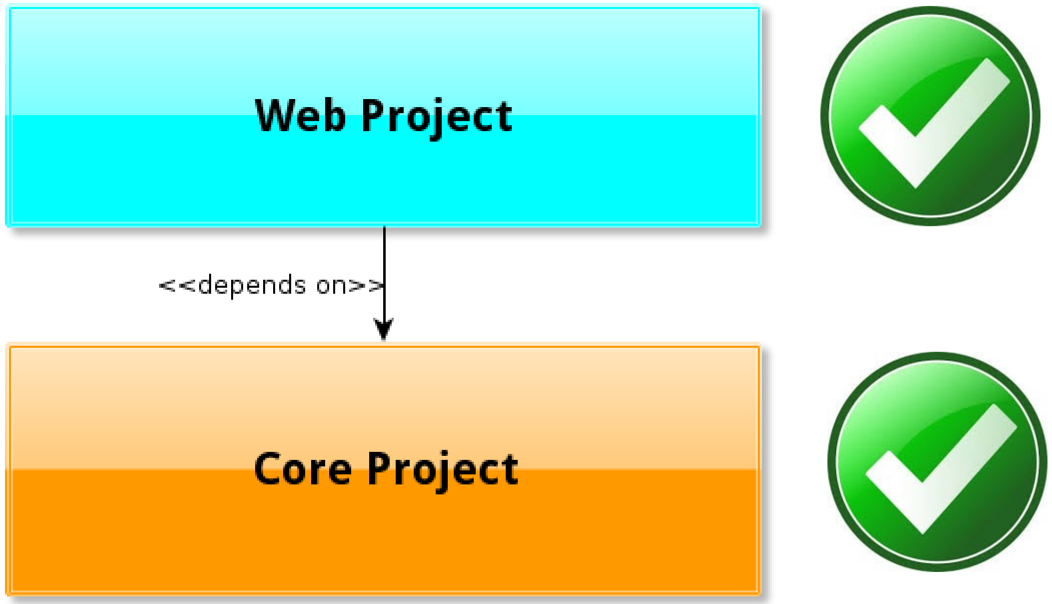
Blocking Builds Upstream



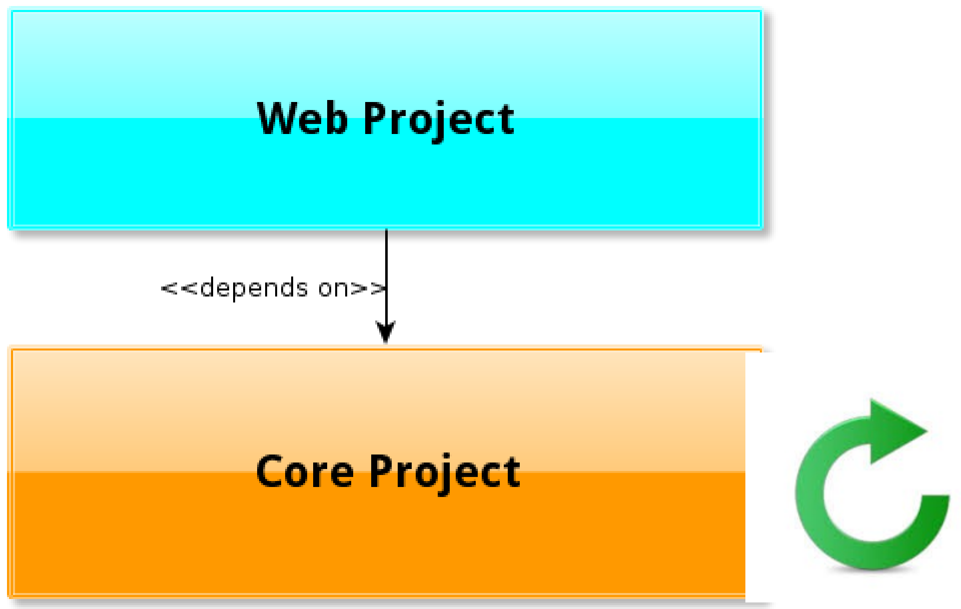
Blocking Builds Upstream



Blocking Builds Upstream



Blocking Builds Downstream


Web Project
<<depends on>>
Core Project