Logging

- Logs on the system
    - Linux
    - Windows
    - Mac OS X
    - Docker
- Logs in Jenkins
- Making logs available outside of the web UI

Logs on the system

When running `jenkins.war` manually with `java -jar jenkins.war`, all logging information by default is output to standard out. Many Jenkins native packages modify this behavior to ensure logging information is output in a more conventional location for the platform.

Linux

By default logs should be made available in `/var/log/jenkins/jenkins.log`, unless customized in `/etc/default/jenkins` (for *.deb) or via `/etc/sysconfig/jenkins` (for */rpm)

Windows

By default logs should be at `%JENKINS_HOME%/jenkins.out` and `%JENKINS_HOME%/jenkins.err`, unless customized in `%JENKINS_HOME%/jenkins.xml`

Mac OS X

Log files should be at `/var/log/jenkins/jenkins.log`, unless customized in `org.jenkins-ci.plist`

Docker

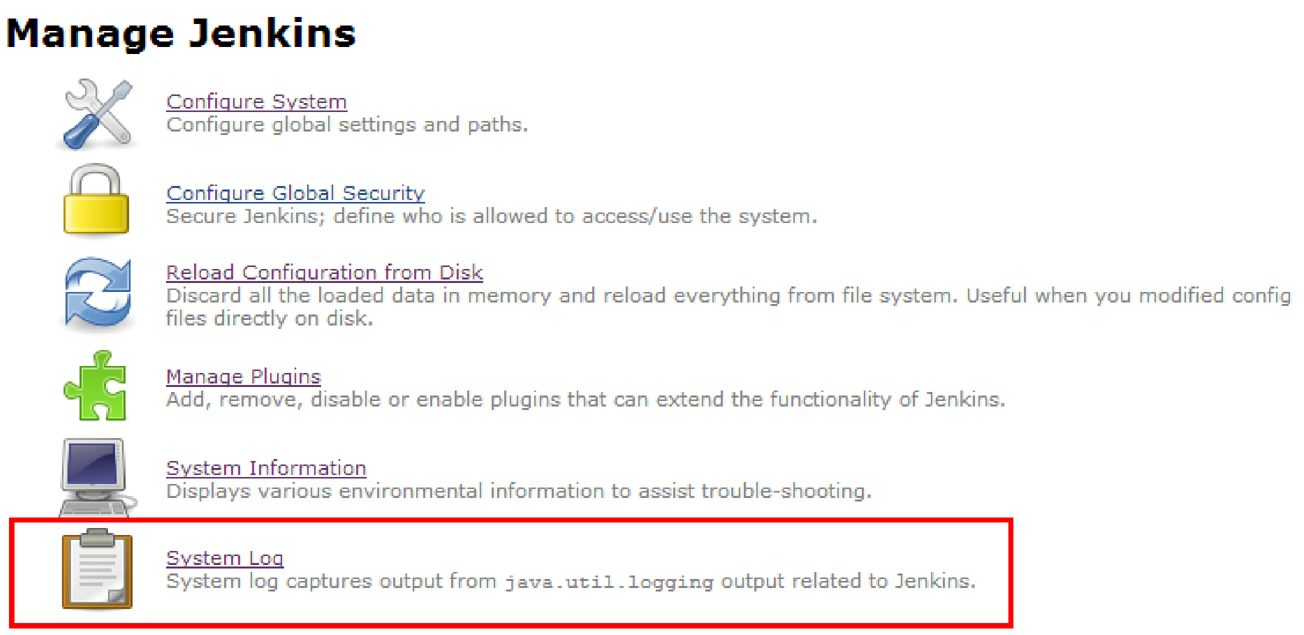
If you run Jenkins inside docker as a detached container, you can use `docker logs containerId` to collect jenkins log

Logs in Jenkins

Jenkins uses `java.util.logging` for logging. The `j.u.l` system by default sends every log above `INFO` to stdout, but unfortunately, most servlet containers alter this behavior, making it difficult for us to tell you exactly where you should look at. Also, they tend to bundle all the logs from the entire JVM into a single place, making it difficult to follow a particular aspect of the system.

Because of these reasons, Jenkins is equipped with a GUI for configuring/collecting/reporting log records of your choosing. This page shows you how to do this.

First, select the "system log" from the "Manage Jenkins" page:


Manage Jenkins

Configure System
Configure global settings and paths.

Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.

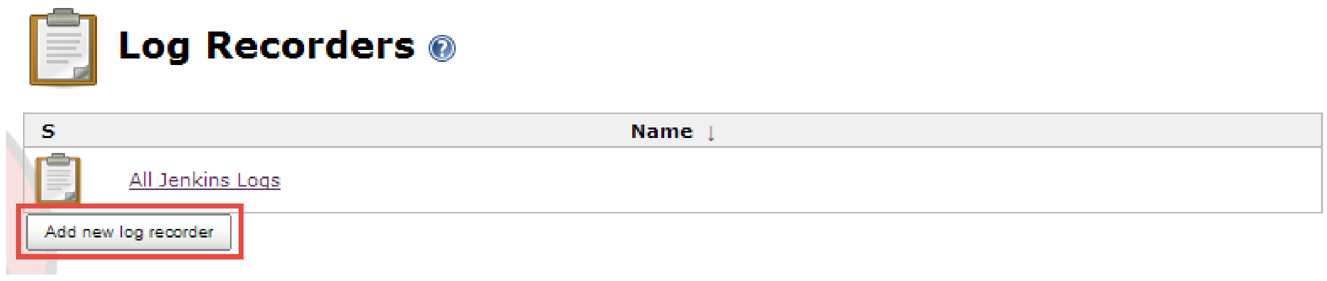
Reload Configuration from Disk
Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.

Manage Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

System Information
Displays various environmental information to assist trouble-shooting.

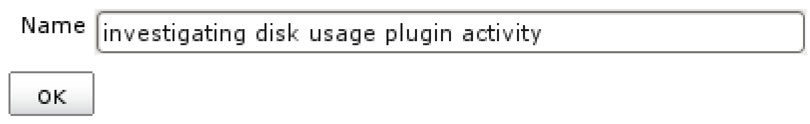
System Log
System log captures output from `java.util.logging` output related to Jenkins.

From there, you can create a custom log recorder, which helps you group relevant logs together while filtering out the noise.

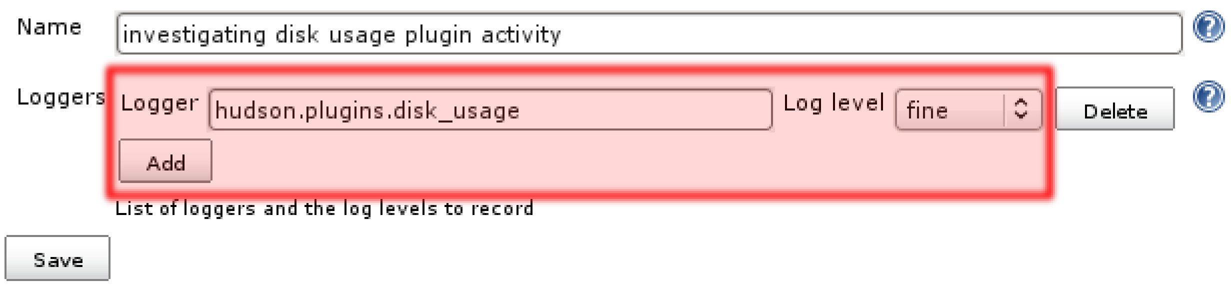

Log Recorders

S	Name ↓
	All Jenkins Logs

Choose a name that makes sense to you.


Name: investigating disk usage plugin activity

You'll be then asked to configure loggers and their levels whose output you'd like to collect. Depending on which part of Jenkins you monitor, you'll need to specify different loggers. Tell us the symptom of your problem in the users list and we should be able to tell you where you need to look at. Also, this is really just a wrapper around the java.util.logging package, so if you program in Java, you might be able to guess what to look at.


Name: investigating disk usage plugin activity

Loggers: Logger: hudson.plugins.disk_usage Log level: fine

List of loggers and the log levels to record

Once the set up is complete, Jenkins will start collecting data. The collected logs are available from the web UI.

Making logs available outside of the web UI

Some people have a need to send log messages to files or a central log server. Another problem with making logs available only in the web UI can arise in case login to Jenkins is broken and therefore you cannot access the log view page which would show you the exceptions that would help you to find out why you cannot login successfully.

Unfortunately, there is no simple way in Jenkins to send log messages generated by Jenkins himself or by any plugins to the console, to a file or anywhere else. No matter what command line options you will use, you just won't succeed. It won't even help if you inject a java.util.logging configuration file directly into the JVM as Jenkins will override any settings in there programmatically.

Technically speaking, Jenkins instantiates a logger called hudson.WebAppMain in the the WebAppMain class and attaches one and only one handler to it: A hudson.util.RingBufferLogHandler. That handler just collects log records in memory and makes them accessible to the log view view in the web UI. All of Jenkins and its plugins inherit from the hudson.WebAppMain logger, which is why nothing being logged there (not even exceptions) ever make it to the console or any log files by default. And as you browse the source code, you'll learn there are absolutely no hooks available at all to allow log output to go anywhere else.

One solution is to use a little known mechanism available in Jenkins: Groovy Hook Script. One can write a piece of Groovy code which will be executed by Jenkins as an init script, i.e. every time Jenkins starts; right when it will be fully initialized and ready to start work. As that Groovy script will execute in the same JVM as Jenkins himself, one can access and manipulate all objects in that VM. So just four lines of Groovy code will be enough to add another handler to the hudson.WebAppMain logger to receive a copy of any log entries generated inside Jenkins and direct them where every you want them to show up.

For example, put the following Groovy script into a file called.jenkins/init.groovy.d/extralogging.groovy

```
import java.util.logging.ConsoleHandler
import java.util.logging.FileHandler
import java.util.logging.SimpleFormatter
import java.util.logging.LogManager
import jenkins.model.Jenkins

// Log into the console
def WebAppMainLogger = LogManager.getLogManager().getLogger("hudson.
WebAppMain")
WebAppMainLogger.addHandler (new ConsoleHandler())

// Log into a file
def RunLogger = LogManager.getLogManager().getLogger("hudson.model.Run")
def logsDir = new File(Jenkins.instance.rootDir, "logs")
if(!logsDir.exists()){logsDir.mkdirs()}
FileHandler handler = new FileHandler(logsDir.absolutePath+"/hudson.
model.Run-%g.log", 1024 * 1024, 10, true);
handler.setFormatter(new SimpleFormatter());
RunLogger.addHandler(handler)
```

This will just copy all log records generated by Jenkins and any plugins to the console and all logs from hudson.model.Run into a file. They will still be available in the log view in the web UI.

Of course, you can edit the Groovy script to add other or more handlers, set their levels, etc. Just refer to the java.util.logging documentation.

A much simpler solution (though with no configurability) is to install the Support Core Plugin, which causes custom logs to be written to disk automatically.

Debug logging in Jenkins

Create a file `logging.properties` in which you define the logging levels and a `ConsoleHandler` Then pass this file to the JVM by adding the system property `-Djava.util.logging.config.file=<pathTo>/logging.properties`. In the logging.properties file please add the below line:

logging.properties

```
.level=TRACE
```