# Hudson Web Architecture

ORACLE®

# Winston Prakash
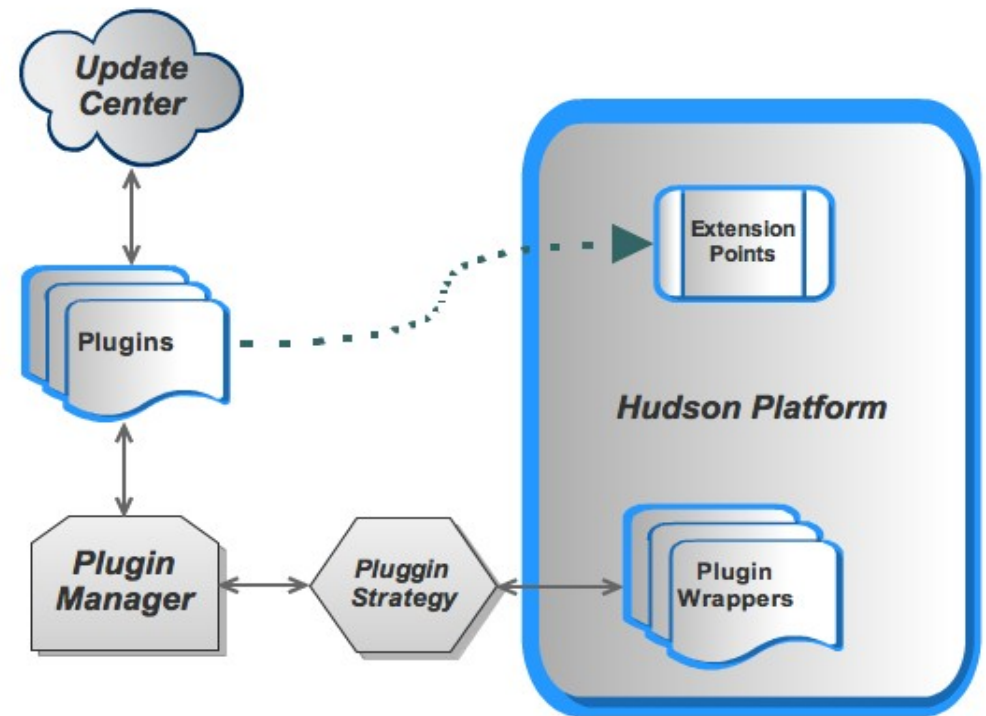
# Hudson Plugin Architecture

Hudson is an extendable Web Application. The basic platform is extended via plugins. Plugins are responsible for most of the functionalities in Hudson.

Pluggability is achieved through three basic components

- Plugin Manager
- Update Center and Update Site
- Plugin Wrapper & Plugin Strategy



ORACLE

# Plugin Manager

Plugin Manager is a service in Hudson core that is responsible for loading the plugins from Update Center as well the bundled plugin.

It also provides services such

- Installing a plugin
- Upload a plugin
- Configure Update Center
- Configure Proxy

It is also a container for list of

- Installed Plugins
- Failed Plugins
- Available Plugins

Provides UI for



**View Provided by Plugin Manager**

| Install | Name ↓ | Version | Installed |
|---------|--------|---------|-----------|
| ☐ | **Maven 2 Project Plugin** <br> Hudson's Maven 2 project type <br> Warning: This plugin is built for Hudson 1.373 or newer. It may or may not work in your Hudson. | 1.373 | 1.372 |
| ☐ | **SSH Slaves plugin** <br> This plugin allows you to manage slaves running on \*nix machines over SSH. | 0.13 | 0.12 |
| ☐ | **Subversion Plugin** <br> This plugin adds the Subversion support (via SVNKit) to Hudson. | 1.17 | 1.11 |
| ☐ | ui-samples-plugin <br> Warning: This plugin is built for Hudson 1.373 or newer. It may or may not work in your Hudson. | 1.373 | 1.372 |

This page lists updates to the plugins you currently use.    [Install]

- Proxy configuration for Update Center Connection
- Update Site Configuration
- Main Page for Hudson → Manage Hudson → Manage Plugins (index.jelly)
- Installed (installed.jelly)/Available (available.jelly/Advance(advanced.jelly) tab of the Update Center

**ORACLE**

# Plugin and Plugin Wrapper

A plugin is a jar file with extension .hpi whose properties are defined via a jar Manifest. The entry point is via a class that extends the base class called "Plugin".

A plugin is bound to URL space of Hudson as <Hudson-server-root>/plugin/plugin-name, where plugin-name is taken from the plugin name "plugin-name.hpi".

**Plugin Wrapper** is a service object that wraps a **plugin** and provides services such

- Enabling a plugin
- Disabling a plugin
- Pin a plugin

A Plugin can have an optional config.jelly page. If present, it will become a part of the system configuration page.

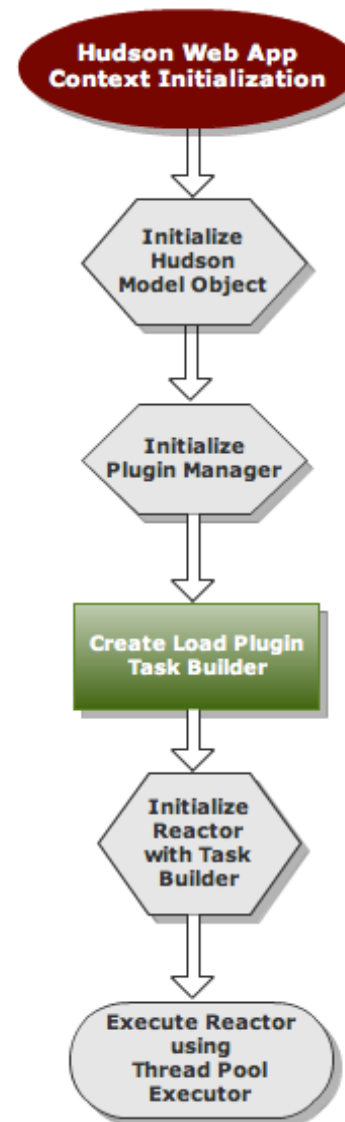# Plugin Manager Initialization

Plugin Manager is initialized suing the Servlet Context Initialization Phase.

During this phase **Hudson Model Object** (singleton) is initialized. Several services are created at this point and one such services is Plugin Manager.

Various tasks to load the plugins are created as a graph of tasks using **Task Builder**

A **Reactor** is initialized with Task Builder along with various other initialization Task Builders in the Hudson Model Object.

Finally the Reactor is executed using **Thread Pool Executor** (Java Concurrent Executor).

**Hudson Web App Context Initialization**

↓

Initialize Hudson Model Object

↓

Initialize Plugin Manager

↓

**Create Load Plugin Task Builder**

↓

Initialize Reactor with Task Builder

↓

Execute Reactor using Thread Pool Executor

# Building Task Graph

During initialization process various tasks needs to be executed. These tasks may depend on each other. For this purpose Hudson introduces a concept of **Task Graph Builder**.

A Task Graph Builder is a **Task Builder** which in turn a container for collection of Tasks.

When a Task Builder finishes it attains a **Milestone**. The Tree Node of TaskGraphBuilder is an **Executable** Task Node. Each Executable Node may require a Milestone attained by another task to continue.

A Graph Builder can loop through a list of task. For the loop to continue, the Task must complete with out non fatal error, or the next task must be marked as "**not fatal**" for any out come of the previous task.

When a Task Builder is constructed it is passed on to a **Reactor** to be executed using Java concurrency **Executor Service**.

A Reactor reacts to a GraphTaskBuilder outcome and notifies its listener.
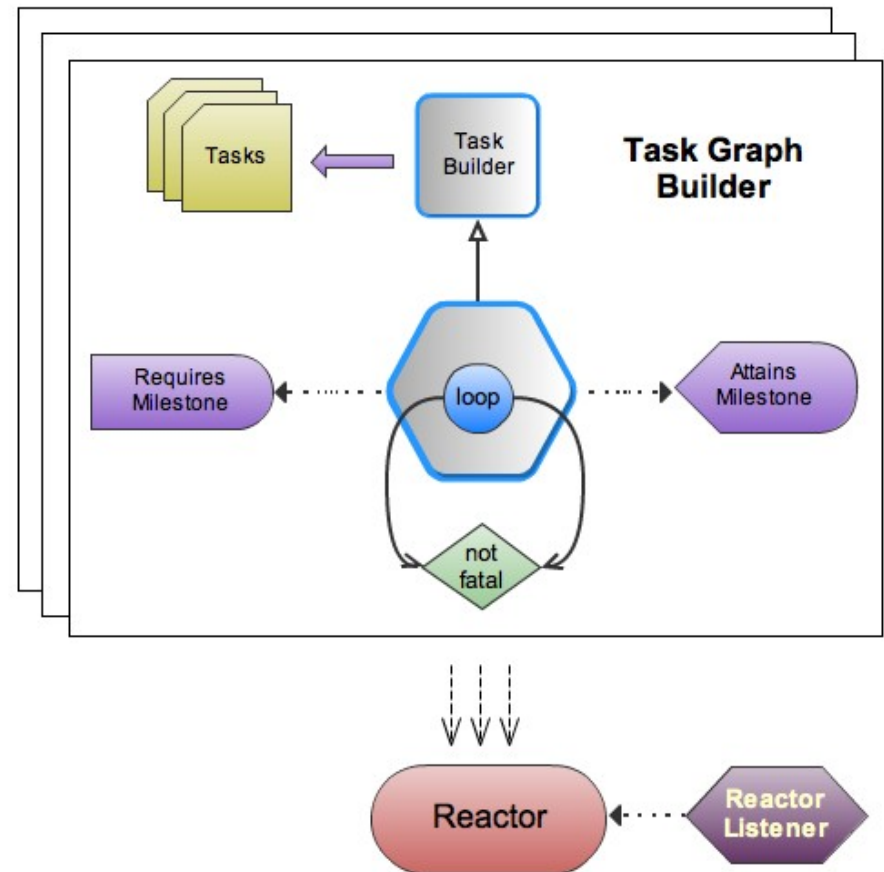
# Load Plugin Task Graph Builder

Loading starts with the bundled plugins first. They are copied from WEB-INF/plugins to the Plugin Root (usually .hudson/plugins)

The list of plugins are obtained using various strategies defined by InitStrategy (Strategy Pattern) such as
- List all (*.hpi or *.hpl) from Plugin Root
- Plugin name specified via property "hudson.bundled.plugins"

Create PlugginWrapper for each plugin using PluginStrategy (another Strategy pattern Object) and list active plugins for loading.
- Load Manifest
  - Find Libraries and Classes
  - Find dependent plugins
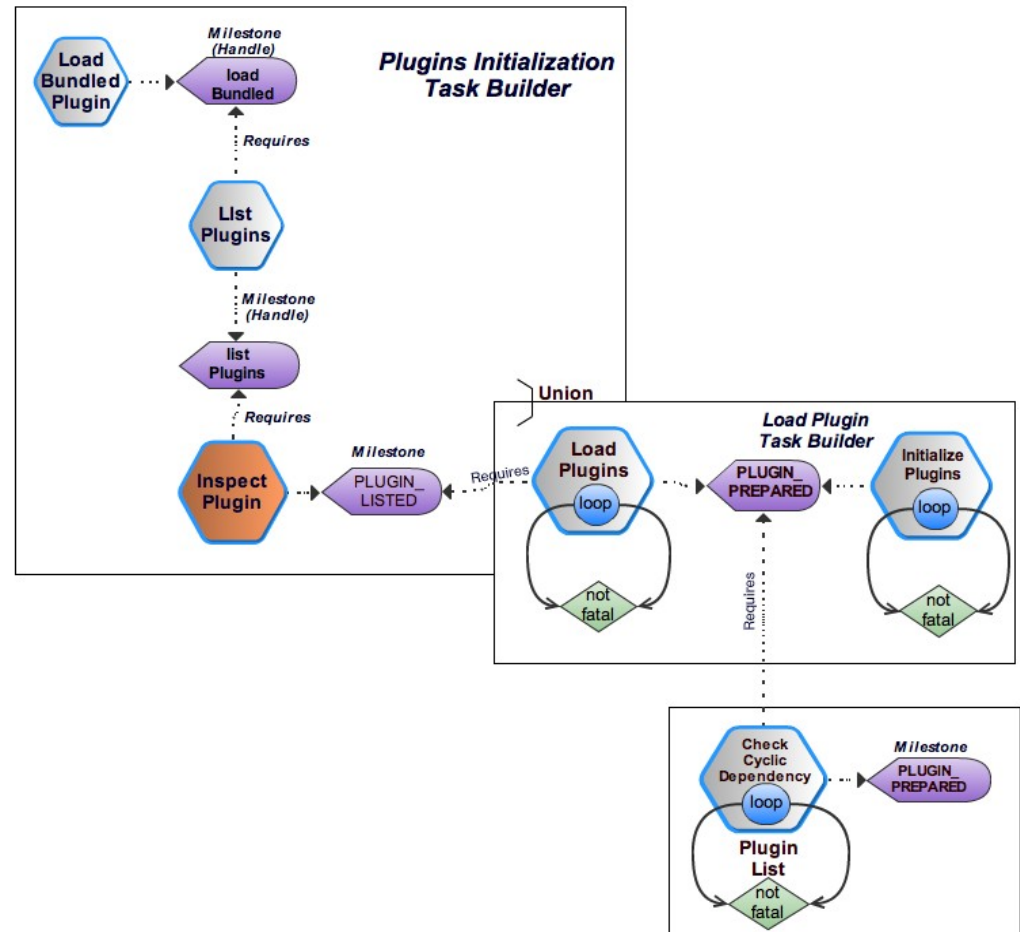- Creating class-loader
- Creating dependency class-loader

Check for any duplicate.

Load the active plugins with PluginStrategy. This involves loading the Plugin Class and call start()

Resolve missing dependencies

Check for Cyclic dependency.

Call Plugin.postInitailize()

# Plugin Class Loader

For security purpose Hudson loads class of each plugins in its own class-loader.

The base class-loader is a URL class-loader that loads classes defined in jars and classes from
- Libraries defined in Manifest property "Libraries"
- Classes defined in property "Class-Path"
- Classes in the directory WEB-INF/classes

The parent of base class-loader is dependency class-loader. Dependency class-loader has the list of dependent plugins. During class loading and resource loading phase, the class-loaders of these plugins are used.

The super parent of the class-loader is the Masking Class Loader, whose parent is the core class-loader. This masks the classes defined in Manifest property "Masked-Classes". The purpose of this class-loader is to allow plugins to load their own library of classes instead of using one from the "core" class-loader.



ORACLE®
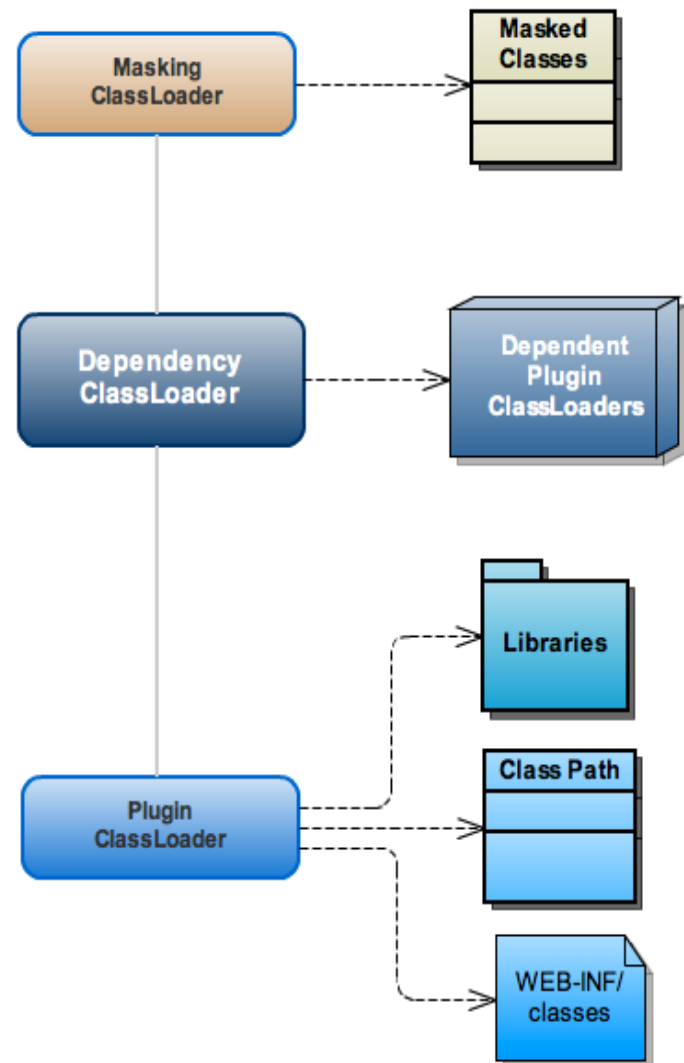
# Plugin Update Center and Update Site

Update Center and Update Sites are two Model Objects that has information about Update Sites and plugins available for install and update.

Hudson support multiple Update Sites. Plugin Manager uses these Model Objects to supply data for Plugin Management UI.

Update Center model Object is the owner of various jobs such as
- Connection Check Job
- Download Job
- Installation Job
- Plugin Upgrade Job
- Hudson Core Upgrade Job

These jobs are executed asynchronously when requests are received from UI.