


Business Process Model & Notation (BPMN) Workflows in Jenkins

Max Spring
Cisco

https://wiki.jenkins-ci.org/display/JENKINS/Jenkow+Plugin

Goals

- Attract users
- Get feedback
- Gain contributors

Agenda

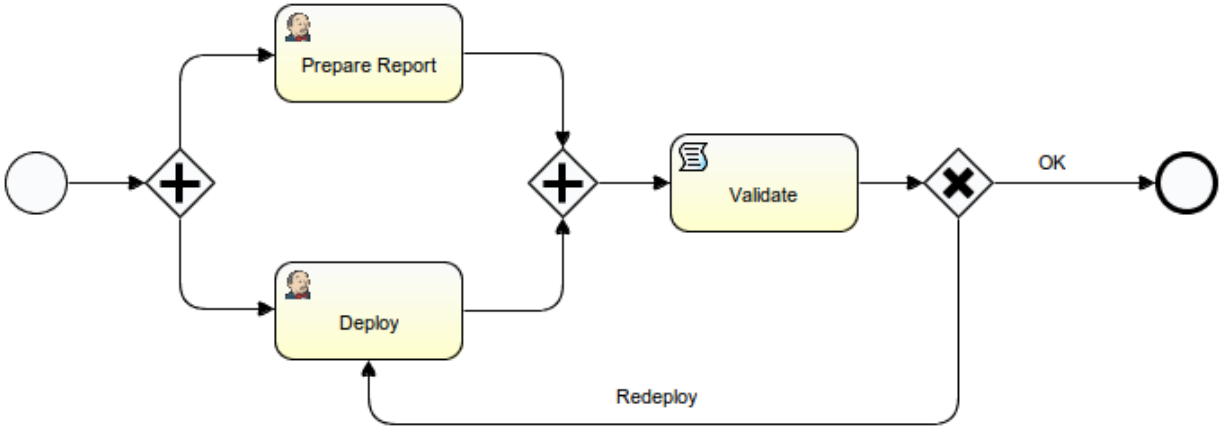
- What is BPMN & Activiti?
- How to setup & use Jenkow plugin
  - For Jenkins admins
- Plugin internals
  - For Plugin developers
- Future Features

Problem: Job Orchestration

- Putting jobs into relation
- Defining logic for execution of multiple jobs
- Existing orchestration mechanisms
  - Built-in Upstream / Downstream
  - Join Plugin
  - Locks and Latches Plugin
  - Drools Plugin (deprecated)
  - Build Flow Plugin
  - ...

BPMN

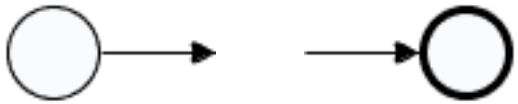
- Business Process Model & Notation
  - Similarity with UML activity diagrams
- BPMN < 2.0
  - Just modeling standard
- BPMN 2.0
  - Introduced in 2011
  - Added well-defined execution semantics



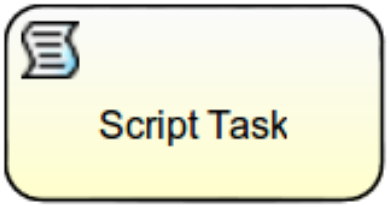
BPMN Concepts

- Process: workflow definition
- Process instance: “running” workflow
- Task: workflow step

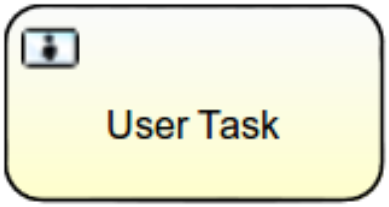
Basic BPMN Constructs



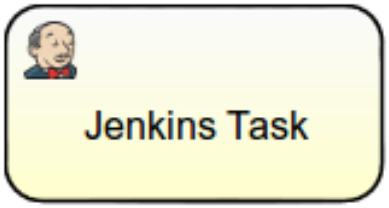
- Start / End Events



- Script Task
  - Groovy
  - Javascript
  - No concurrency!



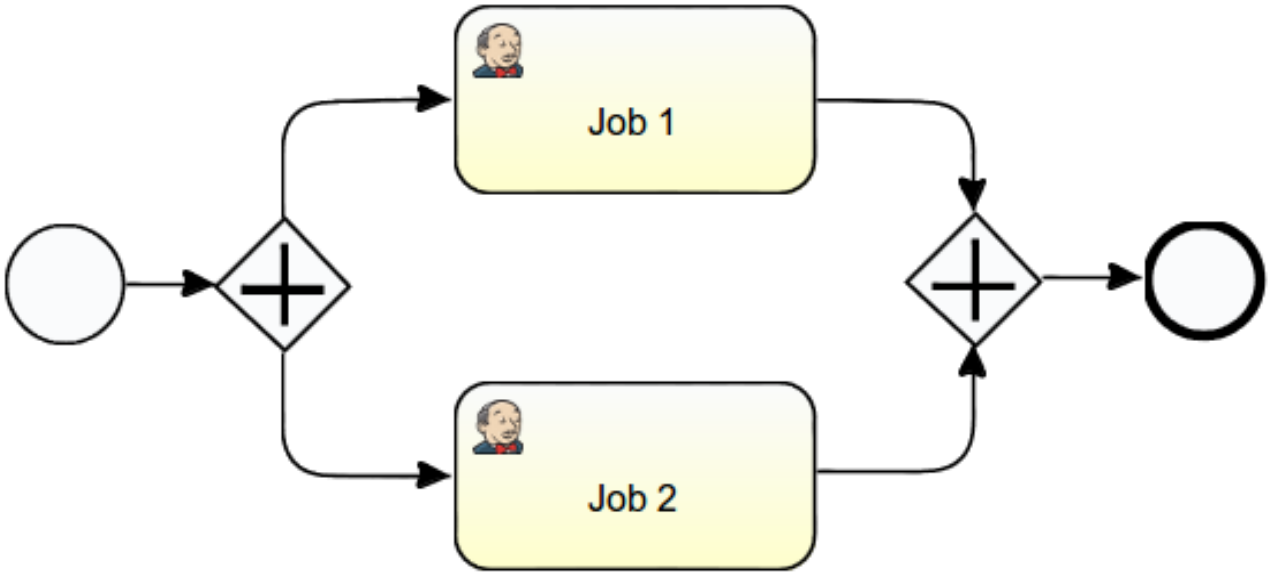
- User Task
  - Executed by real person



- Jenkins Task
  - Invoke Jenkins job

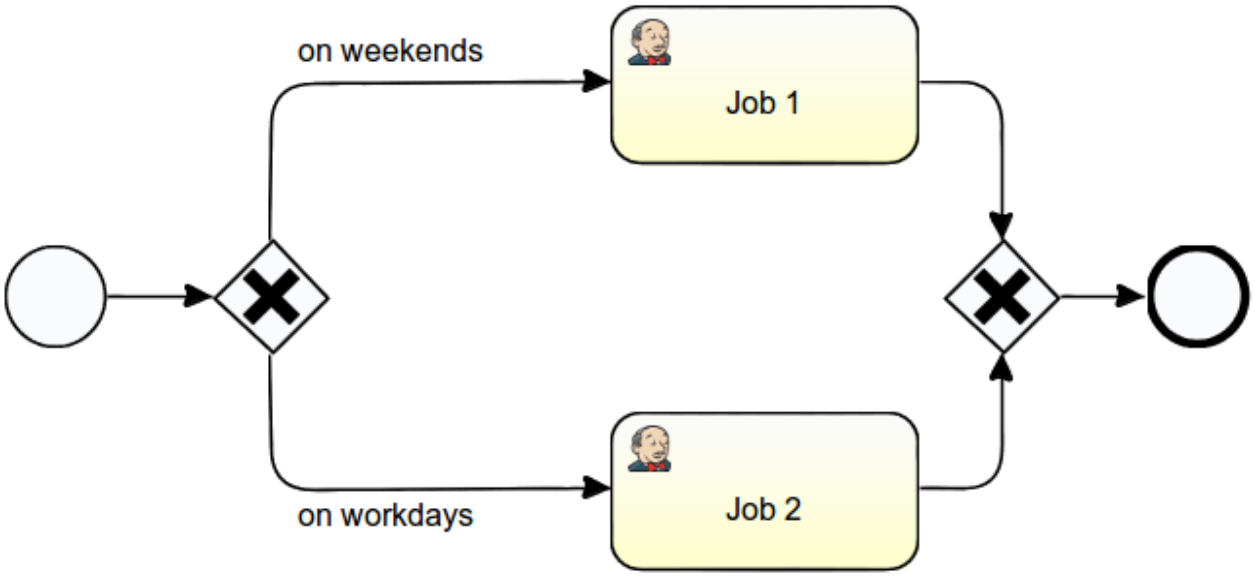
Basic BPMN Constructs

- Parallel Gateway

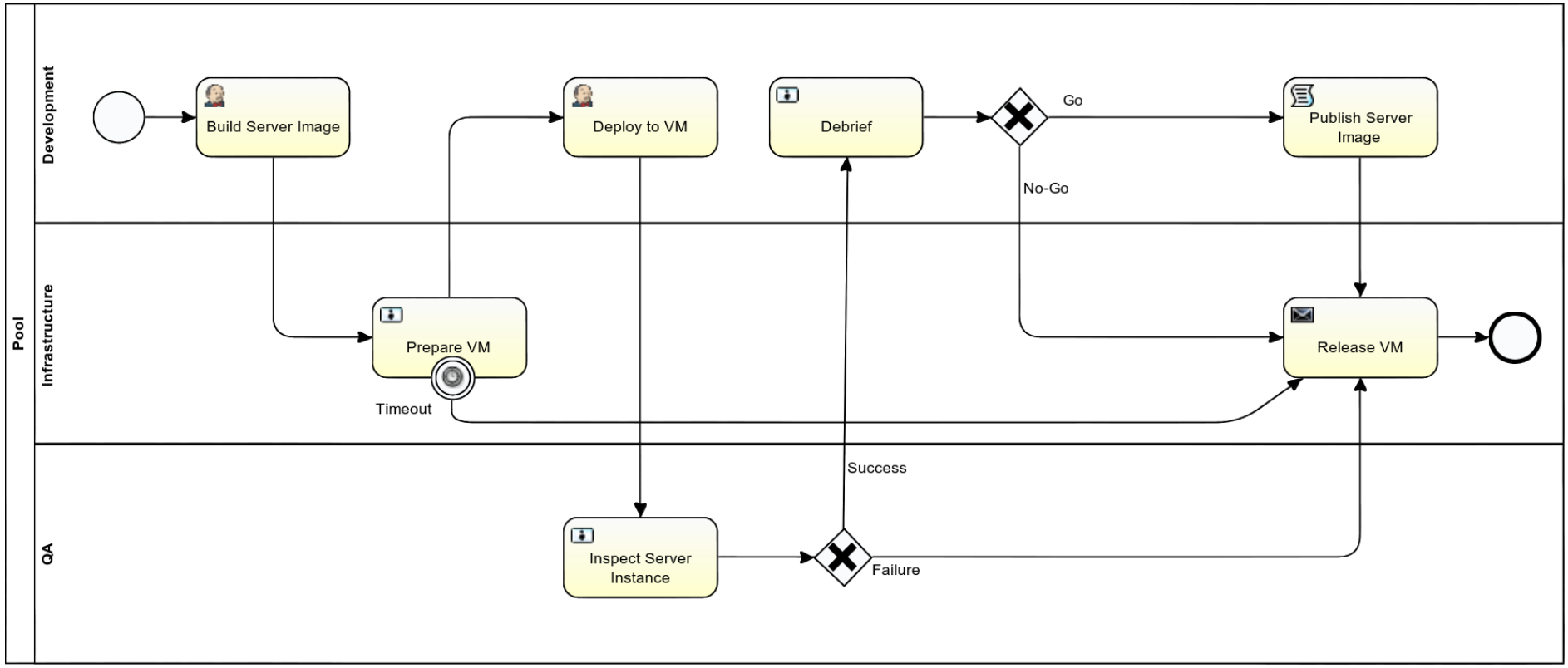


Basic BPMN Constructs

- Exclusive Gateway



More complex workflow



BPMN Benefits

- It's a standard
- Geared towards non-software developers
- Constructs for interactions with other actors (persons and systems)
- Powerful graphical notation
- Existing tooling
- Well-documented XML

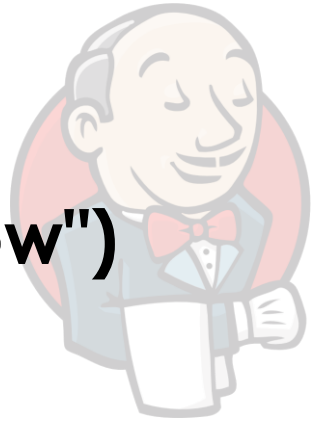
Activiti™

- http://activiti.org
- Open Source (Apache License)
- BPMN 2.0 workflow engine
- Implemented in Java
- Web-based Workflow Management
- Workflow Editor
  - Activiti Designer (Eclipse Plugin)
    - Extensible
  - Web-based Activiti Modeler
    not under active development

Earlier Activiti / Jenkins Integration

- Part of a larger commit automation effort
- Jenkins as Execution Engine
- Activiti Workflow Engine in Apache ServiceMix Container (OSGi)
- Using Jenkins' Channel (ssh) mechanism

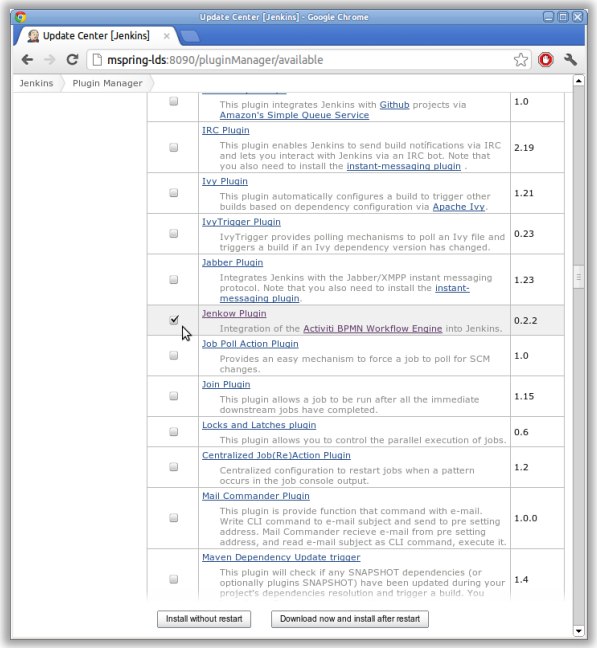
Jenkow Overview

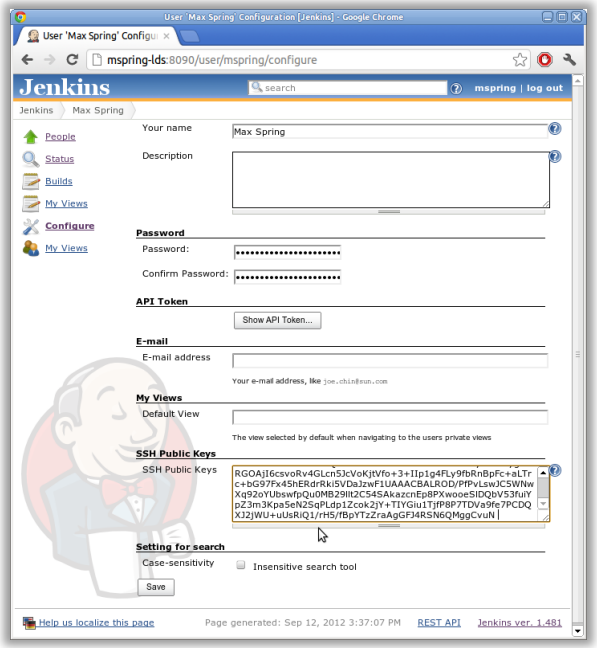


- Jenkins plugin ("Jenkins On Workflow")
- Early release (v0.2.4)
- Emphasis on workflow editor
- Activiti Engine in Jenkins
- Activiti Designer in Eclipse
  - Bundled with Jenkow
  - Served by Jenkins Update Site plugin
- Git repository in Jenkins to store workflows
  - Using Git Server plugin (thanks Kohsuke)
  - Version-controlled workflows

Jenkow Setup (one time)

- Install plugin from "Available" catalog
- If authorization is on, configure SSH public key
  - To allow push into workflow Git repository
  - People⇨*userid*⇨Configure⇨SSH Public Keys

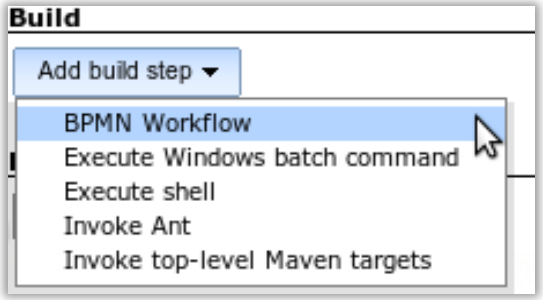


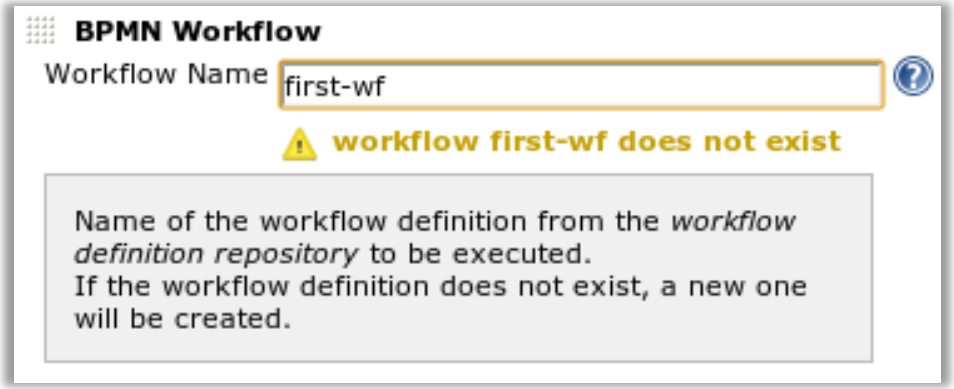


<demo>

Jenkow Setup

- **Create first job with workflow step**
  - Job Configure⇨Build⇨Add build step ⇨BPMN Workflow
    Workflow Name: first-workflow
  - **If workflow doesn't exist, a simple "hello world" workflow gets created**





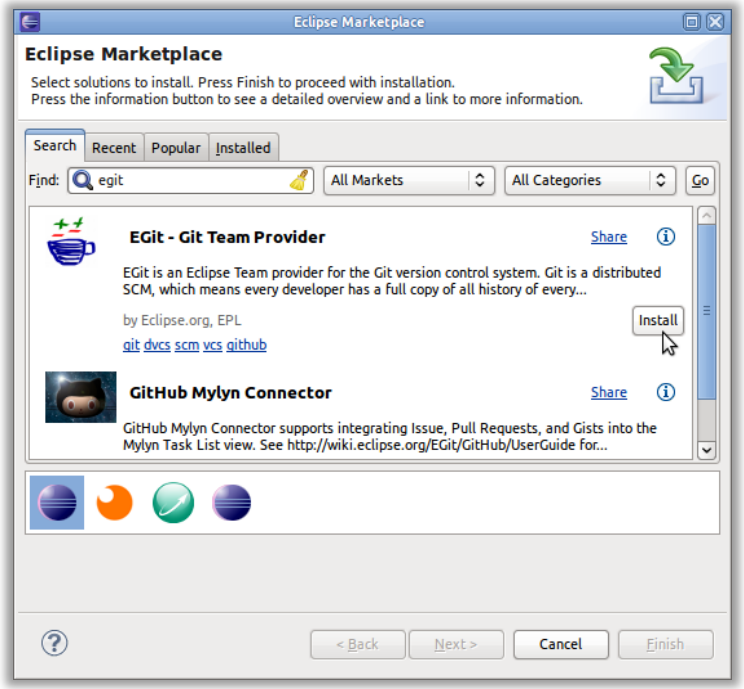
Jenkow Setup

- Build first job

Workflow Editor Setup (one time)

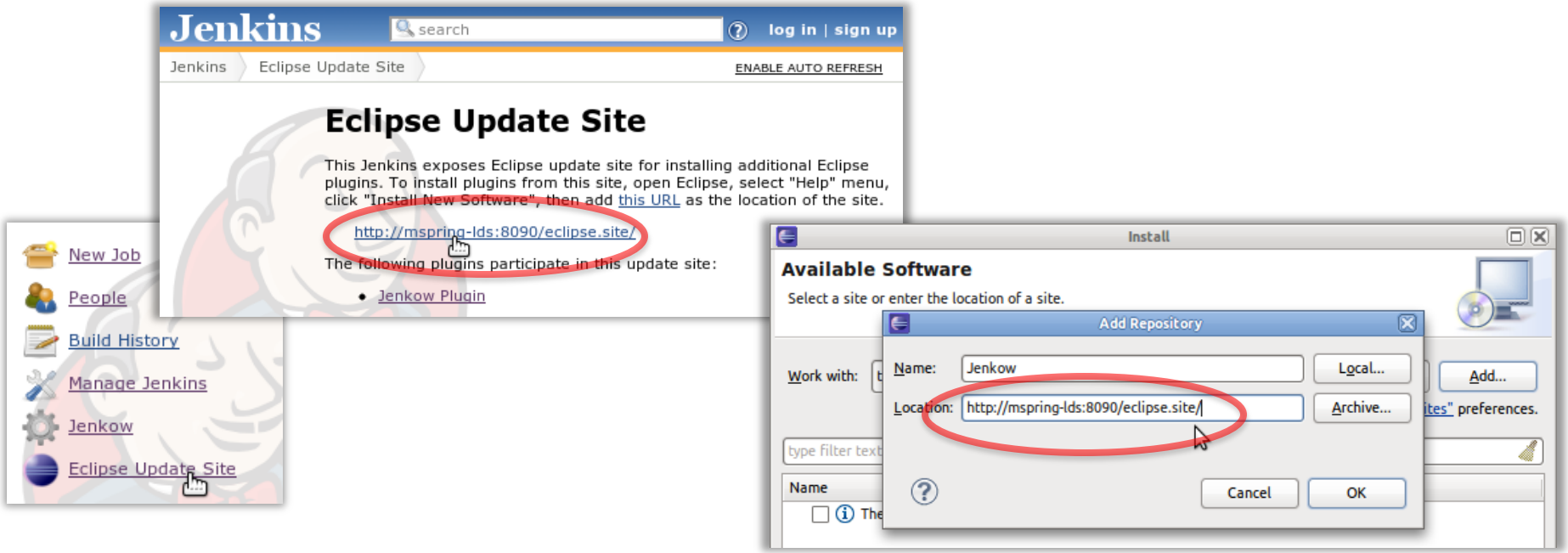
- **Get Eclipse (3.7 Indigo)**
  - http://www.eclipse.org/downloads/ (Eclipse IDE for Java Developer)

- **Or install EGit**
  - Help
    ⇨Eclipse Marketplace…
    ⇨Find:egit⇨Install



Workflow Editor Setup (one time)

- **Install Jenkow Activiti Designer**
  - Jenkins: Eclipse Update Site ⇨ *copy URL*
  - Eclipse:
    Help⇨Install new Software…⇨Add…
    ⇨Name:Jenkow, Location: *paste URL* ⇨OK



Import Workflow Project (one time)

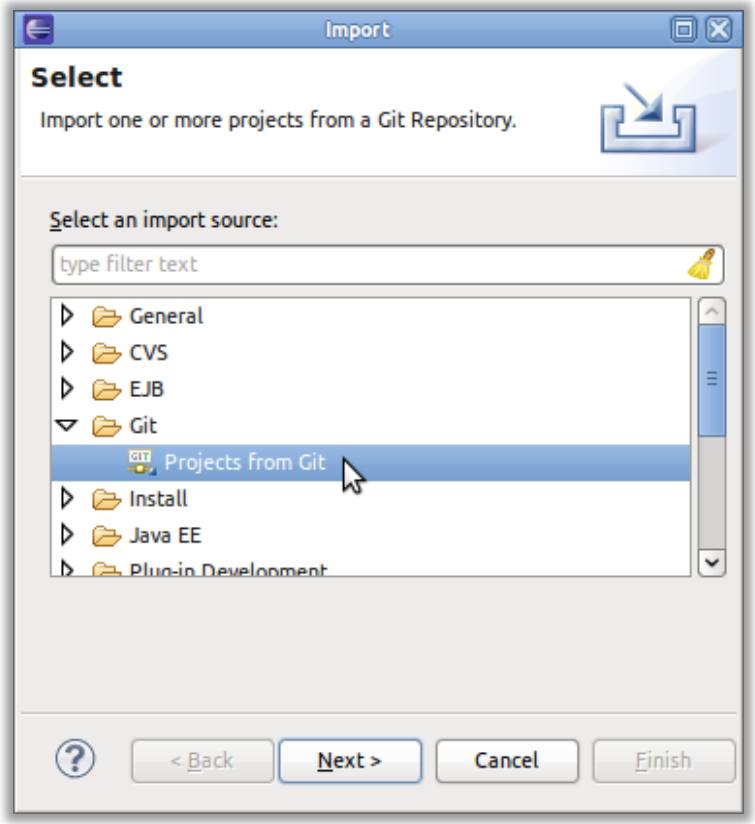
- **Git repository in Jenkins**

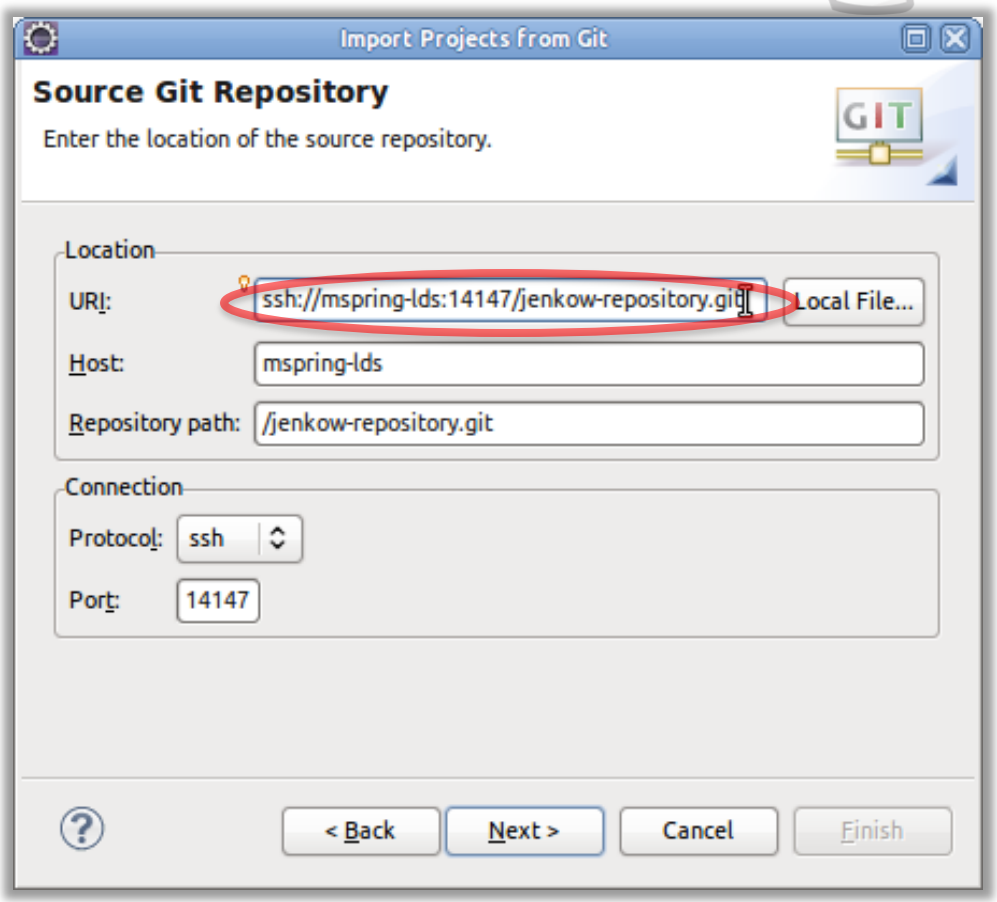
  Jenkow: *copy Git ssh URI*

Import Workflow Project (one time)

Import in Eclipse (1/2)

File⇨Import…⇨Git⇨Projects from Git⇨Next
URI⇨Next ⇨ URI: *paste Git URI*⇨Next

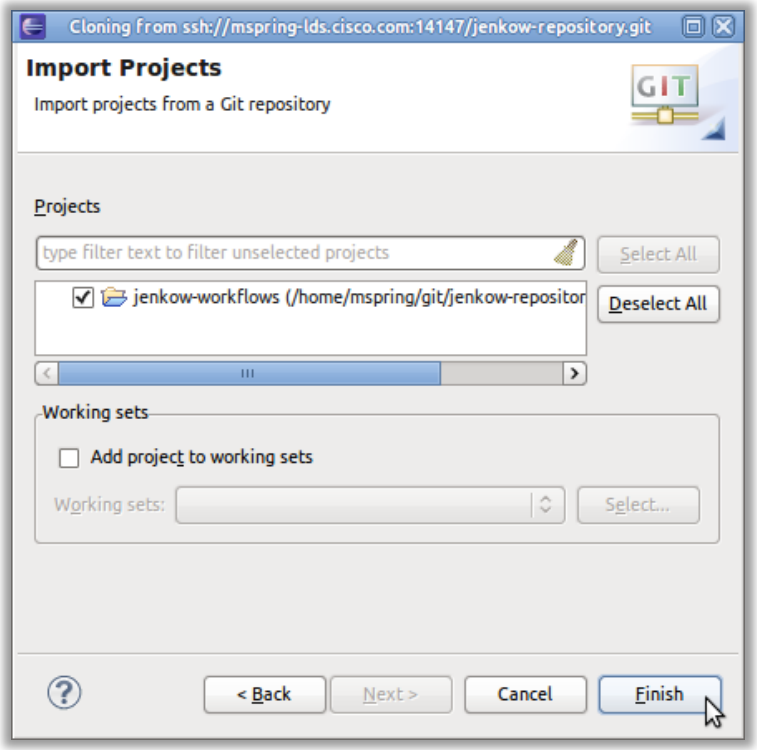




Import Workflow Project (one time)

- **Import in Eclipse (2/2)**

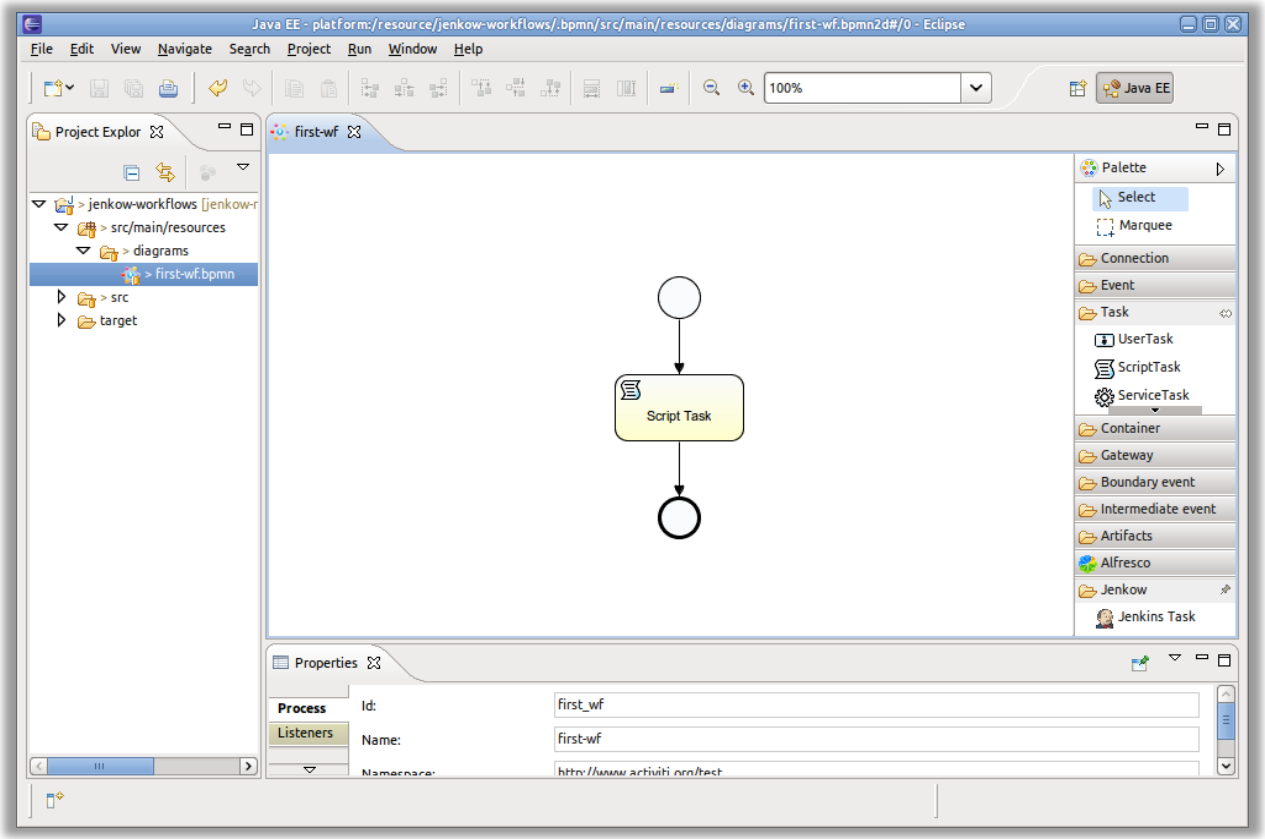
  …
  Branch selection: select master⇨Next
  Local Destination … ⇨Next
  …Import existing project…⇨Next
  Import Projects:
  select jenkow-workflows
  ⇨Finish



Import Workflow Project (one time)

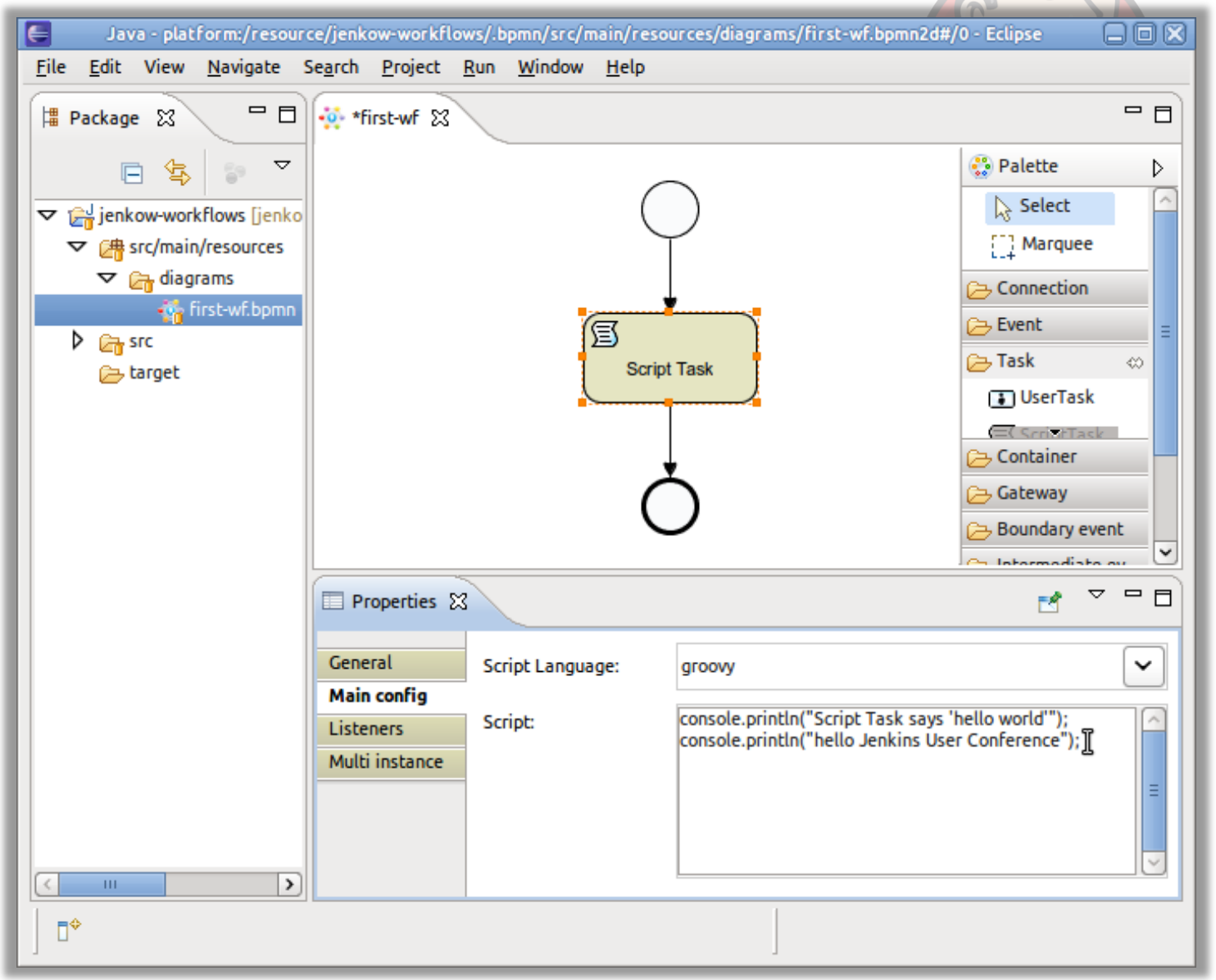
- **Open Workflow Project**

  Project Explorer: jenkow-worfklows
  ⇨src/main/resources⇨diagrams⇨first-wf.bpmn



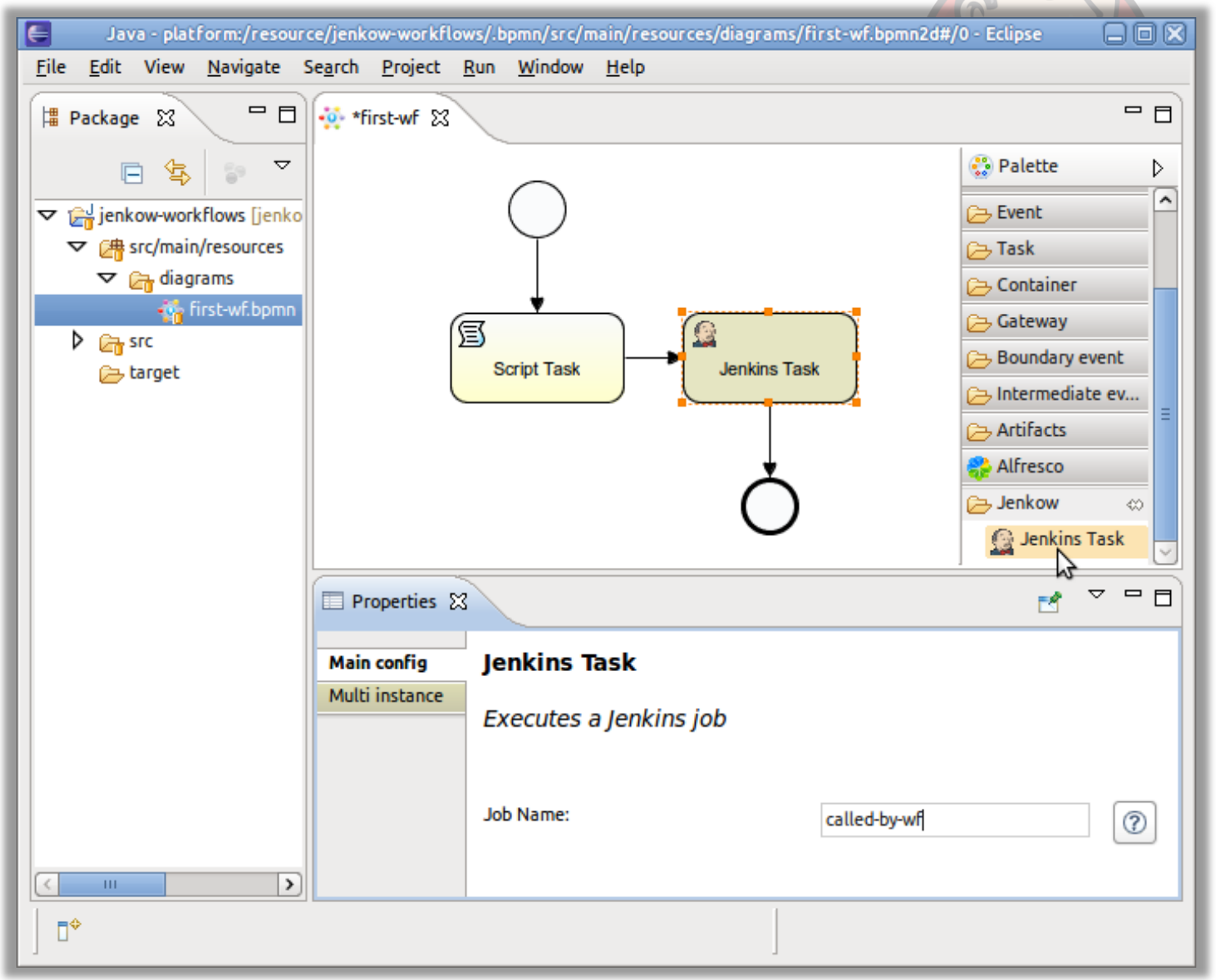
Edit Workflow

- Script Task



Edit Workflow

- Jenkins Task



Push changed Workflow

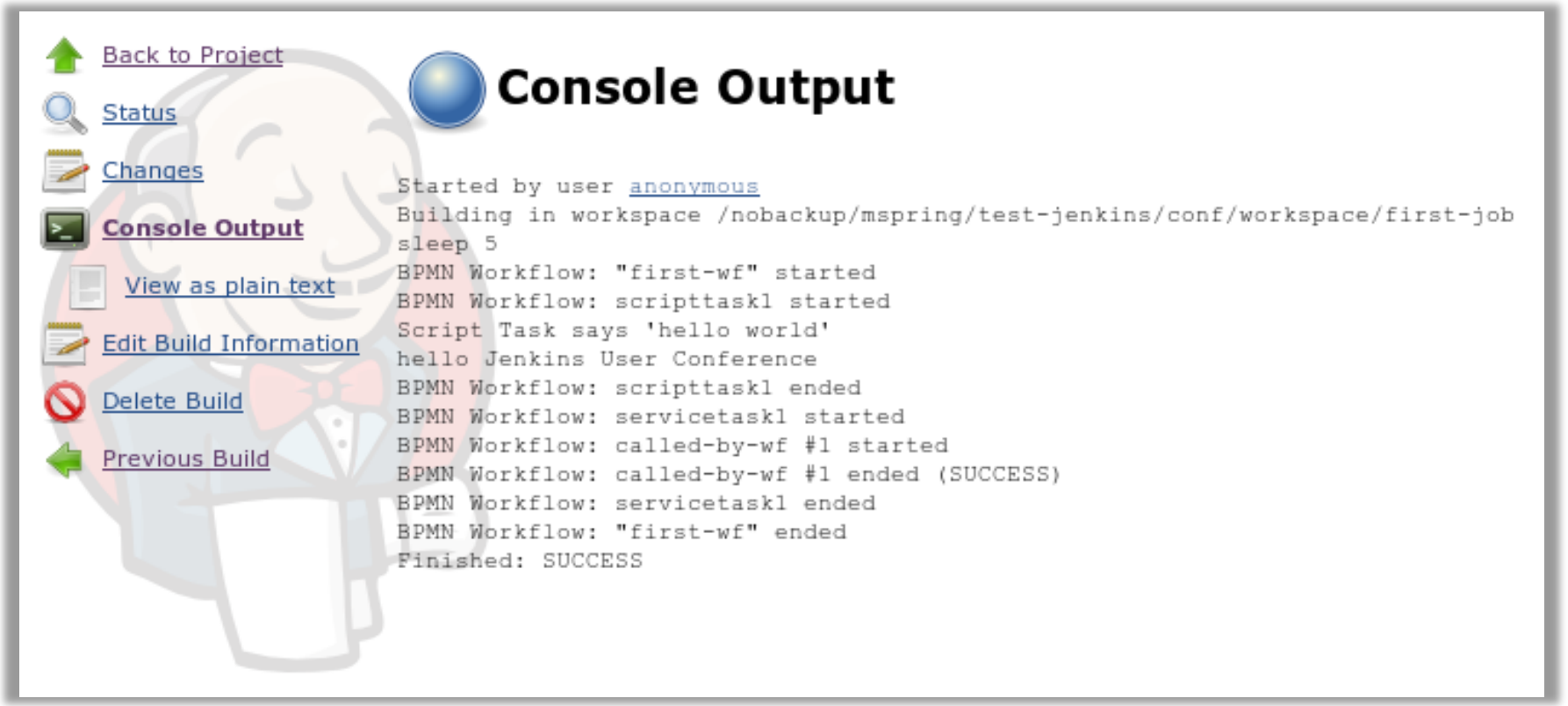
- **Save workflow**
- **Commit**

  Project Explorer: right-click jenkow-workflows ⇨Team⇨Commit…
  ⇨*Commit message*⇨Commit
- **Push**

  Project Explorer: right-click jenkow-workflows ⇨Team⇨Push to Upstream⇨OK

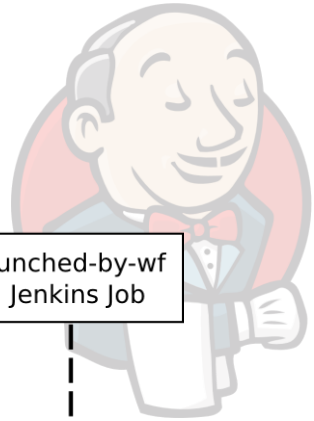
Run changed Workflow

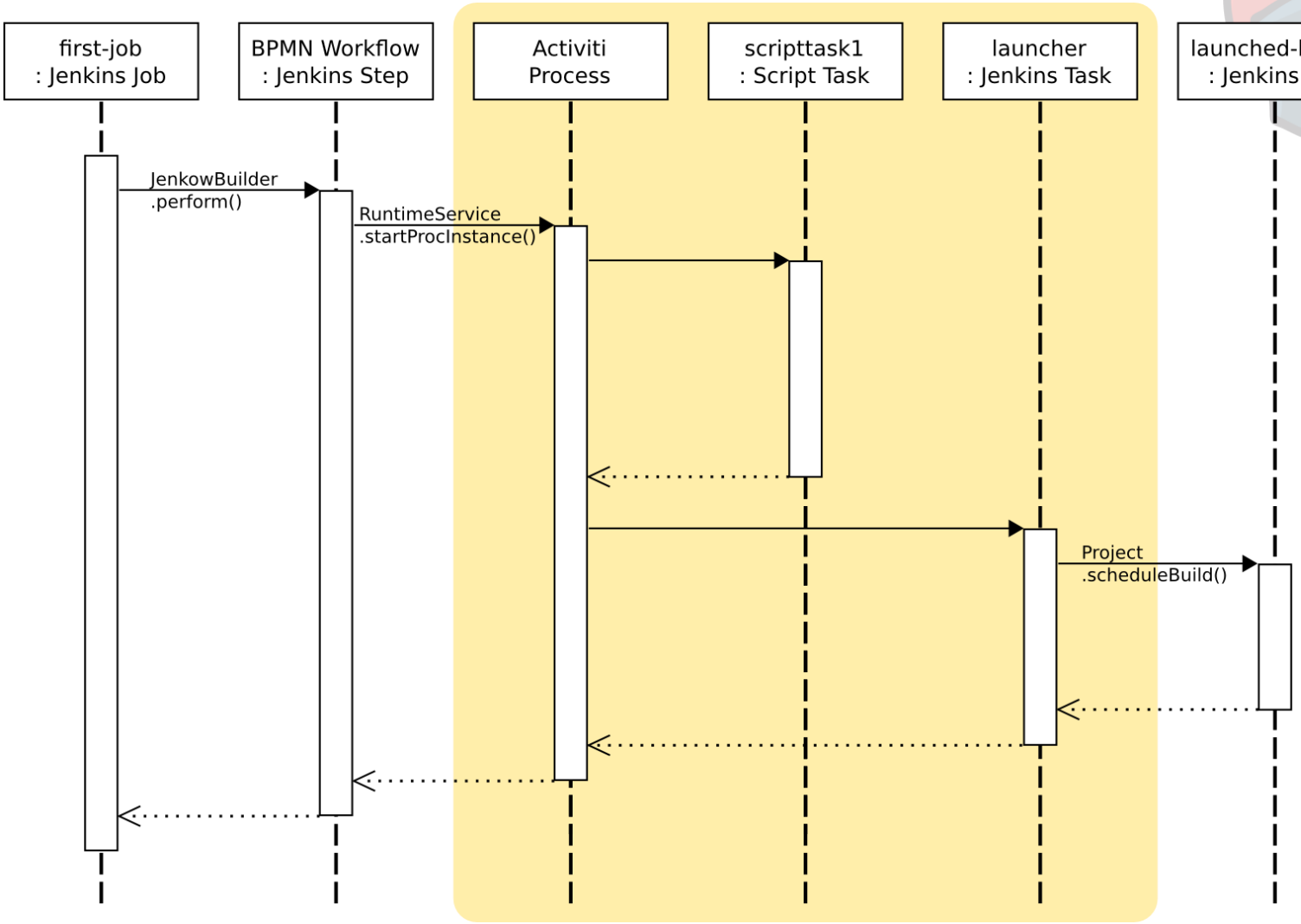
- Create “called-by-wf” job
- Kick “first-job”



</demo>

Example Launch Sequence



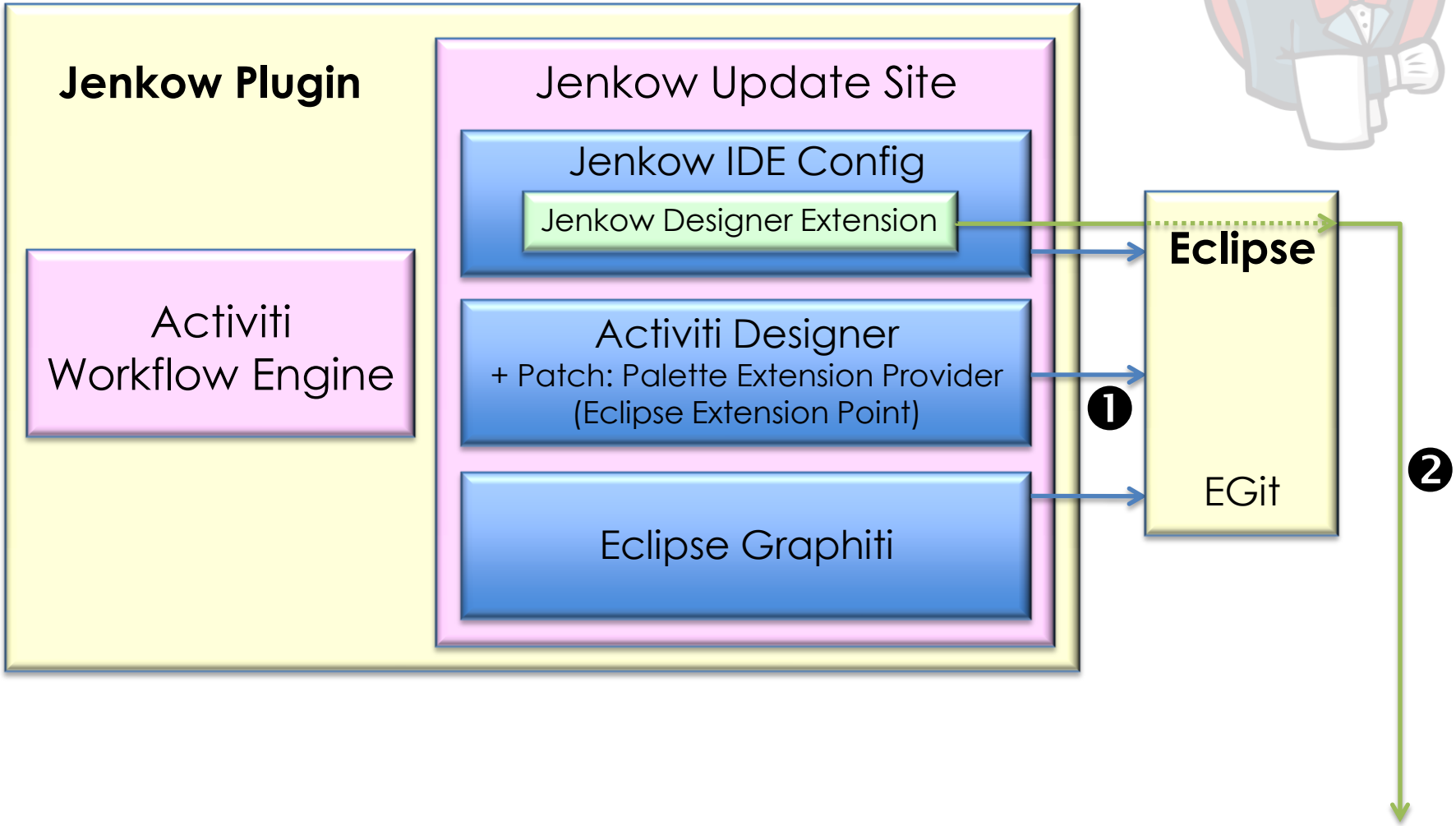


Implementation Details

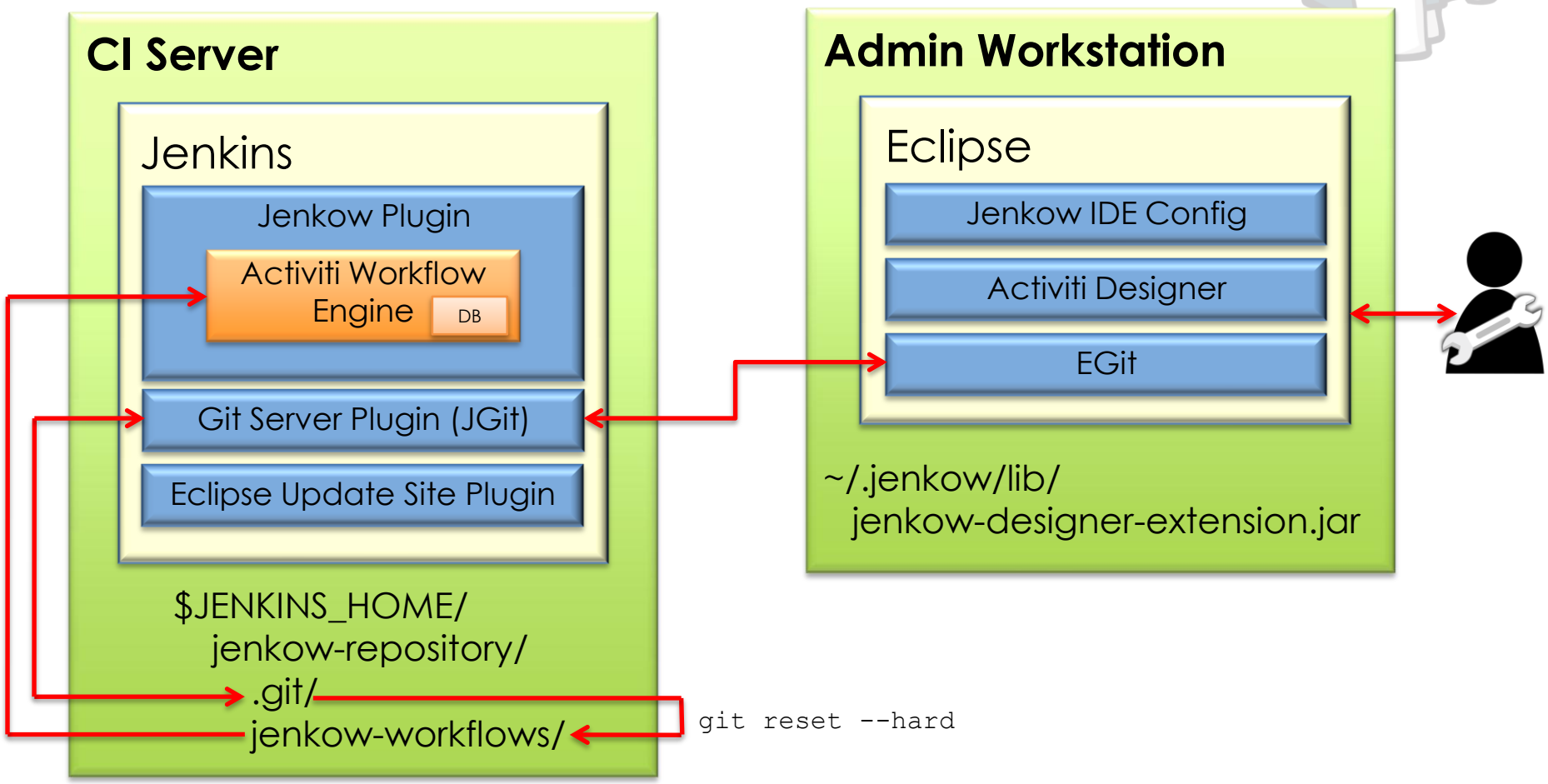
Execution Modes

- "Script Mode"
  - Build step
  - Occupies executor
  - No persistency
  - Short running
- "Workflow Mode" (not yet implemented)
  - Workflow Job Type
  - No executor wasted
  - Workflow state persisted in DB
  - Long "running" (days, weeks, …)
  - Interact with other systems / actors

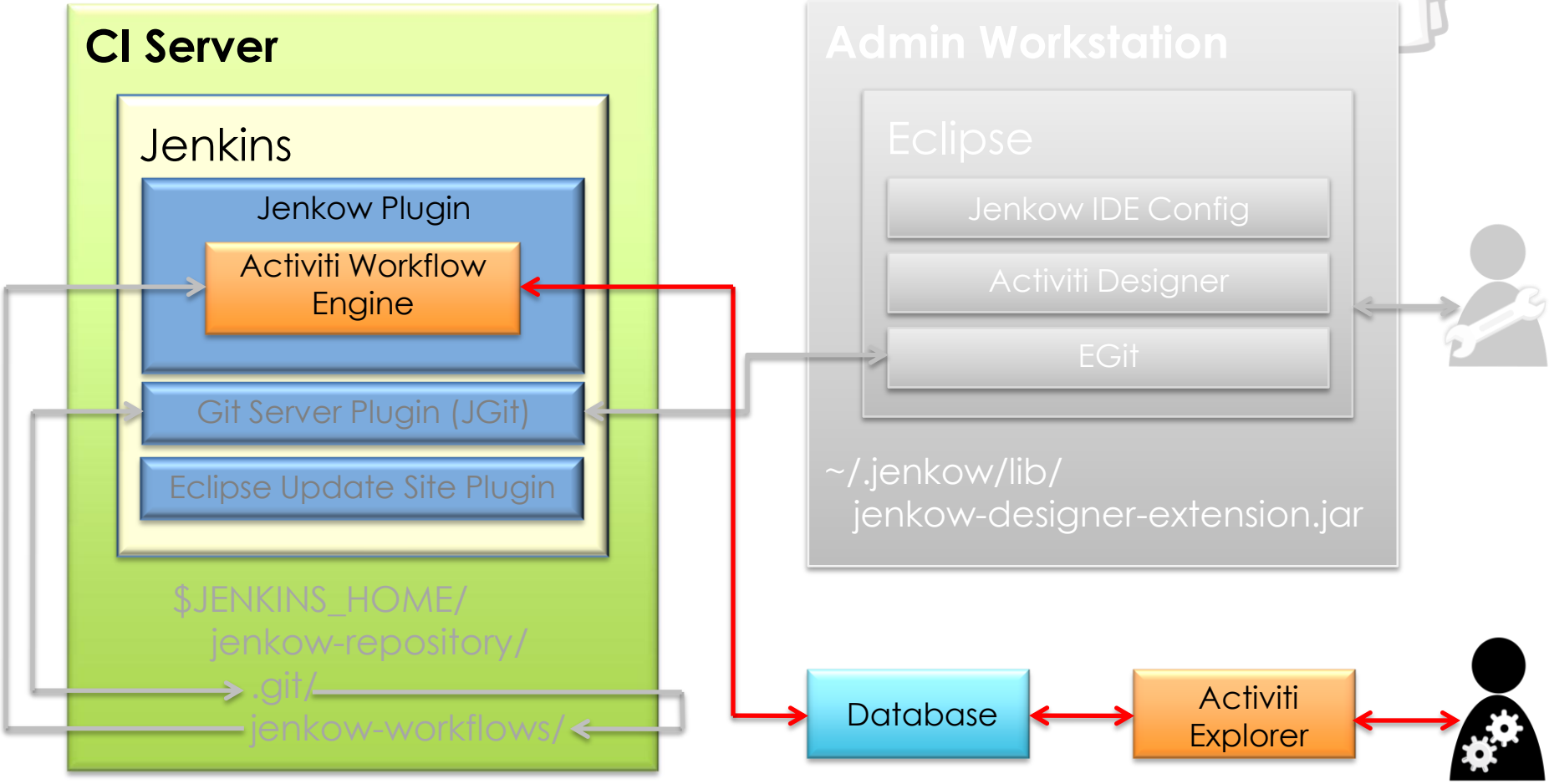
Plugin Elements



Run Time


CI Server
Jenkins
Jenkow Plugin
Activiti Workflow Engine
DB
Git Server Plugin (JGit)
Eclipse Update Site Plugin
$JENKINS_HOME/
jenkow-repository/
.git/
jenkow-workflows/
git reset --hard
Admin Workstation
Eclipse
Jenkow IDE Config
Activiti Designer
EGit
~/.jenkow/lib/
jenkow-designer-extension.jar

Run Time & External DB



JenkowBuilder (start workflow)


```
public class JenkowBuilder extends Builder {
    private String workflowName;

    @Override
    public boolean perform(AbstractBuild build, ...) {
        ProcessEngine eng = JenkowEngine.getEngine();
        RuntimeService rtSvc = eng.getRuntimeService();
        RepositoryService repoSvc = eng.getRepositoryService();

        File wff = getWorkflowFile(workflowName);
        DeploymentBuilder db = repoSvc
                               .createDeployment()
                               .addInputStream(wff.getName(),new FileInputStream(wff));

        Deployment d = db.deploy();
        ProcessDefinition pDef = repoSvc
                                 .createProcessDefinitionQuery()
                                 .deploymentId(d.getId())
                                 .singleResult();

        Map<String,Object> varMap = new HashMap<String,Object>();
        varMap.put("jenkow_build_parent",build.getParent().getName());
        varMap.put("jenkow_build_number",Integer.valueOf(build.getNumber()));
        varMap.put("console",new ConsoleLogger());

        String procId = rtSvc.startProcessInstanceById(pDef.getId(),varMap).getId();
        ...
    }
}
```

JenkowBuilder (end workflow)


```
public class JenkowBuilder extends Builder {

    @Override
    public boolean perform(AbstractBuild build, ...) {
        ...
        HistoryService hstSvc = eng.getHistoryService();
        while (true){
            HistoricProcessInstance hProcInst = hstSvc
                                                .createHistoricProcessInstanceQuery()
                                                .processInstanceId(procId)
                                                .singleResult();
            if (hProcInst.getEndTime() != null){
                break;
            }
            Thread.sleep(1000);
        }
        return true;
    }
}
```

JenkinsTask


```
@Runtime(delegationClass = "JenkinsTaskDelegate")
@Help(displayHelpShort = "Executes a Jenkins job")
public class JenkinsTask extends AbstractCustomServiceTask {

    @Property(type = PropertyType.TEXT, displayName = "Job Name")
    @Help(displayHelpShort = "Name of the Jenkins job to execute.")
    private String jobName = "${executionContext.host}";

    @Override
    public String contributeToPaletteDrawer() {
        return "Jenkow";
    }

    @Override
    public String getName() {
        return "Jenkins Task";
    }

    @Override
    public String getSmallIconPath() {
        return "jenkins.png";
    }
}
```

JenkinsTaskDelegate


```
public class JenkinsTaskDelegate extends ReceiveTaskActivityBehavior{
    private Expression jobName;

    @Override
    public final void execute(ActivityExecution exec) throws Exception {
        String aid = exec.getActivity().getId();

        String jn = (jobName == null)? null : jobName.getValue(exec).toString();
        if (jn != null){
            TopLevelItem it = Jenkins.getInstance().getItem(jn);
            if (it instanceof Project){
                Project p = (Project)it;

                DescribableList wrappers = p.getBuildWrappersList();
                JenkowBuildWrapper wrapper = new JenkowBuildWrapper();
                if (!wrappers.contains(wrapper.getDescriptor())) wrappers.add(wrapper);

                JenkowAction ja = new JenkowAction(aid,exec.getId());
                p.scheduleBuild2(new WorkflowCause("triggered by workflow"),ja);
                return;
            }
        }
        leave(exec); // declare task as done
    }
}
```

Future Features

- Workflow Job Type (workflow mode)
  - How to manage active workflows?
- Workflow diagrams in Jenkins
  - Showing execution state
- Integrate Activiti Web UIs?
  - Activiti Modeler

Future Features

- Job parameter support
- More task types for other Jenkins functions
  - Slave node management
  - Build promotion
  - …
- Multi-instance task support

Future Features

- Find alternative to bundling Eclipse update site with Jenkow plugin
- Other mechanisms to expose workflow repository
  - Eclipse EFS over SSH

Jenkow Take Away

- Use for complex job orchestration
- Git Server Plugin usage example
- Eclipse Update Site usage example
  - Maven / Tycho integration

- Resources

  https://wiki.jenkins-ci.org/display/JENKINS/Jenkow+Plugin

  http://www.activiti.org/

Thank You To Our Sponsors

CloudBees

CLOUDANT

salesforce

Sonatype

sauce LABS

hp

SendGrid
Email Delivery. Simplified.

LIFERAY

Confreaks

ZEROTURNAROUND

YAHOO!

WAN DISCO

New Relic