CI/CD with OpenShift, Jenkins, and Blue Ocean

Objectives

Key objectives of this chapter

- OpenShift
- Jenkins
- Integrating OpenShift with Jenkins
- Jenkins Pipeline
- Blue Ocean

1.1 What is OpenShift

- OpenShift is a computer software product from Red Hat
- It provides a container-based software deployment and management solution
- In short, it's a container orchestration software
- It is a supported distribution of Kubernetes using Docker containers and DevOps tools

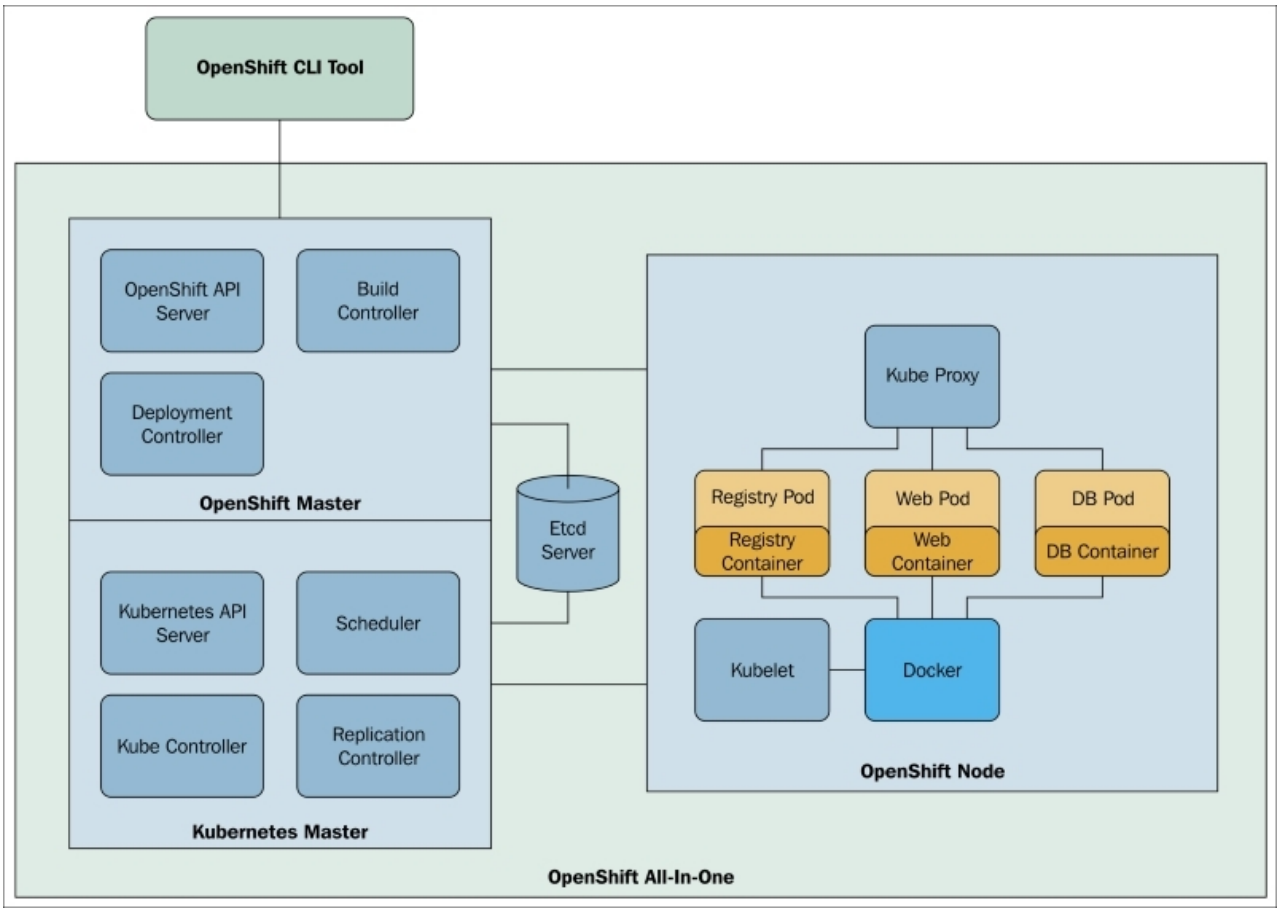
1.2 OpenShift Online

- It is Red Hat's public cloud application development and hosting service.
- It is also available under the Apache License Version
- Online supports a variety of languages, frameworks, and databases via pre-built "cartridges" running under resource-quota "gears"
- Developers can add other language, database, or components via the OpenShift application programming interface which has been deprecated in favor of OpenShift 3

1.3 OpenShift Origin

- Origin provides an open source application container platform
- It is the upstream community project used in OpenShift Online
- It is built around a core of Docker container packaging and Kubernetes container cluster management
- Origin is augmented by application life cycle management functionality and DevOps tooling
- All source code for Origin project is available under the Apache License on GitHub

1.4 OpenShift Architecture



1.5 OpenShift Origin Installation

- Setup Docker
  - #edit /etc/sysconfig/docker file and add --insecure-registry 172.30.0.0/16 to the OPTIONS parameter.
- Install required packages

```
yum install centos-release-openshift-origin
```

```
yum install wget git net-tools bind-utils iptables-services
bridge-utils bash-completion origin-clients
```

- Method 1: Origin

```
yum install centos-release-openshift-origin
yum install origin-clients
oc cluster up
```

- Method 2: Running from a RPM

```
yum install origin

openshift start

export KUBECONFIG="$
(pwd)"/openshift.local.config/master/admin.kubeconfig

export CURL_CA_BUNDLE="$
(pwd)"/openshift.local.config/master/ca.crt

sudo chmod +r "$
(pwd)"/openshift.local.config/master/admin.kubeconfig
```

- Method 3: Running in a Docker Container

```
docker run -d --name "origin" \
    --privileged --pid=host --net=host \ -v /:/rootfs:ro -v
    /var/run:/var/run:rw -v /sys:/sys -v
    /var/lib/docker:/var/lib/docker:rw \ -v
    /var/lib/origin/openshift.local.volumes:/var/lib/origin
    /openshift.local.volumes \
    registry.centos.org/openshift/origin start
```

- Method 4: Using Ansible

```
yum install centos-release-openshift-origin

yum --enablerepo=centos-openshift-origin-testing clean all
```

```
yum --enablerepo=centos-openshift-origin-testing install
atomic-openshift-utils

atomic-openshift-installer install
```

1.6 OpenShift CLI

- oc is the main command line tool
- Start OpenShift

```
oc cluster up
```

- Stop OpenShift

```
oc cluster down
```

- Check status

```
oc status
```

- Check user

```
oc whoami
```

- Connect as admin

```
oc login -u system:admin
```

- Get projects list

```
oc projects
```

- Create new project

```
oc new-project <prj>
```

- Make a project the active project

```
oc project <prj>
```

1.7 OpenShift CLI (Contd.)

- Create a new OpenShift application from a docker image

```
oc new-app –docker-image="<docker-image>"
```

- Get pods list

```
oc get pods
```

- Get service details

```
oc describe svc/<service-name>
```

- Expose a service

```
oc expose svc/<service-name> --name=<internet name>
--hostname=<exposed hostname>
```

- Scale pod

```
oc scale dc/<app> --replicas=<count>
```

- Delete a service

```
oc delete svc/<service-name>
```

1.8 Jenkins Continuous Integration

- Originally developed at Sun by Kohsuke Kawaguchi?
  - Originally “Hudson” on java.net circa 2005
  - Jenkins forked in November 2010
  - Hudson is still live, part of Eclipse Foundation
  - But Jenkins seems to be far more active

1.9 Jenkins Features

- Executes jobs based on a number of triggers
  - Change in a version control system
  - Time
  - Manual Trigger
- A Job consists of some instructions

- Run a script
- Execute a Maven project or Ant File
- Run an operating system command

- User Interface can gather reports
  - Each job has a dashboard showing recent executions

1.10 Running Jenkins

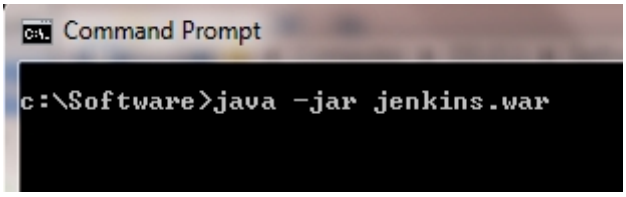
- You can run Jenkins Standalone or inside a web container
- You can setup distributed instances that cooperate on building software
- Can setup jobs in place of what might have been script commands.

1.11 Downloading and Installing Jenkins

- Download Jenkins from the Jenkins website (http://jenkins-ci.org)
  - Jenkins is a dynamic project, and new releases come out at a regular rate.
- Jenkins distribution is bundled in Java web application (a WAR file).
- Windows users, there is a graphical Windows installation MSI package for Jenkins.

1.12 Running Jenkins as a Stand-Alone Application

- Jenkins comes bundled as a WAR file that you can run directly using an embedded Servlet container.
- Jenkins uses the lightweight Servlet engine to allow you to run the server out of the box.
- Flexible to install plug-ins and upgrades on the fly.
- To run Jenkins using the embedded Servlet container, just go to the command line and type the following:



- The Jenkins web application will now be available on port 8080.

- Jenkins can be access directly using the server URL (http://localhost:8080).
- To stop Jenkins, just press Ctrl-C.
- Useful Options:
- --httpPort
    - By default, Jenkins will run on the 8080 port.
    - Jenkins can be start on different port using the --httpPort option:
    - java -jar jenkins.war –httpPort=8081
- --logfile
    - By default, Jenkins writes its logfile into the current directory.
    - Option to redirect your messages to other file:
    - java -jar jenkins.war --logfile=C:/Software/log/jenkins.log
- These options can also be set at JENKINS_HOME/jenkins.xml config file.

1.13 Running Jenkins on an Application Server

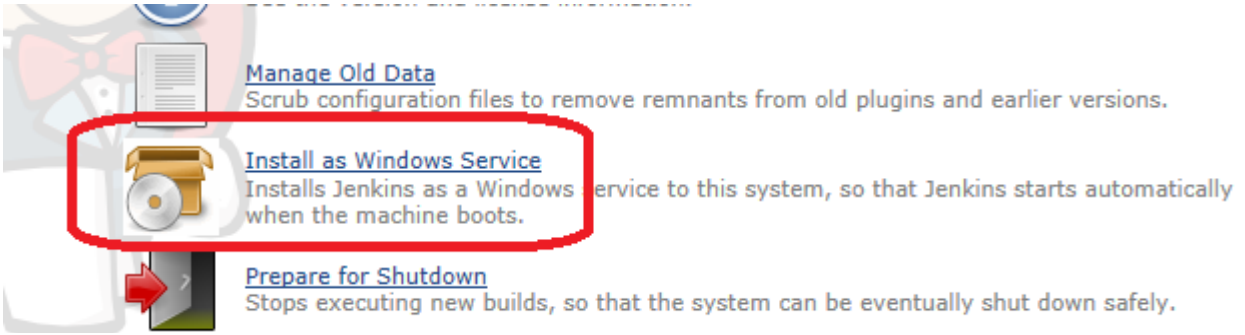
- Jenkins distribution WAR file can be easily deploy to standard Java application server such as Apache Tomcat, Jetty, or GlassFish.
- Jenkins will be executed in its own web application context (typically “jenkins”).
    - URL : http://localhost:8080/jenkins.

1.14 Installing Jenkins as a Windows Service

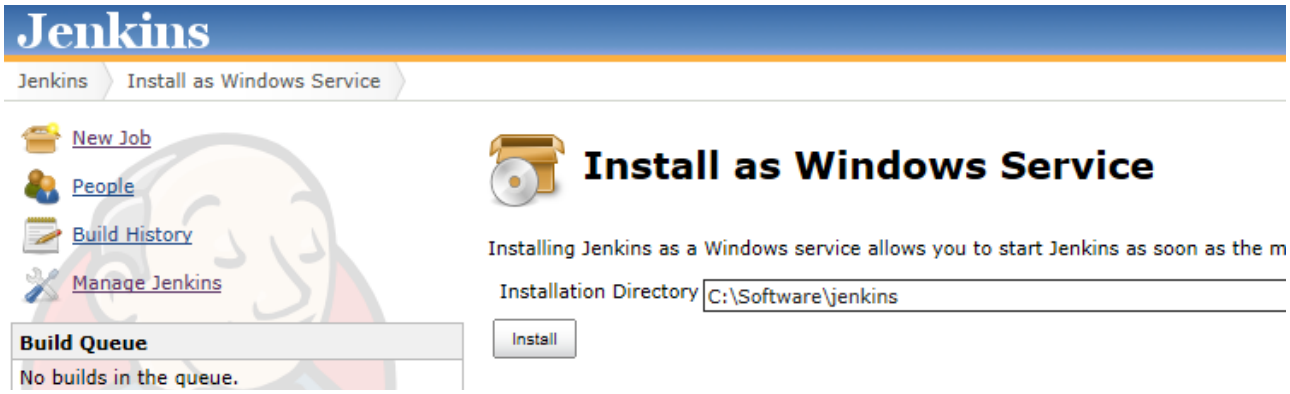
- In production, installation of Jenkins on a Windows box is essential to have it running as a Windows service.

- Jenkins will automatically start whenever the server reboots
- Can be managed using the standard Windows administration tools.

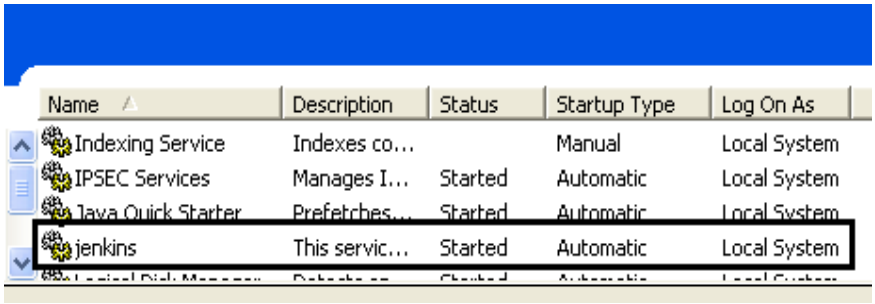
- Jenkins using the windows installer automatically runs Jenkins as a windows service.
  - No need to do anything.
- First, Start the Jenkins server on your machine:
  - java -jar jenkins.war
- Connect to Jenkins by going to the following URL http://<hostname>:8080
- Look for "Install as Windows Service" link in the "Manage Jenkins" page.



- Clicking this link shows you the installation screen:



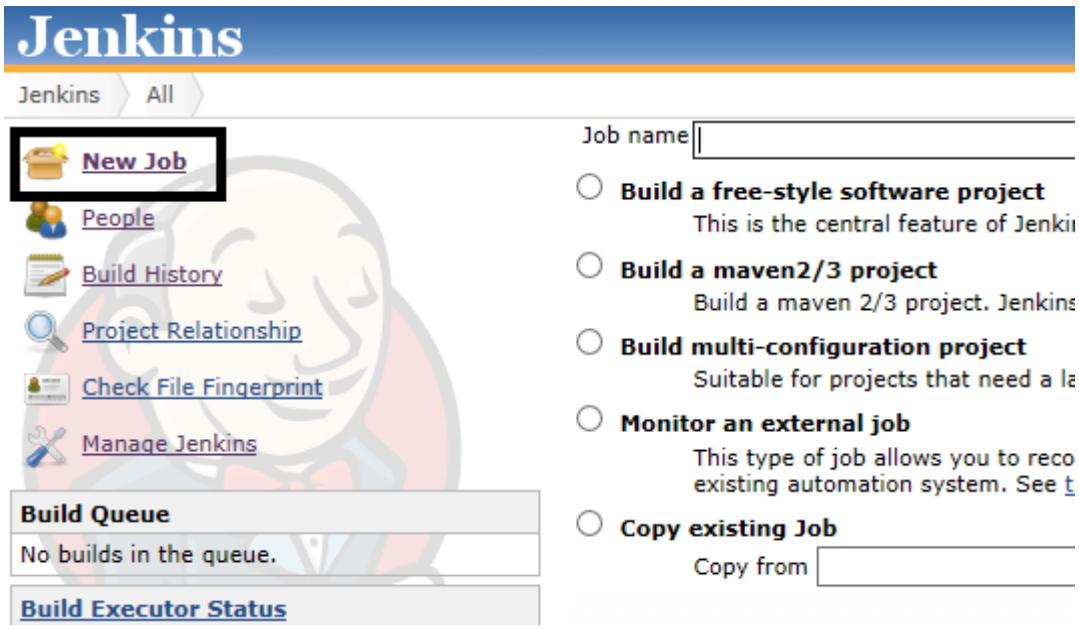
- Choose a directory where Jenkins will be installed, JENKINS_HOME and used to store data files and programs
- Upon successful completion of the installation, you should see a page asking you to restart Jenkins.
- At this point you can use the service manager to confirm that Jenkins is running as a service.



1.15 Different types of Jenkins job

- Jenkins supports several different types of build jobs
    - Freestyle software project:
        - Freestyle projects are general purpose and allow to configure any sort of build job.
        - Highly flexible and very configurable.
    - Maven project:
        - Jenkins understands Maven pom files and project structures.
        - Reduce the work needed to do to set up the project.
    - Monitor an external job:
        - Monitoring the non-interactive execution of processes, such as cron jobs.

- Multi-configuration job:
  - Run same build job in many different configurations.
  - Powerful feature, useful for testing an application in many different environments.



1.16 Configuring Source Code Management(SCM)

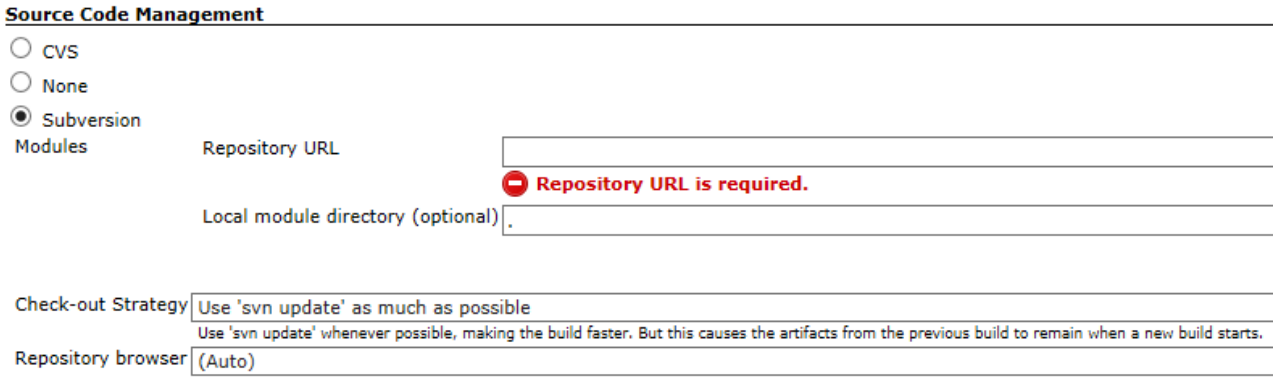
- Monitors version control system, and checks out the latest changes as they occur.
- Compiles and tests the most recent version of the code.
- Simply check out and build the latest version of the source code on a regular basis.
- SCM configuration options in Jenkins are identical across all sorts of build jobs.

- Jenkins supports CVS and Subversion out of the box, with built-in support for Git
- Integrates with a large number of other version control systems via plugins.

1.17 Working with Subversion

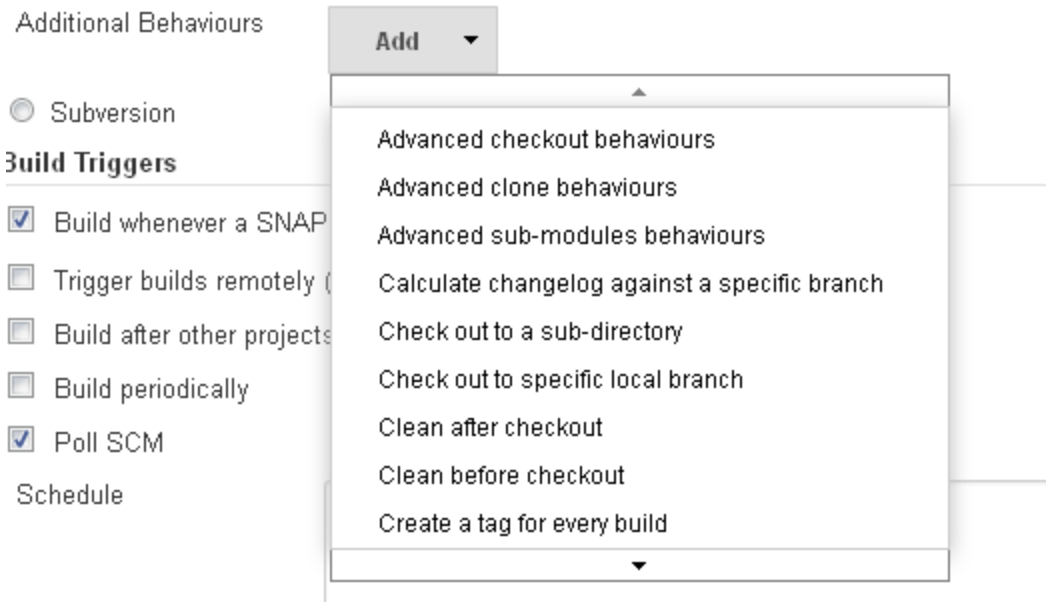
- Simply provide the corresponding Subversion URL
    - Supported protocols http, svn, or file.
- Jenkins will check that the URL is valid as soon as you enter it.
- If authentication needed, Jenkins will prompt you for the corresponding credentials automatically, and store them for any other build jobs that access this repository.
- Fine-tune Jenkins to obtain the latest source code from your Subversion repository by selecting an appropriate value in the Check-out Strategy drop-down list.
- Choose check-out Strategy as “Use ‘svn update’ as much as possible, with ‘svn revert’ before update”
    - No local files are modified, though it will not remove any new files that have been created during the build process.
    - You might want other options, depending on the load on your svn server.

1.18 Working with Subversion (cont'd)



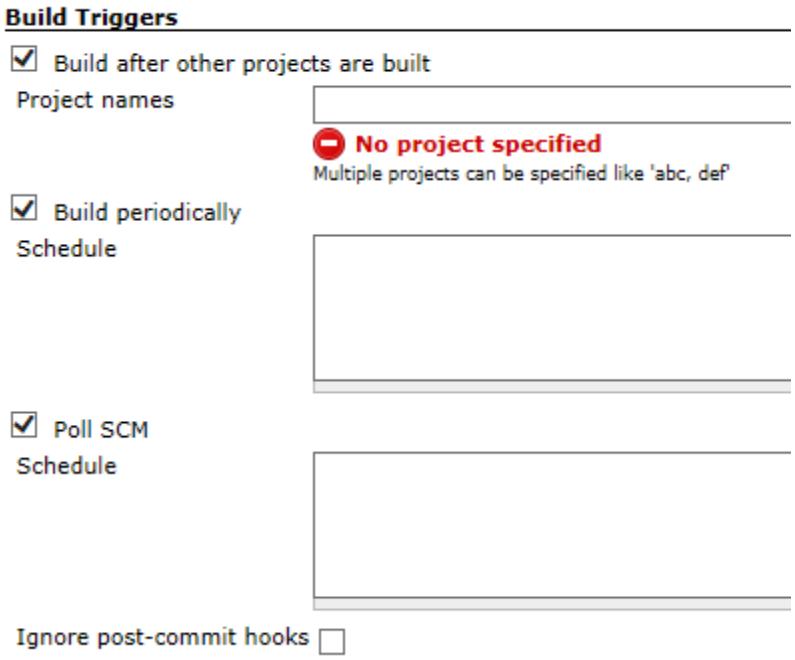
1.19 Working with Git

- Oddly enough, Git support isn't enabled by default
- Enable the 'Git Plugin' through the plugin management screen
- Any type of remote repository can be used:
  - Could be https, ssh, or local
- Jenkins will check that the URL is valid as soon as you enter it.
- You can store ssh credentials or http credentials
- There are a wide variety of other "Additional Checkout Behaviors" that are possible



1.20 Build Triggers

- In a Freestyle build, there are three basic ways a build job can be triggered
  - Start a build job once another build job has completed
  - Kick off builds at periodical intervals
  - Poll the SCM for changes



1.21 Schedule Build Jobs

- Build job at regular intervals.
- For all scheduling tasks, Jenkins uses a cron-style syntax, consisting of five fields separated by white space in the following format:
  - MINUTE : Minutes within the hour (0–59)
  - HOUR : The hour of the day (0–23) DOM
  - DOM : The day of the month (1–31)
  - MONTH : The month (1–12)
  - DOW : The day of the week (0–7) where 0 and 7 are Sunday.
- There are also a few short-cuts:

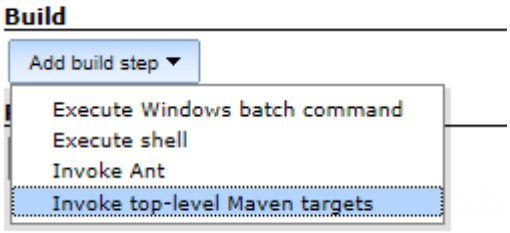
- “*” represents all possible values for a field. For example, “* * * * *” means “once a minute.”
- You can define ranges using the “M–N” notation. For example “1-5” in the DOW field would mean “Monday to Friday.”
- You can use the slash notation to defined skips through a range. For example, “*/5” in the MINUTE field would mean “every five minutes.”
- A comma-separated list indicates a list of valid values. For example, “15,45” in the MINUTE field would mean “at 15 and 45 minutes past every hour.”

1.22 Polling the SCM

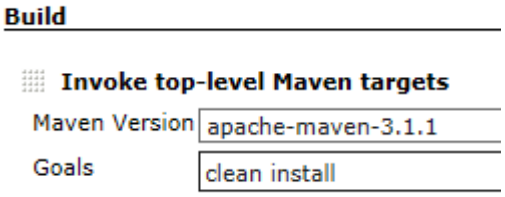
- Poll SVN server at regular intervals if any changes have been committed.
- Jenkins kicks off a build, source code in the project, .
- Polling frequently to ensure that a build kicks off rapidly after changes have been committed.
- The more frequent the polling is, the faster the build jobs will start, and the more accurate.
- In Jenkins, SCM polling is easy to configure, and uses the same cron syntax we discussed previously.

1.23 Maven Build Steps

- Jenkins has excellent Maven support, and Maven build steps are easy to configure and very flexible.
- Select “Invoke top-level Maven targets” from the build step lists.



- Select a version of Maven to run (if you have multiple versions installed)
- Enter the Maven goals you want to run. Jenkins freestyle build jobs work fine with both Maven 2 and Maven 3.
- The optional POM field lets you override the default location of the Maven pom.xml file.



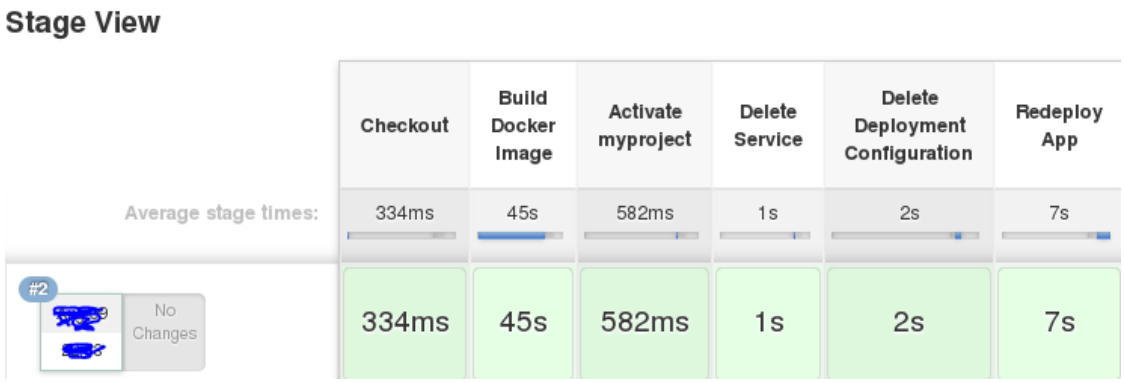
1.24 Jenkins / OpenShift Pipeline

- Same Pipeline code

```
node {
 stage 'Checkout'
 git url: '/var/lib/jenkins/repos/hello-node'

 stage 'Build Docker Image'
 sh 'docker build -t node-app:v1.0 .'

 stage 'Activate myproject'
 sh 'oc project myproject'
```

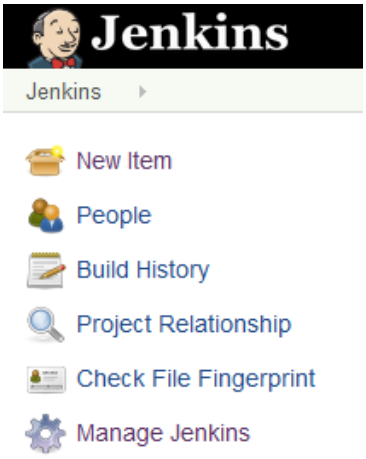
```
    stage 'Delete Service'
    sh 'oc delete svc/node-app'

    stage 'Delete Deployment Configuration'
    sh 'oc delete dc/node-app'

    stage 'Redeploy App'
    sh 'oc new-app --docker-image="node-app:v1.0"'
}
```

1.25 Jenkins / OpenShift Pipeline Output

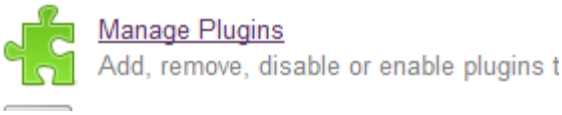


1.26 Installing Jenkins Plugins

- For example, installing the Jenkins Emma plugin for code coverage.



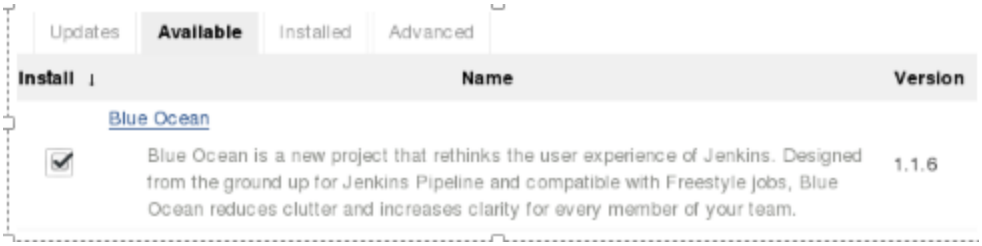
- Click **Manage Jenkins**
- Then on click the **Manage Plugins** link:



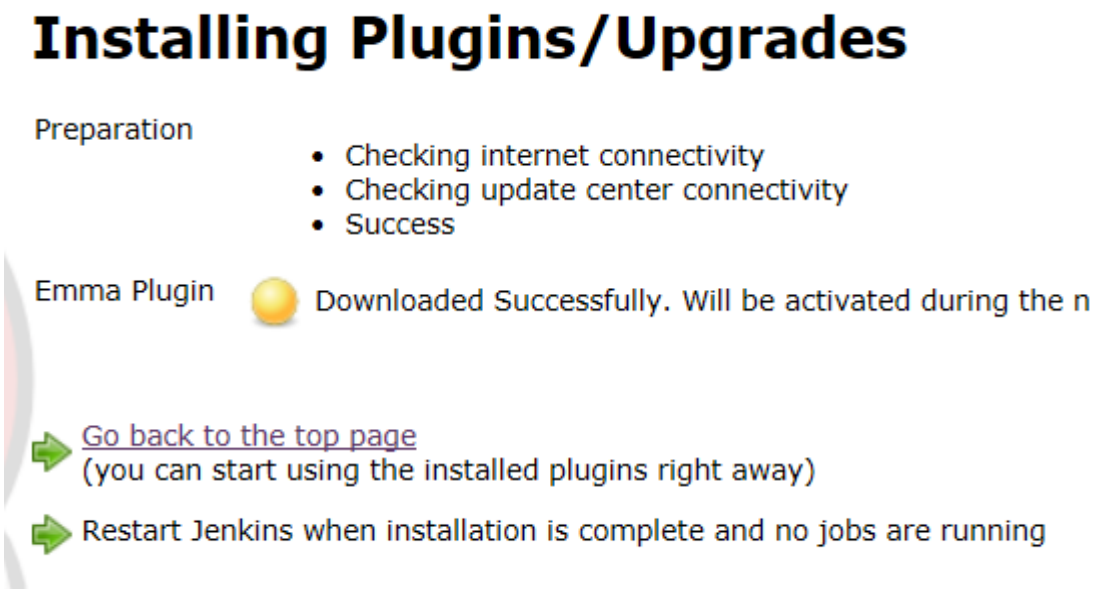
- Select the Available tab:

- For quick access, enter plugin name on the filter input box, located at right top corner.

- Scroll down the list of plugins to find the Blue Ocean Plugin.
- Select the check box next to Emma Plugin.
- Click **Download now and install after restart**



This will download the Blue Ocean plugin for Jenkins. The download may take some time to finish. Once the installation is complete, you will get the following:



- Select the Restart Jenkins when installation is complete and no jobs are running.
- You have done the installation
- Note that quite often, the "Install without restart" option works too

1.27 The Blue Ocean Plugin

- Blue Ocean is a project that rethinks the user experience of Jenkins

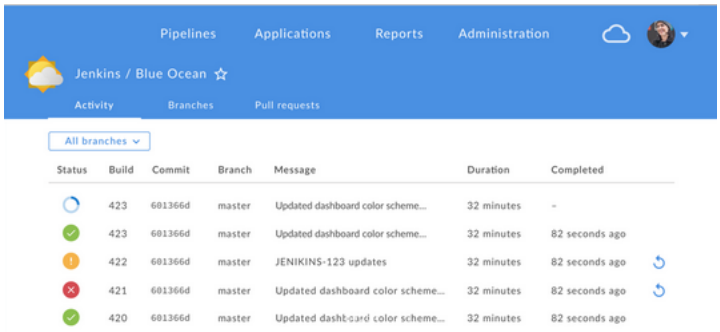
- It models and presents the process of software delivery by surfacing information that's important to development teams with as few clicks as possible
- It rethinks Jenkins while still staying true to the extensibility that is core to Jenkins.
- Jenkins users can install Blue Ocean side-by-side with the Jenkins Classic UI via a plugin

1.28 Blue Ocean Plugin Features

- New modern user experience
- Advanced Pipeline visualizations with built-in failure diagnosis
- Branch and Pull Request awareness
- Personalized View

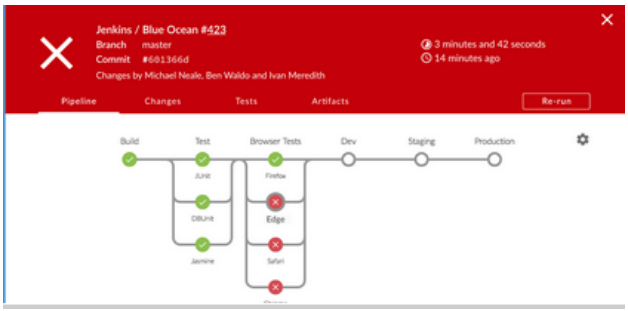
1.29 New modern user experience

- The UI aims to improve clarity, reduce clutter and navigational depth to make the user experience very concise.
- A modern visual design gives developers much needed relief throughout their daily usage and screens respond instantly to changes on the server without requiring manual page refreshes



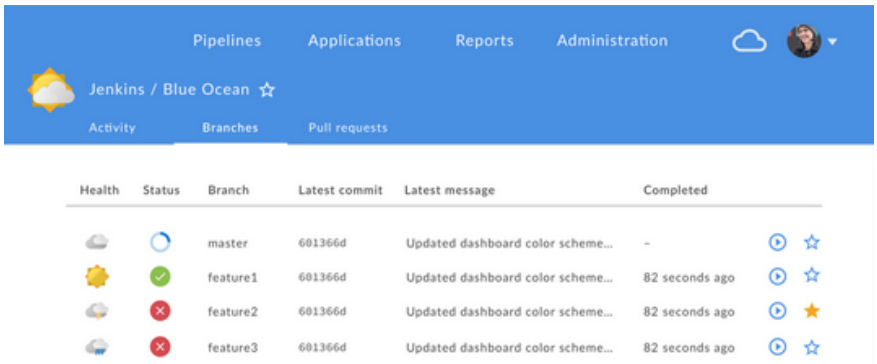
1.30 Advanced Pipeline visualizations with built-in failure diagnosis

- Pipelines are visualized on screen along with the steps and logs to allow simplified comprehension of the continuous delivery pipeline – from the simple to the most sophisticated scenarios.



1.31 Branch and Pull Request awareness

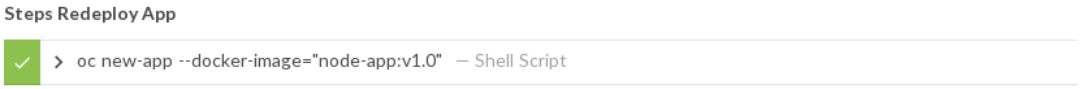
- Modern pipelines make use of multiple Git branches, and Blue Ocean is designed with this in mind.
- When Jenkinsfile defining a pipeline is dropped into a Git repository, it automatically discovers and starts automating any branches and validating pull requests.



1.32 Personalized View

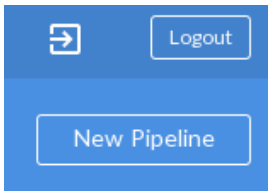
- Blue Ocean offers a dashboard.
- Jobs that need user attention, such as Pipeline awaiting approval or a failing job, appear on the top of the dashboard.

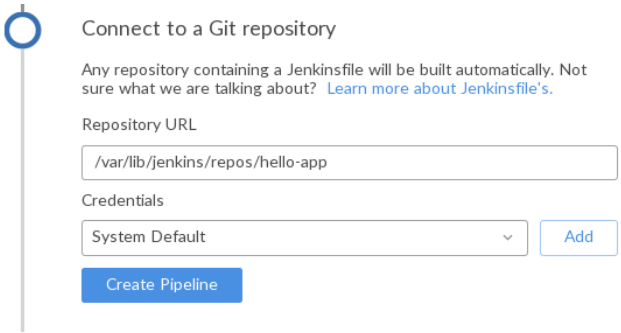
1.33 OpenShift Pipeline Output

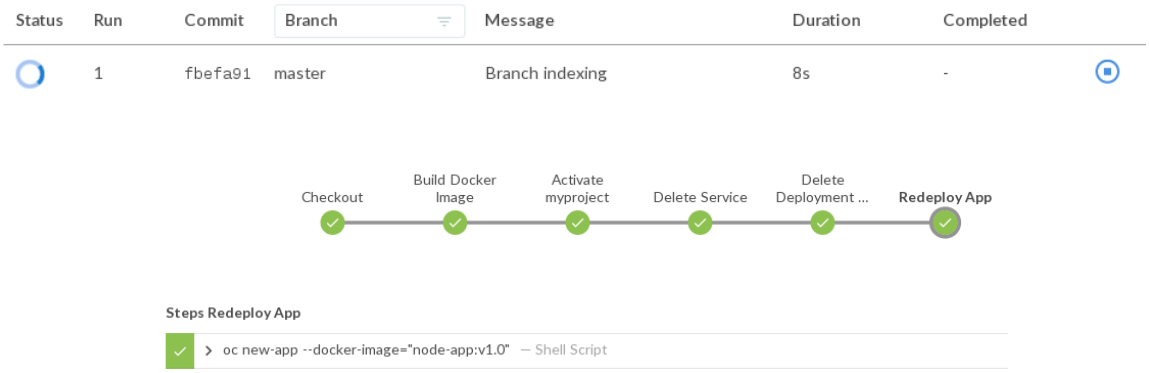


1.34 Creating OpenShift Blue Ocean Pipeline

- You can also create an OpenShift Blue Ocean pipeline from scratch







1.35 Summary

- OpenShift is an open-source container orchestration software
- OpenShift can be combined with Jenkins and Blue Ocean plugin to implement CI / CD.
- Blue Ocean plugin revamps Jenkins UI.