
Implementing CI/CD



olsen software

Overview

- Jenkins is a build automation tool, widely used in Citi
 - You can create a Jenkins "build" to automate s/w builds
 - Helps you do Continuous Integration / Continuous Delivery

Contents

- Creating a Jenkins Freestyle project
- Using Jenkins with OpenShift

The Environment

- We've created a separate Linux VM for each student, with Jenkins and OpenShift installed
 - Start your Linux VM via <http://student.conygre.com>

<https://student.conygre.com>

Student Machine Manager

Find a Machine Connection Instructions

Using this application you can turn your classroom machine on and off and also

Step 1: Find your machine

Region:

Machine name:

Domain:

Where is your virtual machine? Your instructor should have told you. If you are unsure, select **Don't Know**.

Enter the first part of the name of the machine you have been using for the training. So if your machine is **training1.conygre.com**, enter **training1**

The top level domain for your machine.

[Find my Machine](#)

Step 2: Turn on Your Machine

We have found your machine! If you plan to use it now, you can turn it on. Use the refresh button once you have turned it on to obtain the connection instructions.

Machine Name/IP	Status	Actions	Available hours (UTC)	Refresh
singaporedevops1.conygre.com 100.25.185.16	● running	Stop	From -1 to 16	Refresh Status

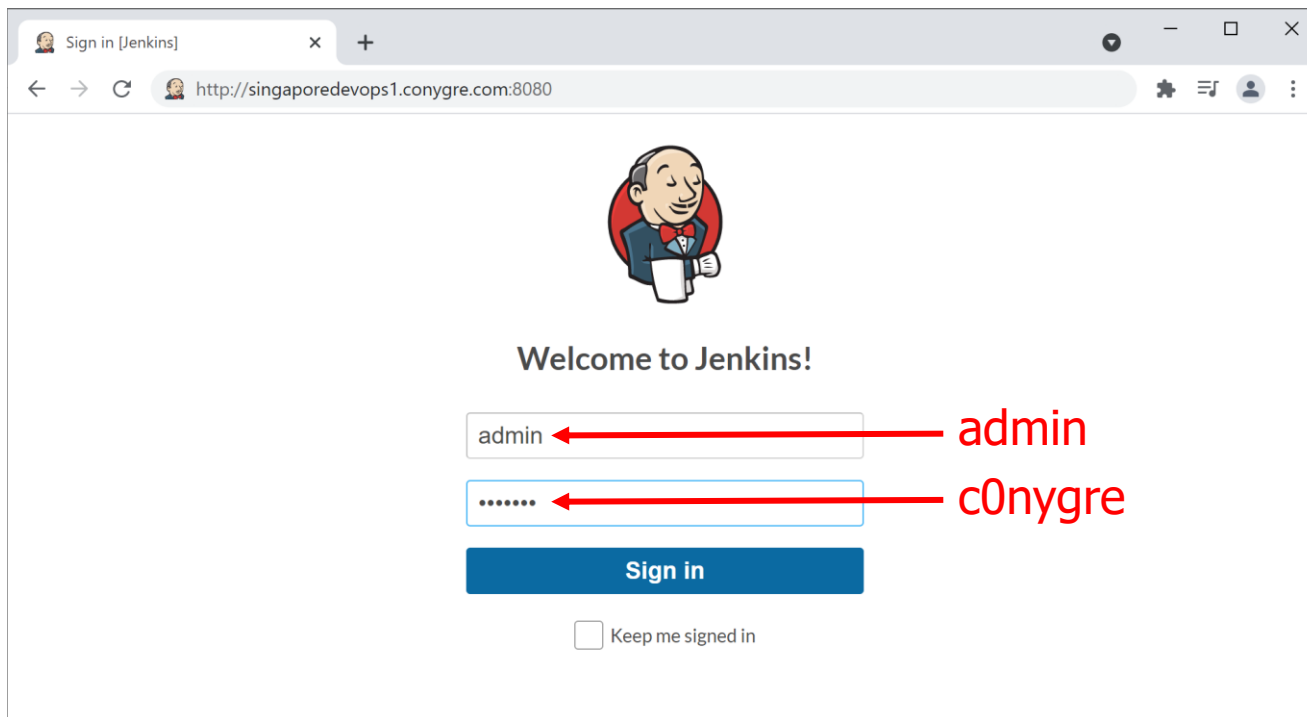
Machine names are like
linuxops-dallas1.conygre.com

Creating a Jenkins Freestyle Project

- Displaying the Jenkins Dashboard
- Creating a Freestyle Project
- Running the build
- Viewing the build log

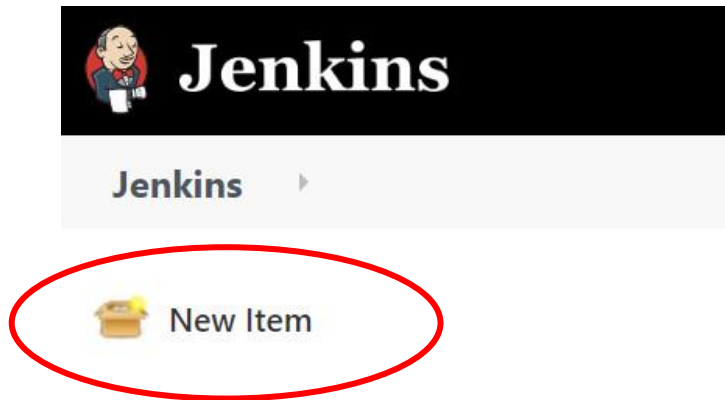
Displaying the Jenkins Dashboard

- We're going to see how to create a Freestyle project in Jenkins on the Linux VM
 - The first step is to open the Jenkins Dashboard on the Linux VM
 - To do this, browse to <http://linuxops-dallasX.conygre.com:8080>



Creating a Freestyle project

- In the Jenkins Dashboard click **New Item**



Creating a Freestyle Project

- On the next page give the project a name
- Select **Freestyle project** and click **OK**

Enter an item name

» Required field



Freestyle 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.



Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



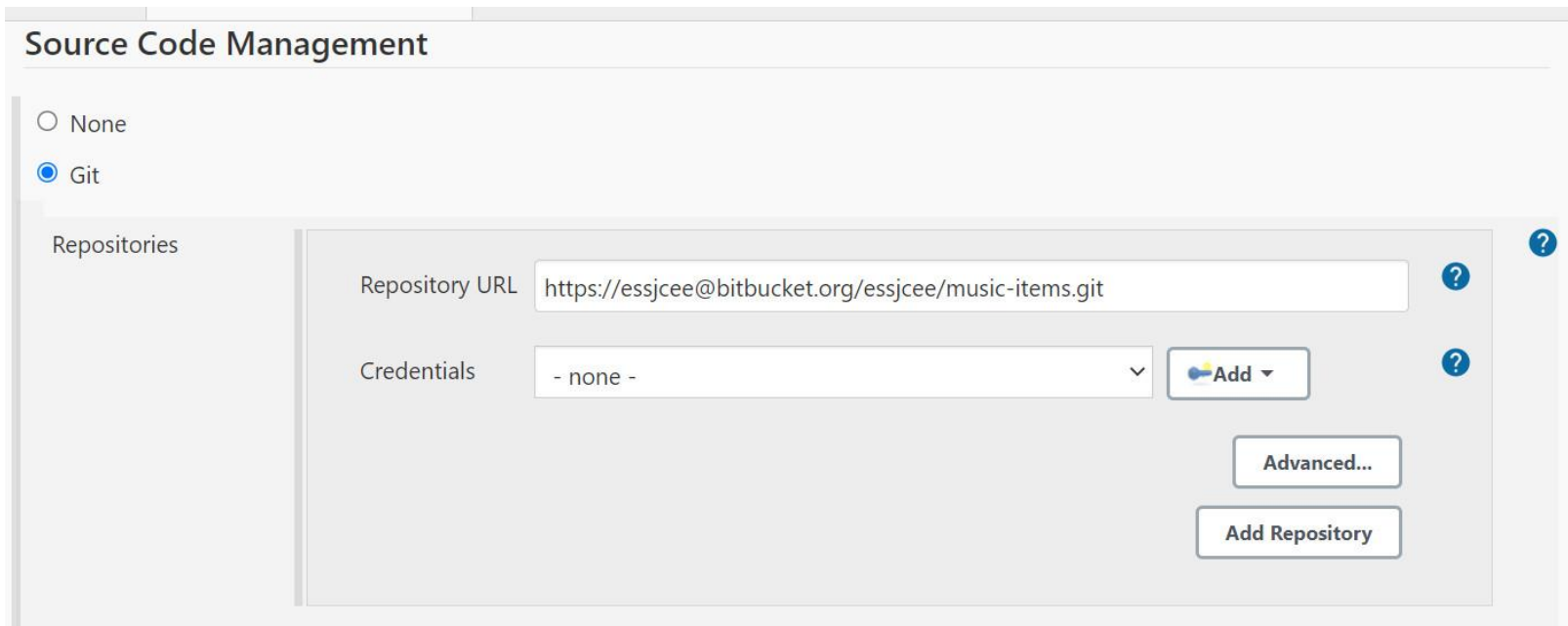
Multi-configuration project

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

OK

Creating a Freestyle Project

- On the next page under **Source Code Management** select **Git** and enter this repository url:
 - <https://essjcee@bitbucket.org/essjcee/music-items.git>
 - You don't need any credentials
 - After doing this scroll down to the **Build** section



The screenshot shows the 'Source Code Management' configuration window. On the left, under 'Repositories', the 'Git' radio button is selected. The main area contains a 'Repository URL' text field with the value 'https://essjcee@bitbucket.org/essjcee/music-items.git' and a 'Credentials' dropdown menu set to '- none -'. There are two question mark icons to the right of these fields. At the bottom right, there are three buttons: 'Add' (with a plus icon), 'Advanced...', and 'Add Repository'.

Source Code Management

☐ None

☒ Git

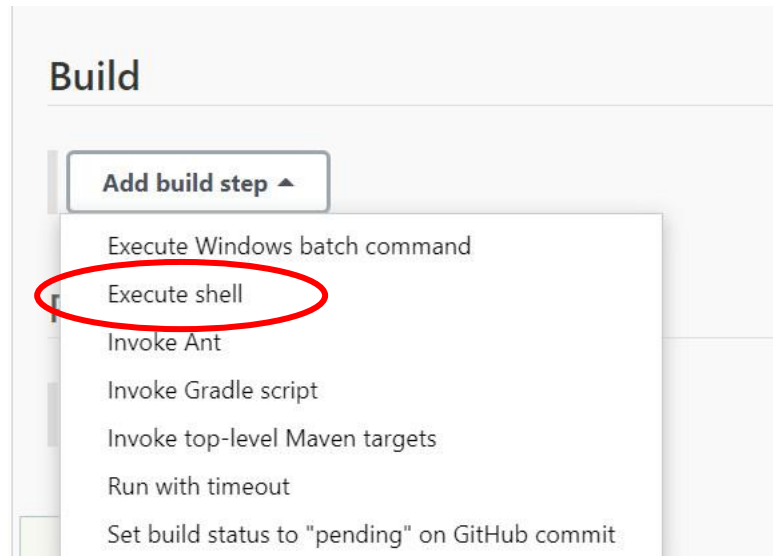
Repositories

Repository URL

Credentials

Creating a Freestyle Project

- In the **Build** section select **Execute shell** from the **Add builds step** dropdown



Creating a Freestyle Project

- Paste the provided script into the **Execute shell Command** text area and click **Save**

Build

Execute shell

Command

```
docker stop musicapp || echo 'App not running'
docker container rm musicapp || echo 'No container to remove'
docker image rm music/app:latest || echo 'No image to remove'
chmod a+x mvnw
./mvnw package
docker build -t music/app .
docker run -d -p 8081:8080 --name musicapp music/app
```

[See the list of available environment variables](#)

Add build step ▾

Post-build Actions

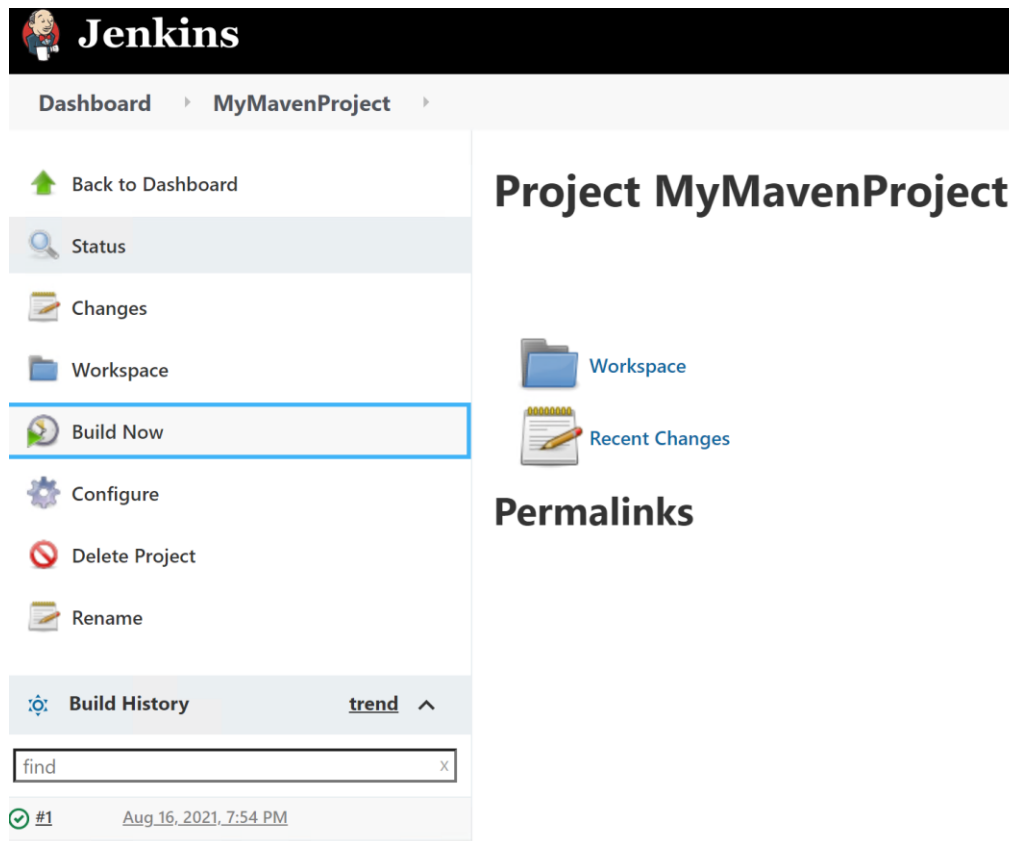
Add post-build action ▾

Save

Apply

Running the Build

- In the Project page click **Build Now**
- The build indicator will go blue or red, indicating whether the build succeeded or failed



Viewing the Build Log

- To view the build log to see what happened:
 - Click the #1 link in the Build History section
 - In the next screen, click Console Output

The screenshot shows the Jenkins web interface in a browser. The address bar indicates the URL is `localhost:8081/job/MyMavenProject/1/console`. The Jenkins logo and name are in the top left, and a search bar and user profile (Andy Olsen) are in the top right. The main content area is titled "Console Output" with a green checkmark icon. On the left sidebar, the "Console Output" link is highlighted. The console log shows the following output:

```
Skipping 444 KB. Full Log
19:57:04.953 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default resource location for test class
[demo.restservices.DemoRestServicesApplicationTests]: class path resource [demo/restservices/DemoRestServicesApplicationTestsContext.groovy] does not
exist
19:57:04.953 [main] INFO org.springframework.test.context.support.AbstractContextLoader - Could not detect default resource locations for test class
[demo.restservices.DemoRestServicesApplicationTests]: no resource found for suffixes {-context.xml, Context.groovy}.
19:57:04.955 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils - Could not detect default configuration classes
for test class [demo.restservices.DemoRestServicesApplicationTests]: DemoRestServicesApplicationTests does not declare any static, non-private, non-
final, nested classes annotated with @Configuration.
19:57:05.046 [main] DEBUG org.springframework.test.context.support.ActiveProfilesUtils - Could not find an 'annotation declaring class' for annotation
type [org.springframework.test.context.ActiveProfiles] and class [demo.restservices.DemoRestServicesApplicationTests]
19:57:05.261 [main] DEBUG org.springframework.context.annotation.ClassPathScanningCandidateComponentProvider - Identified candidate component class:
file [C:\Andy\sprint3\Spring Boot\SpringBootDev\Demos\demo-rest-services\target\classes\demo\restservices\Application.class]
19:57:05.340 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Found @SpringBootConfiguration
demo.restservices.Application for test class demo.restservices.DemoRestServicesApplicationTests
19:57:05.481 [main] DEBUG org.springframework.boot.test.context.SpringBootTestContextBootstrapper - @TestExecutionListeners is not present for class
[demo.restservices.DemoRestServicesApplicationTests]: using defaults.
19:57:05.482 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper - Loaded default TestExecutionListener class names from
location [META-INF/spring.factories]: [org.springframework.boot.test.mock.mockito.MockitoTestExecutionListener,
org.springframework.boot.test.mock.mockito.ResetMocksTestExecutionListener,
org.springframework.boot.test.autoconfigure.restdocs.RestDocsTestExecutionListener,
org.springframework.boot.test.autoconfigure.web.client.MockRestServiceServerResetTestExecutionListener,
org.springframework.boot.test.autoconfigure.web.servlet.MockMvcPrintOnlyOnFailureTestExecutionListener,
```

View the deployed containerized Spring Boot Application

- Browse to port 8081 on your VM to see the deployed Spring Boot Application

← → ↻ 🏠 ⚠ Not secure | jcdemoopenshift1.conygre.com:8081

Music Collection

ID

Name

Artist or Group

Genre

Id	Name	Artist or Group	Genre	Operation
1	Imagine	John Lennon	Rock	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
2	Ghost Town	The Specials	Ska	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

Using Jenkins with OpenShift

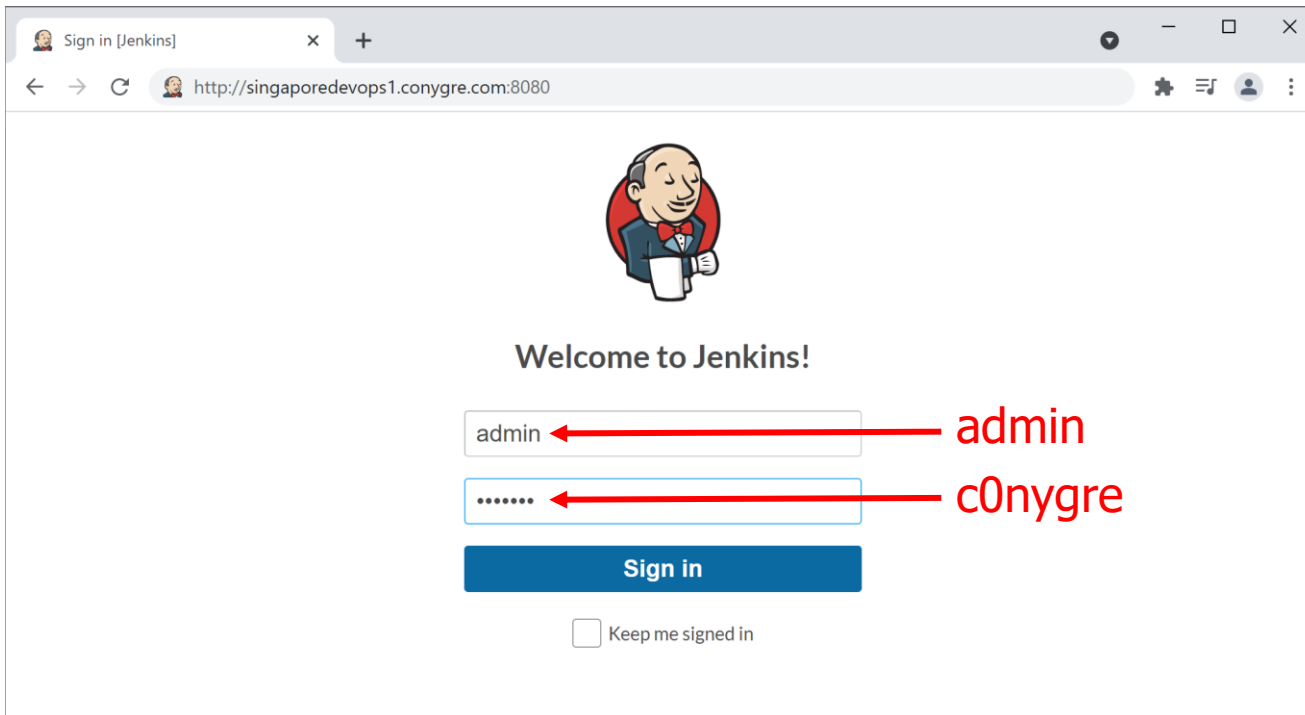
- Overview
- The environment
- Connecting to the Linux VM
- Displaying the Jenkins Dashboard
- Creating a pipeline build
- Running a build
- Verifying the deployed app in OpenShift
- Pinging the app via the exposed Route

Overview

- In this section we'll show how to use Jenkins and OpenShift together
- We'll show how to write a Jenkinsfile (Jenkins build script) that defines a pipeline as follows:
 1. Build a Docker image for our app
 2. Deploy and run the containerized app in OpenShift

Displaying the Jenkins Dashboard

- We're going to see how to create a "build pipeline" in Jenkins on the Linux VM
 - The first step is to open the Jenkins Dashboard on the Linux VM
 - To do this, browse to <http://linuxops-dallasX.conygre.com:8080>



A screenshot of a web browser displaying the Jenkins login page. The browser's address bar shows the URL `http://singaporedevops1.conygre.com:8080`. The page features the Jenkins logo (a cartoon man in a suit) and the text "Welcome to Jenkins!". Below this, there are two input fields: the first contains the text "admin" and the second contains a masked password ".....". Red arrows point from the labels "admin" and "c0nygre" to their respective input fields. A blue "Sign in" button is positioned below the password field. At the bottom, there is a checkbox labeled "Keep me signed in".

Sign in [Jenkins]

http://singaporedevops1.conygre.com:8080

Welcome to Jenkins!

admin

.....

Sign in

☐ Keep me signed in

admin

c0nygre

Creating a Pipeline Build (1 of 3)

- Create a new "pipeline build" as follows:
 - In the Jenkins Dashboard, click **New Item**
 - Enter a project name, click **Pipeline**, then click **OK**

The image consists of two screenshots of the Jenkins web interface. The left screenshot shows the Jenkins Dashboard with the 'New Item' button highlighted in a red box. The right screenshot shows the 'New Item' form with the project name 'MyProductsProject' entered and the 'Pipeline' option selected, both highlighted in red boxes.

Left Screenshot: Jenkins Dashboard

- Browser: Dashboard [Jenkins] | http://singaporedevops1.conygre.com:8080
- Header: Jenkins
- Left Sidebar: New Item (highlighted), People, Build History, Project Relationship, Check File Fingerprint, Manage Jenkins, My Views, Lockable Resources, New View.
- Right Panel: All, S, W, Name | BasicSpringD, demo-project, Icon: S M L, Legend.
- Bottom: Build Queue (No builds in the queue.)

Right Screenshot: New Item [Jenkins]

- Browser: New Item [Jenkins] | http://singaporedevops1.conygre.com:8080/view/all/newJob
- Header: Jenkins | search | Administrator | log out
- Form: Enter an item name (MyProductsProject - highlighted), Freestyle project, Pipeline (highlighted), Multi-configuration project, Folder, GitHub Organization, Multibranch Pipeline.
- Bottom: OK button, type to autocomplete.

Creating a Pipeline Build (2 of 3)

- Scroll down to **Pipeline**, then enter **Destination** info:

The screenshot shows the Jenkins configuration page for 'MyProductsProject' with the 'Pipeline' tab selected. The 'Definition' is set to 'Pipeline script from SCM'. The 'SCM' is set to 'Git'. The 'Repository URL' is set to 'https://olsensoft@bitbucket.org/olsensoft/cicd-products-:'. The 'Credentials' are set to 'none'. The 'Branches to build' section has a 'Branch Specifier (blank for 'any')' set to '*/master'. The 'Repository browser' is set to '(Auto)'. The 'Script Path' is set to 'Jenkinsfile'. The 'Lightweight checkout' checkbox is checked.

Annotations on the screenshot:

- A red box labeled 'Pipeline script from SCM' points to the 'Definition' dropdown.
- A red box labeled 'Git' points to the 'SCM' dropdown.
- A red box containing the URL <https://essjcee@bitbucket.org/essjcee/music-items.git> has an arrow pointing to the 'Repository URL' field.
- A red box labeled 'master' has an arrow pointing to the 'Branch Specifier' field.

Creating a Pipeline Build (3 of 3)

- Also enter the following additional details:

The screenshot shows the Jenkins configuration page for 'MyProductsProject' in the 'Pipeline' tab. The page is divided into several sections: 'Credentials' (set to '- none -'), 'Branches to build' (with a 'Branch Specifier' of '*/master'), 'Repository browser' (set to '(Auto)'), 'Additional Behaviours' (set to 'Add'), 'Script Path' (set to 'Jenkinsfile'), 'Lightweight checkout' (unchecked), and 'Pipeline Syntax'. Red arrows and text annotations highlight key actions: 'Jenkinsfile' is highlighted in the 'Script Path' field; 'Deselect this option' points to the 'Lightweight checkout' checkbox; and 'Click Save' points to the 'Save' button at the bottom left. The browser address bar shows 'http://singaporedevops1.conygre.com:8080/job/MyProductsProject/configure'.

MyProductsProject Config [Jenkins] x +

Not secure | http://singaporedevops1.conygre.com:8080/job/MyProductsProject/configure

Jenkins > MyProductsProject >

General Build Triggers Advanced Project Options **Pipeline**

Credentials - none - Add Advanced... Add Repository

Branches to build

Branch Specifier (blank for 'any') */master Add Branch

Repository browser (Auto)

Additional Behaviours Add

Script Path Jenkinsfile

Lightweight checkout ☐

Pipeline Syntax

Save Apply

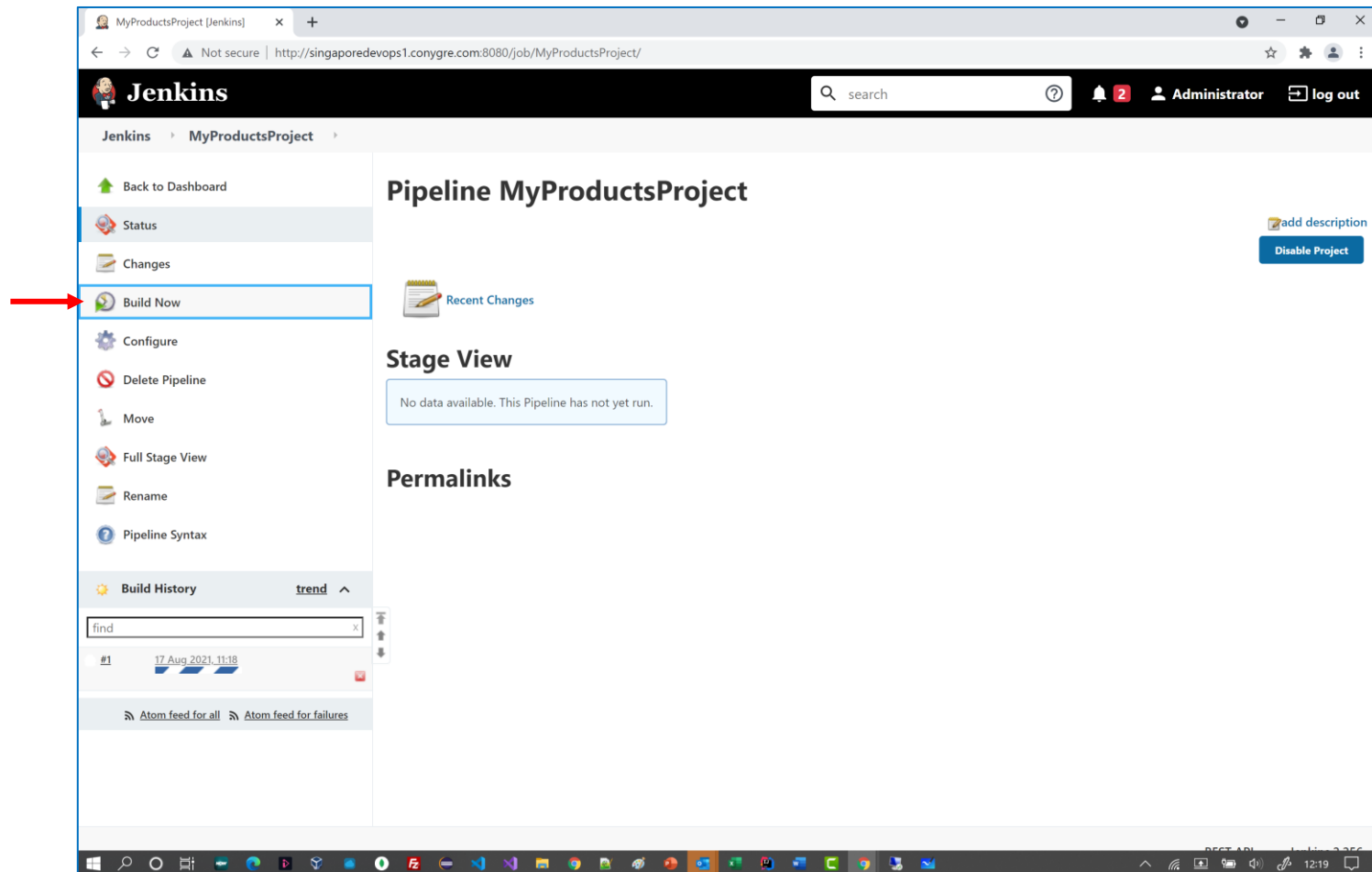
Jenkinsfile

Deselect this option

Click Save

Running a Build (1 of 2)

- Back in the Dashboard it shows your pipeline
 - In the menu on the left, click **Build Now**



Running a Build (2 of 2)

- You'll see a series of green squares
 - Indicating it has completed all the stages of the Jenkins build

The screenshot shows the Jenkins web interface for a pipeline named 'MyProductsProject'. The left sidebar contains navigation links: Back to Dashboard, Status, Changes, Build Now, Configure, Delete Pipeline, Move, Full Stage View, Rename, Pipeline Syntax, Build History, and Atom feed for all/Atom feed for failures. The main content area displays the 'Pipeline MyProductsProject' view. It includes a 'Recent Changes' section, a 'Stage View' table, and a 'Permalinks' section.

Pipeline MyProductsProject

add description
Disable Project

Recent Changes

Stage View

	Declarative: Checkout SCM	Build docker image	Deploy container To OpenShift
Average stage times: (Average full run time: ~30s)	3s	11s	7s
#1 Aug 17 12:18 No Changes	3s	11s	7s

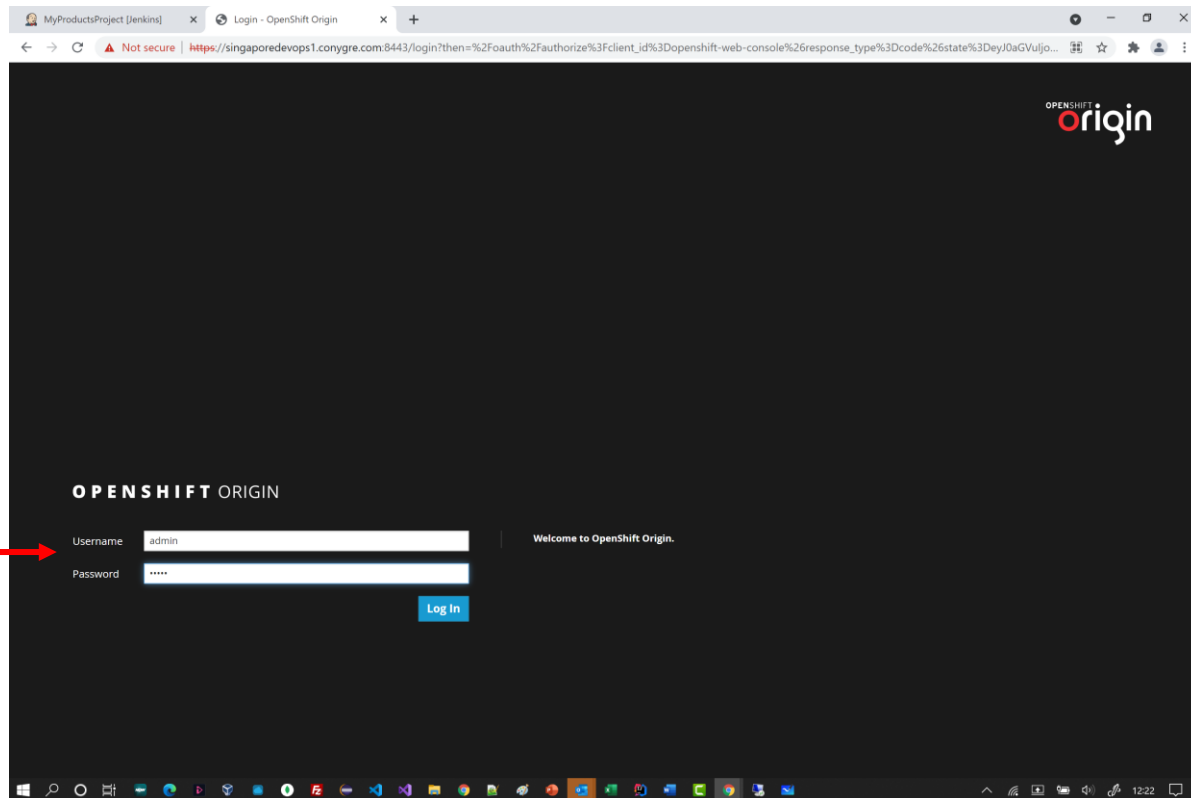
Permalinks

- Last build (#1), 3 min 52 sec ago
- Last stable build (#1), 3 min 52 sec ago
- Last successful build (#1), 3 min 52 sec ago
- Last completed build (#1), 3 min 52 sec ago

Verifying the Deployed App in OpenShift (1 of 3)

- The Jenkins build has deployed the app to OpenShift
- To verify this, browse to the OpenShift console as follows:
 - <https://linuxops-dallasX.conygre.com:8443/>

admin
admin



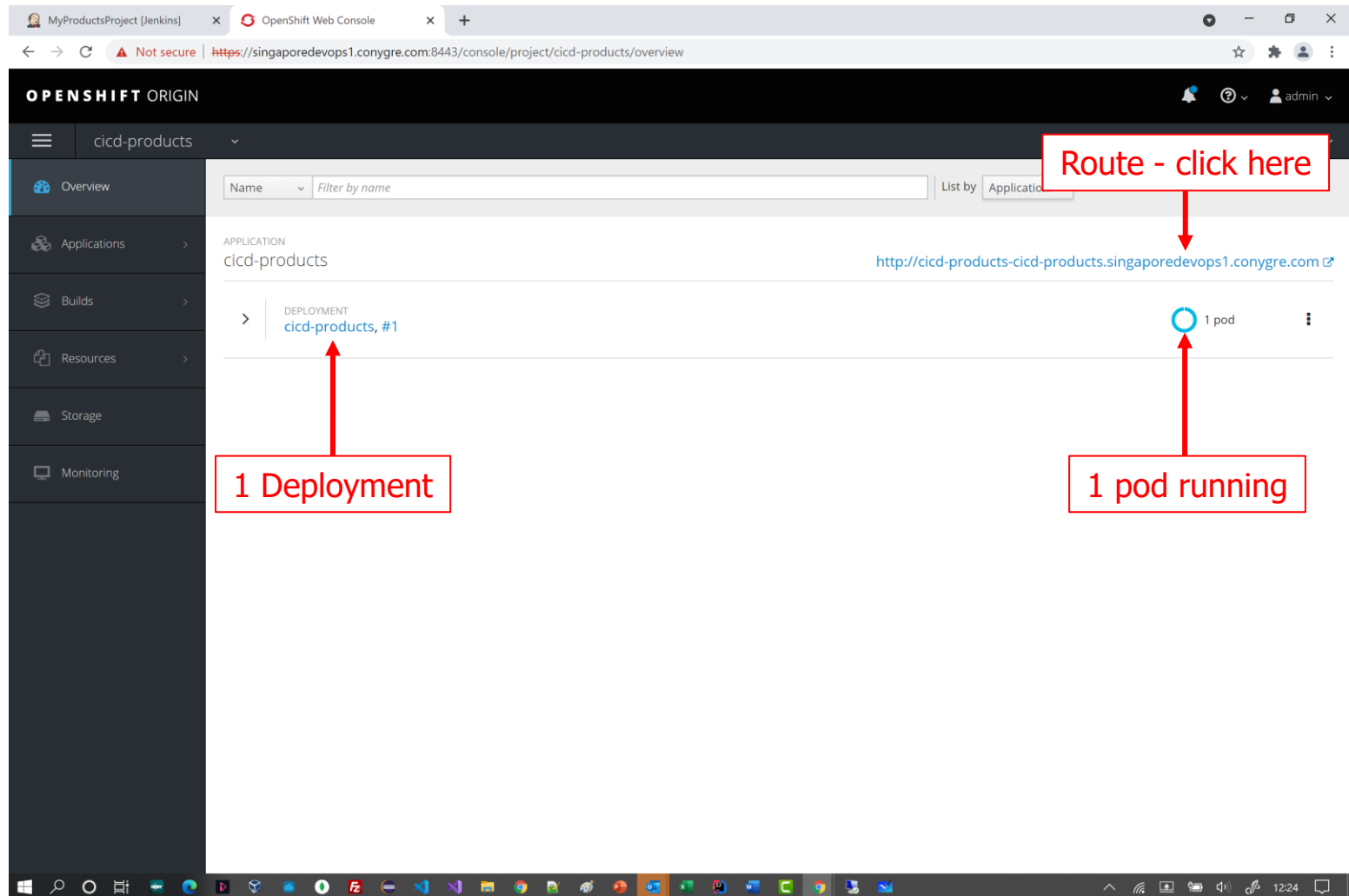
Verifying the Deployed App in OpenShift (2 of 3)

- On the right, you'll see your project in the list of projects

The screenshot displays the OpenShift Web Console interface. On the right sidebar, under 'My Projects', there is a list of projects. The project 'cicd-products' is highlighted with a red box, indicating it is the selected project. Below this list, a red arrow points from a text box that says 'Click your project to see the details' to the 'cicd-products' entry. The main area of the console shows the 'Browse Catalog' with various application templates available for deployment, such as Apache HTTP Server, CakePHP + MySQL, Django + PostgreSQL, Jenkins, etc.

Verifying the Deployed App in OpenShift (3 of 3)

- Your project appears as follows



Pinging the App via the Exposed Route

- When you click the Route that exposes the app, you've just pinged your running application - success!

Music Collection

ID

Name

Artist or Group

Genre

Id	Name	Artist or Group	Genre	Operation
1	Imagine	John Lennon	Rock	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
2	Ghost Town	The Specials	Ska	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

Any Questions?

