



ORACLE
OPEN
WORLD

Modern Business in the Cloud

DevOps in the Cloud with Oracle
Developer Cloud Service

Murtaza Amiji
Oracle Product Management

Abhinav Shroff
Oracle Product Management

Stephane Moriceau
Oracle APAC Business Development

ORACLE

Oracle Cloud Infrastructure DevOps Automation

In this lab, you'll learn how to build a Docker image for a Node.js REST service on Oracle Developer Cloud Service (DevCS), and push it to Docker registries such as Oracle Cloud Infrastructure Registry and DockerHub.

Why Oracle Developer Cloud Service?

In recent years, the world of application development has adopted new methodologies that aim to improve the quality and speed with which applications are being delivered. The introduction of innovative development approaches, such as test-driven development and Agile development, gave rise to a set of new techniques and tools that enable those methodologies. Tools such as automatic build utilities combined with continuous integration platforms, as well as enhanced collaborative tools such as wikis and code review utilities, simplify the adoption of these new methodologies.

However, for many IT shops, setting up these environments and maintaining them was difficult and cost prohibitive, so many organizations stuck to the old way of building applications. Oracle has changed this by introducing a simpler way to adopt modern development methodology and tools with a cloud-based offering known as Oracle Developer Cloud Service.

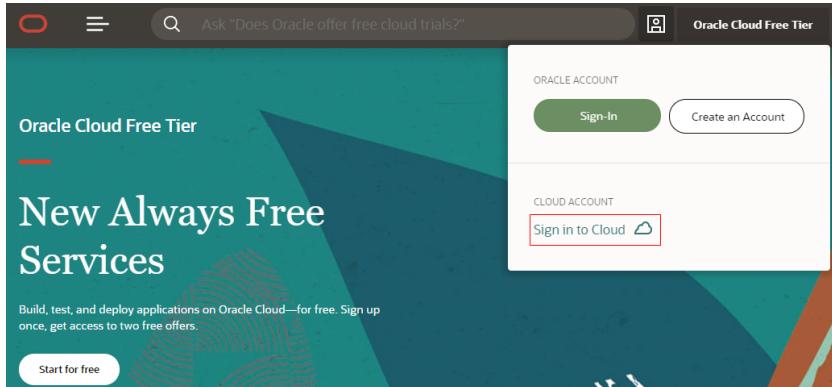
In this tutorial, you will work with DevCS and learn about some of its most important features. Specifically, you will:

- Create a DevCS *project*, which serves as the organizing principle for your software development efforts;
- Pull the code from an existing GitHub repository and use it to populate a DevCS-hosted Git repository;
- Create a task within DevCS and an Agile board to track it;
- Complete the task by branching, making changes to the code using the online editor, then merging the code;
- Build the Docker image with Continuous Integration in DevCS;
- Push the Docker image to Oracle Cloud Infrastructure Registry and DockerHub.

Getting Started –

Before we start with the lab, please follow the below instructions to setup your instance:

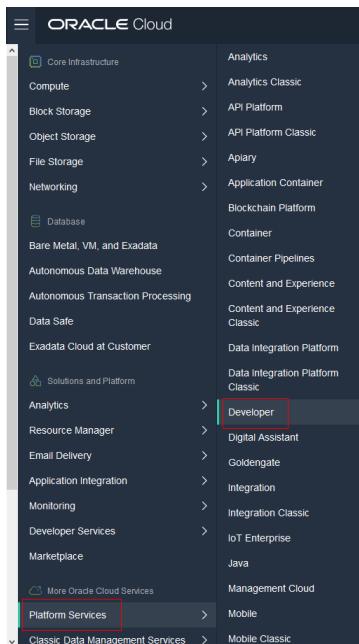
1. Request a free account



- Request a free Oracle Cloud account with credits from <https://www.oracle.com/cloud/>.
- See the [Overview of the Free Oracle Cloud Promotion](#) infographic for more information.

2. Create the DevCS service instance

- When you get a *Welcome to Oracle Cloud* email, sign into your Oracle Cloud account using the details provided.
- On the Oracle Cloud page, open the navigation menu. Under **More Oracle Cloud Services**, select **Platform Services**, and then select **Developer**.



- Click **Create Instance**.
- On the Create New Instance page, enter the required details and click **Next**.
- On the Service Details page, verify the entered details and click **Next**.
- On the Confirmation page, click **Create**.

3. Set up Compute and Storage

The OCI Compute services provide the virtual machines (VMs) on which DevCS runs its builds. The OCI Storage services provide storage for build and Maven artifacts.

- Follow this [link](#) to set up the compute service (Note: Ignore the “Set Up the OCI Classic Connection” section)

Now set up the storage service as follows:

- Open the navigation menu. Under **Core Infrastructure**, click **Object Storage**.
 - A list of the existing buckets in the compartment you're viewing is displayed.
- Select the compartment you created earlier from the **Compartment** list on the left side of the page.
 - A list of existing buckets is displayed.
- Click **Create Bucket**.
- In the **Create Bucket** dialog box, specify the attributes of the bucket:
 - **Bucket Name:** The system generates a default bucket name that reflects the current year, month, day, and time, for example **bucket-20190306-1359**.

(Note: If you change this default to any other bucket name, use letters, numbers, dashes, underscores, and periods. Do not include any confidential information.)

- **Storage Tier: Select the Standard option.**

(Note: Available tiers include:

- **Standard** is the primary, default Object Storage tier for storing frequently accessed data that requires fast and immediate access.
- **Archive** is a special tier for storing infrequently accessed data that requires long retention periods. Access to data in the **Archive** tier is not immediate. You must restore archived data before it's accessible. For more information, see [Overview of Archive Storage](#).)
- **Object Events: Select the default option.**

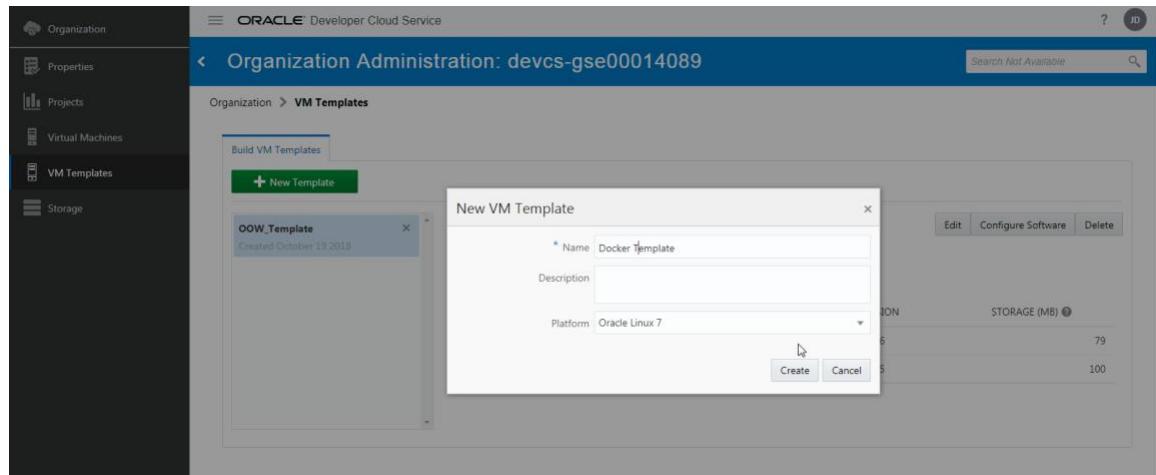
(Note: Select **Emit Object Events** if you want to enable the bucket to emit events for object state changes. For more information about events, see [Overview of Events](#).)

- **Encryption:** Buckets are encrypted with keys managed by Oracle by default, but you can optionally encrypt the data in this bucket using your own Key Management encryption key. **Select the default option.**

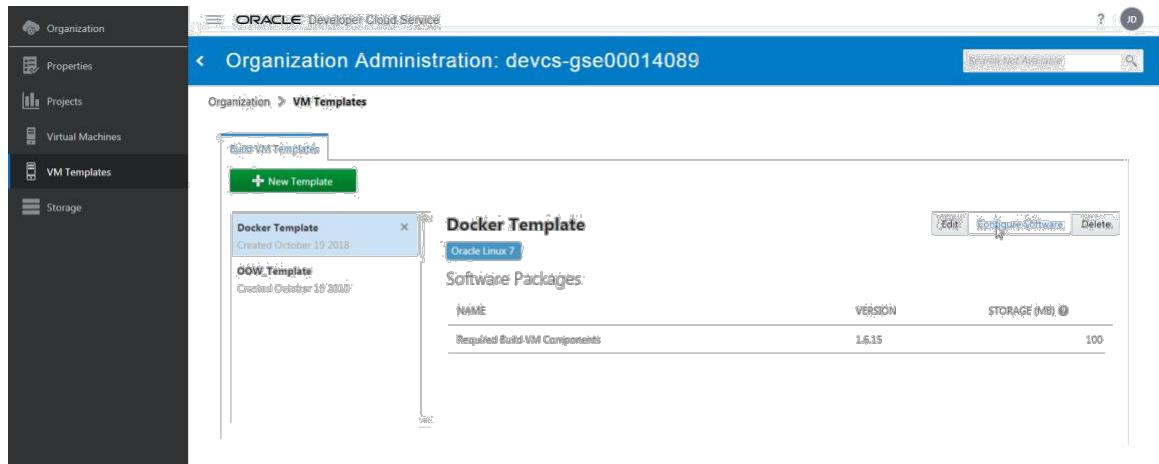
(Note: To use Key Management for your encryption needs, select **Encrypt Using Customer-Managed Keys**. Then, select the **Vault Compartment** and **Vault** that contain the master encryption key you want to use. Also select the **Master Encryption Key Compartment** and **Master Encryption Key**. For more information about encryption, see [Overview of Key Management](#). For details on how to create a vault, see [Managing Vaults](#).)
- **Tags:** Optionally, you can apply tags. If you have permissions to create a resource, you also have permissions to apply free-form tags to that resource. To apply a defined tag, you must have permissions to use the tag namespace. For more information about tagging, see [Resource Tags](#). If you are not sure if you should apply tags, skip this option (you can apply tags later) or ask your administrator.
- Click **Create Bucket**.
 - The bucket is created immediately and you can add objects to it. Objects added to archive buckets are immediately archived and must be restored before they are available for download.

4. For the builds to work in this lab, we will also have to configure Build Template and Build VM as per the below instructions:

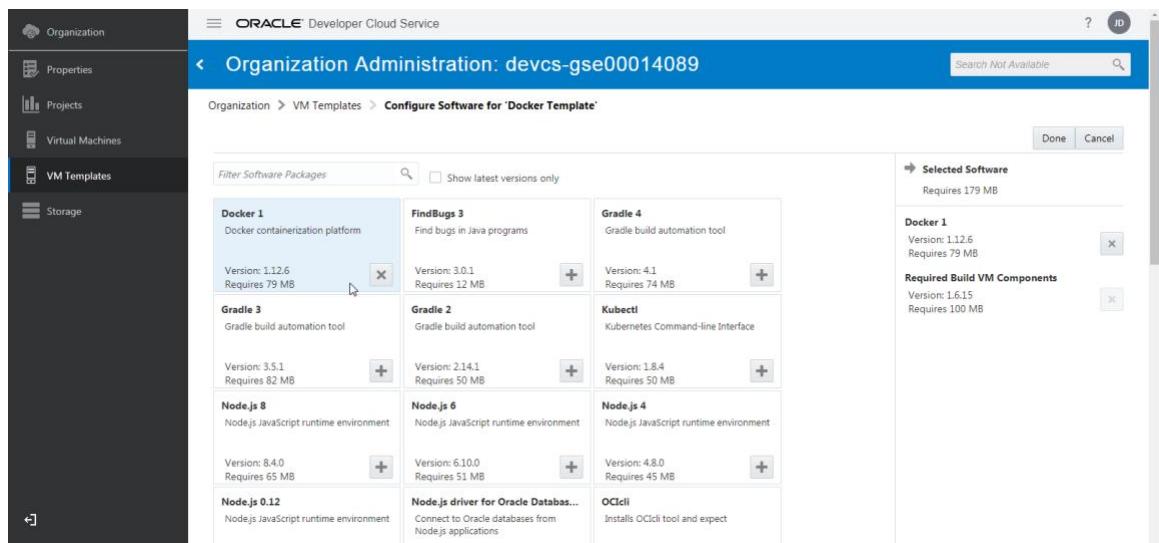
1. In the Organization menu where you have configured the Storage and Compute Cloud Service Configurations. Click on the VM Templates tab and then on the “Create” button. In the dialog box that pops up, give a template name such as **DockerTemplate**. Select Platform as Oracle Linux 7 and click on the Create button.



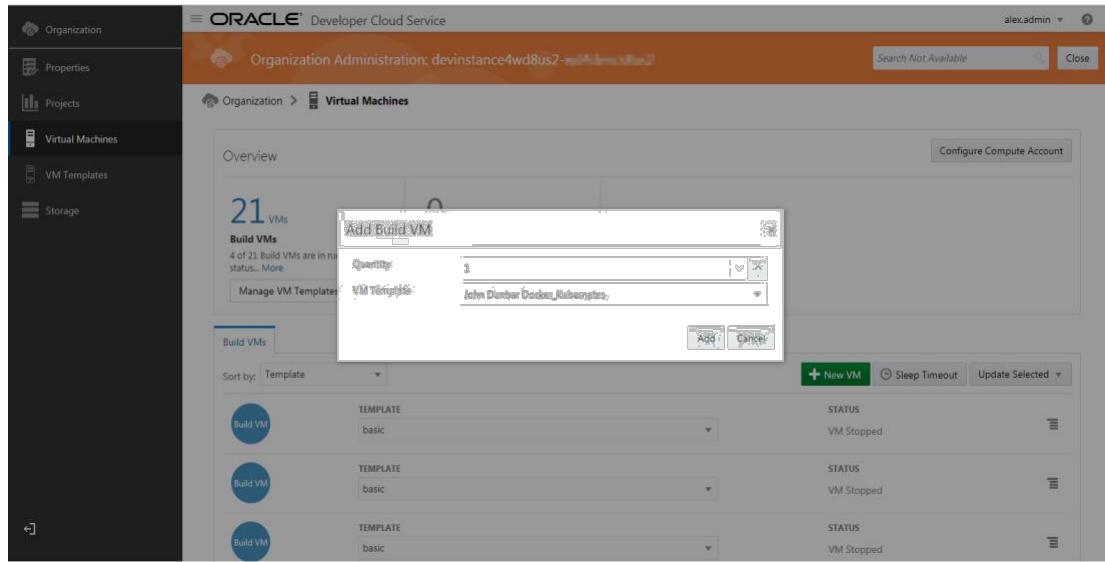
2. Now on creation of the template click on “Configure Software” button.



3 Select Docker from the list of software bundles available for configuration and click on the + sign for each software bundle that you intend to add to the template. Then click on “Done” button to complete the Software configuration.



4 Click on the Build Virtual Machines tab, then click on “Create” button and in the dialog box that pops up, enter the Quantity as 1 and select Docker Template for the VM Template dropdown. Then click the Add button.

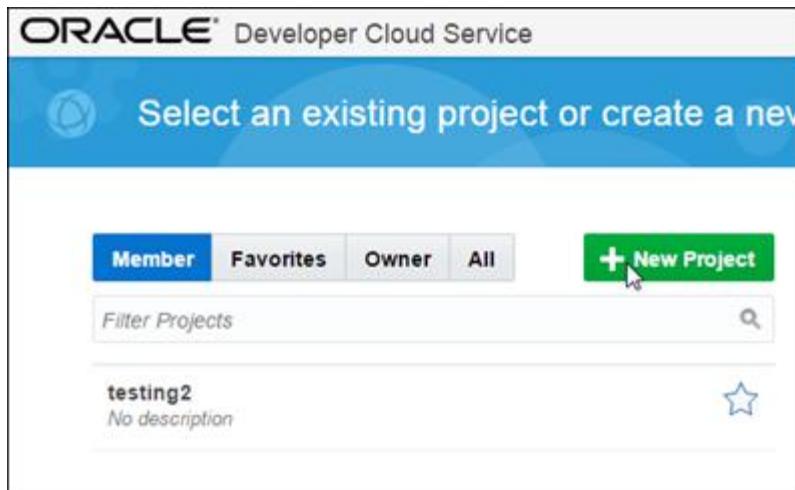


Now let's get started, with the lab!

Milestone 1: Create a project environment for your team

In this section, you'll provision a complete development platform for your team by leveraging DevCS's web interface.

1. Click on the Projects Tab of the Organization page in DevCS.
2. On the Welcome page, click **Create**.



4. Give your project a name that begins with your own name, such as John Dunbar OOW Hello, to make your project unique. Then:

- a. Enter a **Description**, such as OOW Hello project.
- b. Leave the Security setting specified as **Private**.
- c. Click **Next**.

New Project

Back Details Template Properties Next >

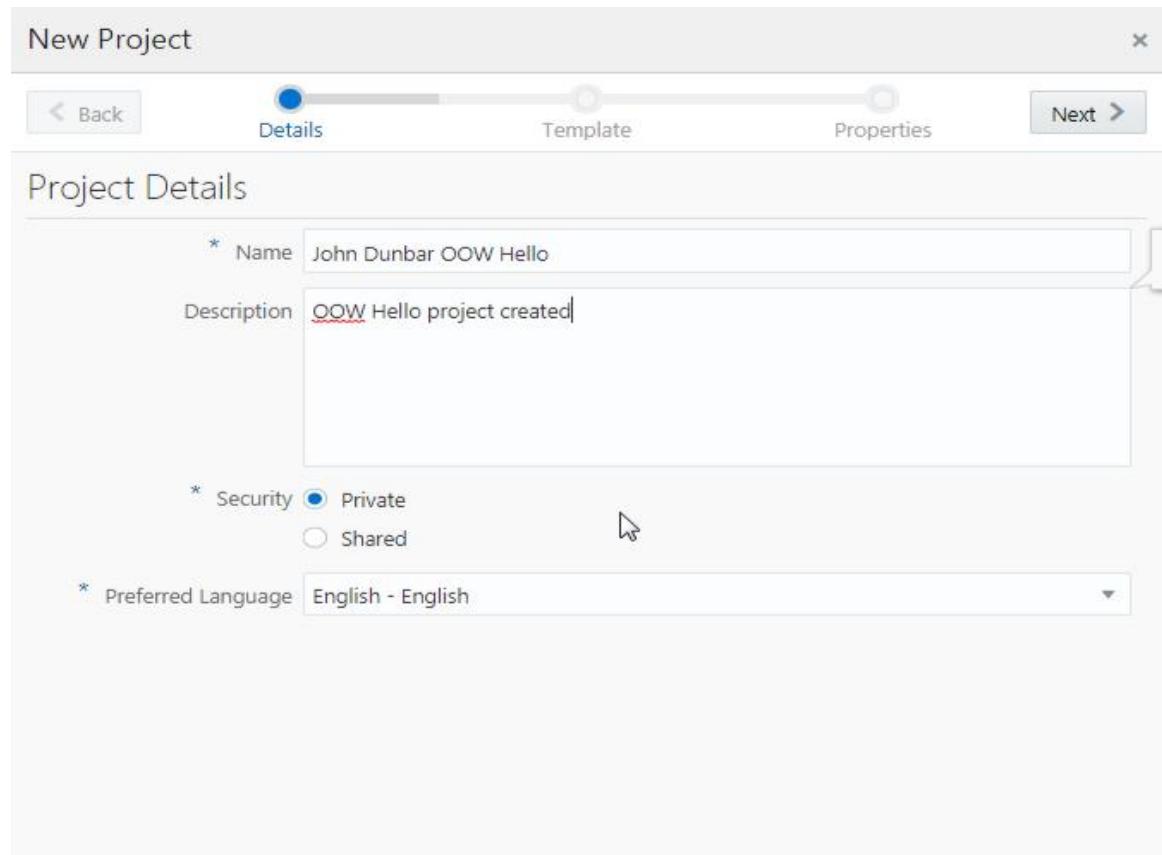
Project Details

* Name John Dunbar OOW Hello

Description OOW Hello project created

* Security Private Shared

* Preferred Language English - English



5. Click **Empty Project**, then click **Next**

New Project

Back Details Template Properties Next >

Template

Optionally create your project from a template to clone its settings, including any git repository content and build jobs.

- Empty Project**
Create a project with no preconfigured settings or content.
- Initial Repository**
Create a project with initial repository (empty, with README.md file or imported).
- Import Project**
Create a project from exported project data stored in Cloud Storage.
- Private Template**
Create a project from private template using its Private Key.

6. Select your preferred wiki markup language, then click **Finish**.

New Project

Back Details Template Properties Finish

Project Properties

* Wiki Markup Markdown ▾

- Wait while the project modules are provisioned, which can take a minute or two. You can see the indicators turn green as the associated modules are provisioned.



Project John Dunbar OOW Hello is being provisioned.

Provisioning may take up to several minutes.
Please wait until all modules are provisioned.

Agile	Docker	Merge Requests
Binary Repository	Environments	Mobile Build
Code	Issues	Project
Component Catalog	Mako	Wiki
Deploy	Maven	

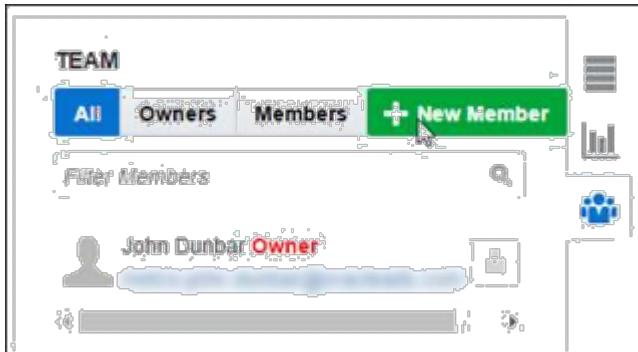
- When everything is provisioned, the project Home page opens, which contains details about your newly created project:

The screenshot shows the Oracle Developer Cloud Service Project Summary page for 'John Dunbar OOW Hello'. The left sidebar lists various project modules: Code, Maven, Environments, Releases, Snippets, Merge Requests, Issues, Agile, Build, Deploy, Docker Registry, Wiki, and Administration. The main content area has tabs for ENVIRONMENTS, RECENT ACTIVITIES - TODAY, and RECENT ACTIVITIES - YESTERDAY. The ENVIRONMENTS tab shows a 'Create Your First Environment' button and a 'Learn about environments' link. The RECENT ACTIVITIES section shows several log entries from 'Alex Admin' related to repository deletions and creation. On the right side, there are three panels: 'New Developer Cloud Service Discussion Forum' (with a note that it's part of the Cloud Customer Connect Platform), 'REPOSITORIES' (showing a 'Create New Repository' button and a note that no git repositories exist), and 'Maven' and 'Docker' sections (both showing empty lists). A search bar at the top right says 'Search Activities'.

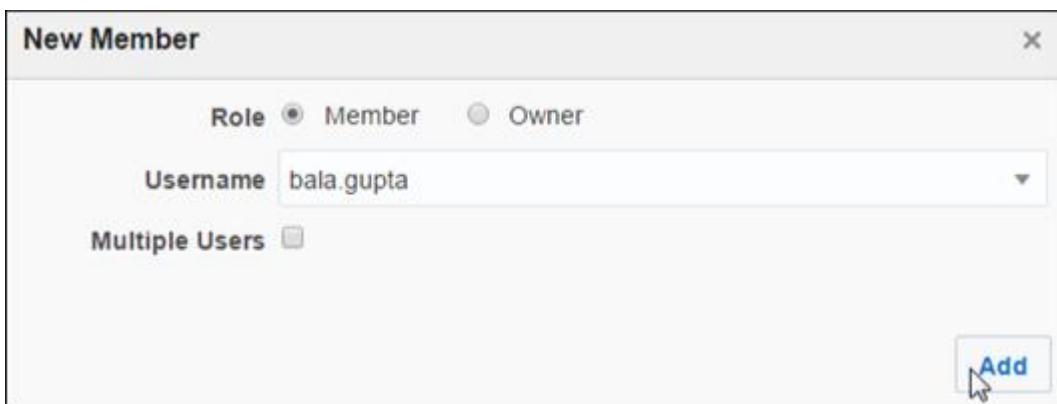
Let's take a look at this page (you may need to scroll to see the whole thing):

- On the left side is an activity feed.
- Tabs on the right side show you where the Git source code and Maven repositories are located.
- Also on the right you can see project statistics, as well as the UI where you can manage team members. Let's take a look at that UI now.

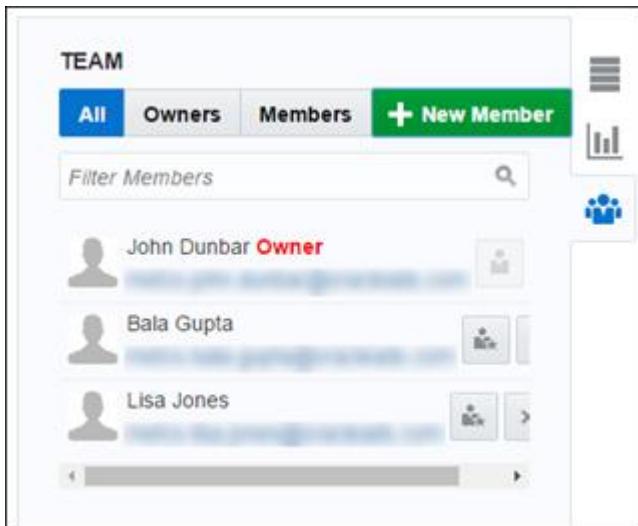
- Scroll to the right and click the Team tab, then click **+Add Member**.



10. Select another team member from the drop-down list and click **Add**.



11. Repeat steps 7 and 8 again so that your project team contains two other members besides yourself.



Each team member can now log in to the environment and start collaborating on project development.

Milestone 2: Fetch and review code from the Git repository

1. On the right side of the home page, click **New Repository**.

2. In the New Repository dialog, enter these details:

- NodeJS Docker in the **Name** field
- Nodejs Docker Repository in the **Description** field
- Import Existing Repository under **Initial content**
- <https://github.com/abhinavshroff/NodeJSMicroDocker> in the text box:

New Repository

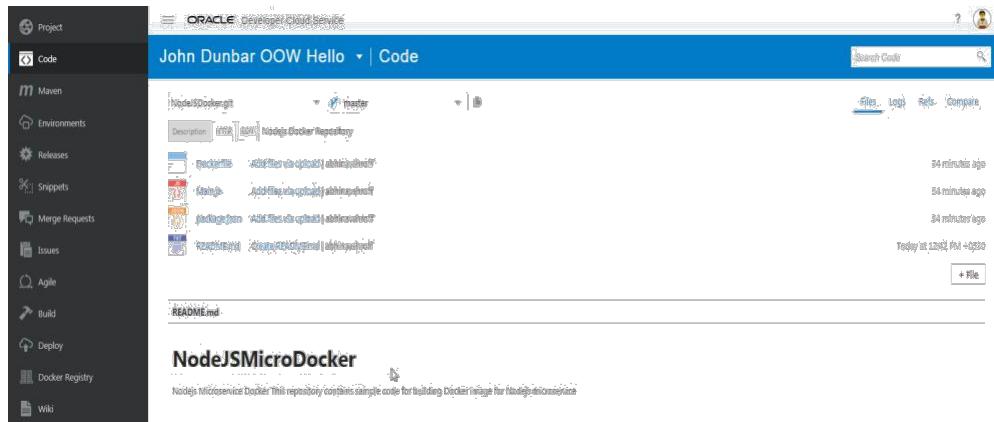
* Name

Description

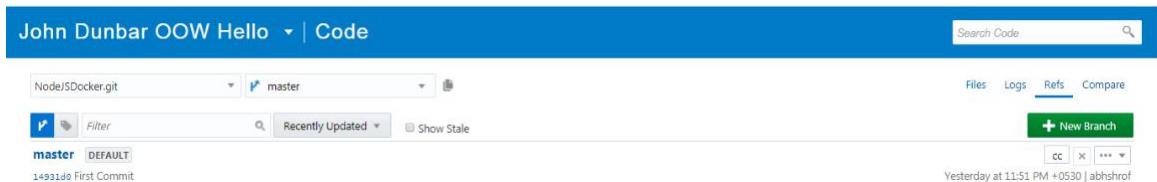
Initial content Empty Repository
 Initialize repository with README file
 Import existing repository

 ▶ Credentials for non-public repos

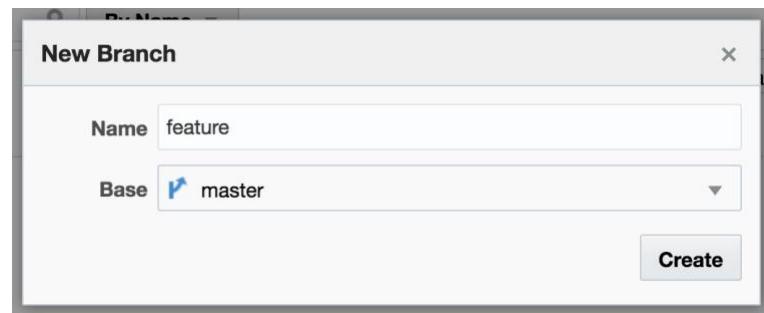
- Click **Create**.
- You should now be on the Code tab, which shows that you have a new DevCS git repository, NodeJS Docker.git. This new repository contains imported code from the GitHub repository you specified.



5. Click the **Refs** tab to view the current list of branches. At this point there's only one, the master branch:

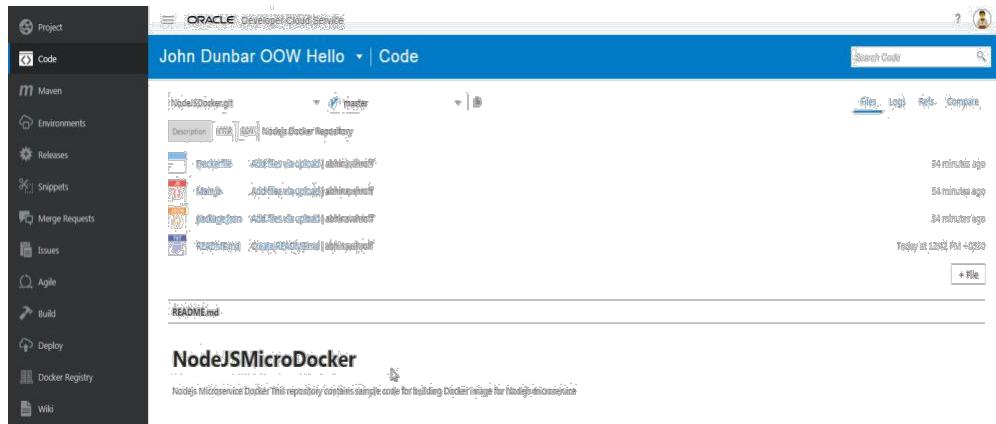


6. Click **New Branch**, and create a new branch called `feature`, based on the master branch.



7. Click **Files** in the upper right.

When you create a new branch, four files are automatically created: `Main.js`, `Dockerfile`, `package.json`, and `Readme.md`, which describes the repository.



7. Take a look at what's in each file by clicking each file name successively. In particular, notice that:

- Main.js contains the Node.js REST API code
- Dockerfile contains the commands to build a Docker image
- package.json defines the dependencies for the Node.js code

8. Let's go back to Main.js, and on the right notice that there are five methods for viewing a file on the Code tab:

- File
- Blame
- Logs
- Refs
- Compare

9. Click **Blame** tab to view the code and the file's commit history:

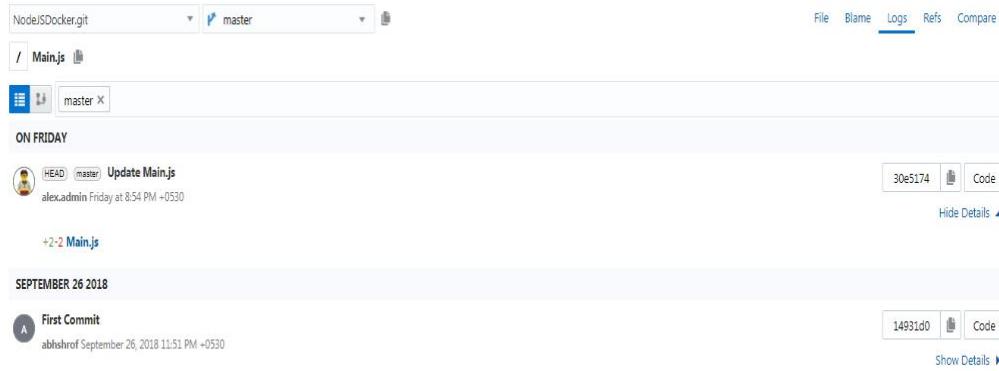
The screenshot shows the Main.js blame tab. It displays the commit history for the file, starting with the first commit from 'abhshrof' on Friday at 8:54 PM +0530, which updated Main.js. The code shown is:

```

1 var express = require("express");
2 var bodyParser = require("body-parser");
3 var app = express();
4 app.use(bodyParser.urlencoded());
5 app.use(bodyParser.json());
6 var router = express.Router();
7
8 router.get('/message',function(req,res){
9   res.json({"error": false, "message": "Hello OOW!"}); // Change OOW to your name here...
10 });
11
12 router.post('/add',function(req,res){
13   res.json({"error": false, "message": "success", "data": req.body.num1 + req.body.num2});
14 });
15
16 app.use('/',router);
17
18 app.listen(80,function(){
19   console.log("Listening at PORT 80");
20 });

```

10. Next, click **Logs**, where you can see the list of the commits for this branch, as well as the code that was pushed with each commit:



File Blame Logs **Logs** Refs Compare

/ Main.js

master X

ON FRIDAY

HEAD (master) Update Main.js
alex.admin Friday at 8:54 PM +0530

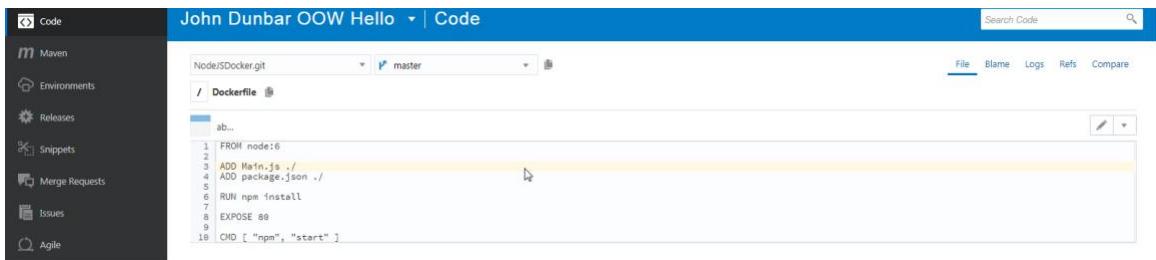
+2-2 Main.js

SEPTEMBER 26 2018

A First Commit
abnshrof September 26, 2018 11:51 PM +0530

14931d0 Code Show Details ▾

11. Click **Files**, then click **Dockerfile**. We'll use this file a bit later, when we build the Docker image.



Code

John Dunbar OOW Hello | Code

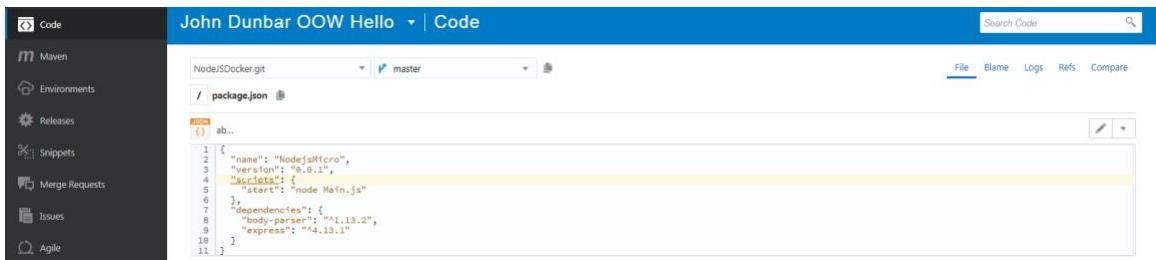
NodeSDocker.git master

/ Dockerfile

```
FROM node:6
ADD Main.js /
ADD package.json /
RUN npm install
EXPOSE 80
CMD [ "npm", "start" ]
```

File Blame Logs Refs Compare

12. Click the forward slash next to the Dockerfile name, then click **package.json** to review the Node.js code dependencies.



Code

John Dunbar OOW Hello | Code

NodeSDocker.git master

/ package.json

```
{ "name": "NodejsHello", "version": "0.0.1", "scripts": { "start": "node Main.js" }, "dependencies": { "body-parser": "^1.13.2", "express": "^4.13.1" } }
```

File Blame Logs Refs Compare

Milestone 3: Track issues for your project

In this section, we'll start tracking tasks (issues) that our team needs to take care of. We'll then organize these tasks into a sprint.

1. In the left nav bar click **Issues**, then click **New Issue**:

The screenshot shows the 'Track Issues' interface with the 'Recently changed' search selected. The results table is currently empty, showing the message 'No data to display.' Below the table, there is a navigation bar with buttons for 'Page', '1', '(0 of 0 items)', and a magnifying glass icon.

2. For your first issue:

- Type Change the message in Main.js in the **Summary** field
- Type Change OOW in message to my name in Main.js in the **Description** field
- Select **Task** in the **Type** drop-down list.

The screenshot shows the 'New Issue' creation form. In the 'Summary' field, the text 'Change the message in Main.js' is entered. In the 'Description' field, the text 'Change OOW in message to my name in Main.js' is entered. The 'Type' dropdown menu is open, showing 'Task' as the selected option.

- Scroll down, and from the **Status** drop-down list, select **Assigned**.
- From the **Owner** drop-down list, select **kevinevans** (your user).
- In the **Story Points** field, enter 3.

The screenshot shows the 'New Issue' creation form with more detailed information. The 'Story Points' field is set to 3. Other fields shown include 'Due Date' (mm/dd/yy), 'Estimated' (days), and 'Story Points' (3). The 'Type' dropdown is still set to 'Task'.

- Click **Create Issue**.
- Click the **Issues** tab to see the new issue:

The screenshot shows the 'All issues' search results. There is one item listed:

Type	ID	Summary	Component	Status	Owner	Created	Changed
Change	1	Change the message in Mail	Default	Assigned	Alex Admin	10/3/2018	10/3/2018

Below the table, it says 'Page 1 of 1 (1 of 1 items)'.

Task 4: Create an Agile board and a sprint to track issues

In this section, we explore the DevCS features that help you manage your agile development process.

1. In the left nav bar, click **Board**, then click **New Board**.

The screenshot shows the 'Find Board' page with the following interface elements:

- Filter buttons: Owned (selected), Favorites, All.
- Board Type buttons: Scrum (selected), Kanban, All Board Types.
- Search input: Filter Boards with a magnifying glass icon.
- Action button: + New Board (green button).

2. In the New Board dialog box:
 - a. Type Hello Board in the **Name** field
 - b. Select **Story Points** from the **Estimation** drop-down list
 - c. Click **Create**.

The screenshot shows the 'New Board' dialog box with the following fields:

Type	Scrum	Kanban
* Name	Hello Board	
* Search	All issues	
* Estimation	Story Points	

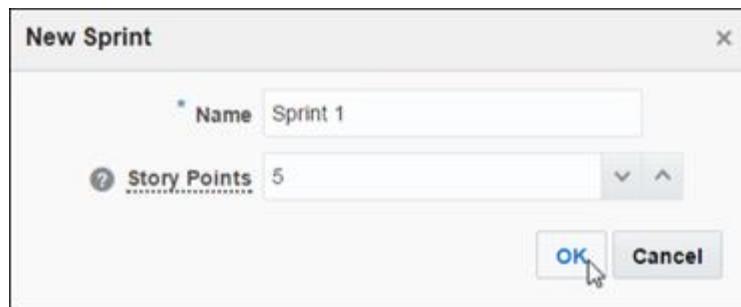
At the bottom are 'Create' and 'Cancel' buttons.

You can see the issue you created earlier in Backlog. Let's add it to a sprint.

3. Click **New Sprint**.

The screenshot shows the Jira interface with the 'Hello Board' board selected. At the top, there is a 'New Sprint' button. Below it, the backlog shows one issue: 'Task 1' with the description 'Change the message in Main.js'. A 'New Issue' button is also visible.

4. Let's call our new sprint **Sprint 1**. In the **Story Points** field enter 5, then click **OK**.



You can now see the new sprint, along with instructions on how to add issues to it.

5. Drag and drop Task 1 to Sprint 1, taking care to drop it right on top of the instructions.

The screenshot shows the Jira interface with the 'Sprint 1' board selected. The board header indicates 'Sprint 1 0 issues'. Below the header, there is a message: 'This sprint contains no issues. Drag and drop issues from Backlog or right click an issue to send it to a sprint.' A 'New Issue' button is also visible.

6. Click **Start Sprint** to make the sprint active.

The screenshot shows the Jira interface with the 'Sprint 1' board selected. The board header indicates 'Sprint 1 1 issue'. Below the header, there is a message: 'This sprint contains 1 issue. Change the message in Main.js'. A 'Start Sprint' button is visible. To the right of the board, there is a progress bar with three colored boxes: blue, orange, and green, with the number '3' above each.

7. In the Start Sprint dialog box, accept the default start and end dates and click **Start**.

By default, a sprint lasts two weeks. You can now see the default dates above the issue list for Sprint 1.

The screenshot shows a 'Start Sprint' dialog box. It has fields for 'Name' (Sprint 1), 'Start Date' (10/03/18 23:15), 'End Date' (10/17/18 23:15), and 'Story Points' (5). There are also 'Start' and 'Cancel' buttons.

- Click **Active Sprints** to view the progress of the sprint. You can see at a glance all the tasks considered To Do, In Progress, or Completed.

The screenshot shows the DevCS interface with the 'Hello Board' selected. The 'Active Sprints' tab is active. The board has three columns: 'To Do (0)', 'In Progress (1)', and 'Completed (0)'. A single task titled 'Task 1: Change the message in Main.js' is in the 'In Progress' column, assigned to 'alex.admin'.

Milestone 5: Edit your code and commit it to the working branch

In this section, we'll see how DevCS helps you edit your code and commit it to the working branch.

- Click **Git** in the left nav, then select the NodejsDocker.git repository and the feature branch, as shown here:

NodeJSMicroDocker

Nodejs Microservice Dockerfile repository contains sample code for building Docker image for Nodejs microservice

2. Click **Main.js**, then click the pencil icon to edit the file.

```
1 var express = require("express");
2 var bodyParser = require("body-parser");
3 var app = express();
4 app.use(bodyParser.urlencoded());
5 app.use(bodyParser.json());
6 var router = express.Router();
7
8 router.get('/message',function(req,res){
9   res.json({error : false, "message" : "Hello OOW!"}); // Change OOW to your name here...
10 });
11
12 router.post('/add',function(req,res){
13   res.json({error : false, "message" : "success", "data" : req.body.num1 + req.body.num2});
14 });
15
16 app.use('/',router);
17
18 app.listen(80,function(){
19   console.log("Listening at PORT 80");
20 })
```

3. Find where it says “Hello OOW!” and change “OOW” to your own name.

```
1 var express = require("express");
2 var bodyParser = require("body-parser");
3 var app = express();
4 app.use(bodyParser.urlencoded());
5 app.use(bodyParser.json());
6 var router = express.Router();
7
8 router.get('/message',function(req,res){
9   res.json({error : false, "message" : "Hello Abhinav!"}); // Change OOW to your name here...
10 });
11
12 router.post('/add',function(req,res){
13   res.json({error : false, "message" : "success", "data" : req.body.num1 + req.body.num2});
14 });
15
16 app.use('/',router);
17
18 app.listen(80,function(){
19   console.log("Listening at PORT 80");
20 })
```

4. Click **Commit**.

5. In the **Commit changes** dialog, click **Commit** to commit these changes to the NodeJSDocker.git's feature branch:

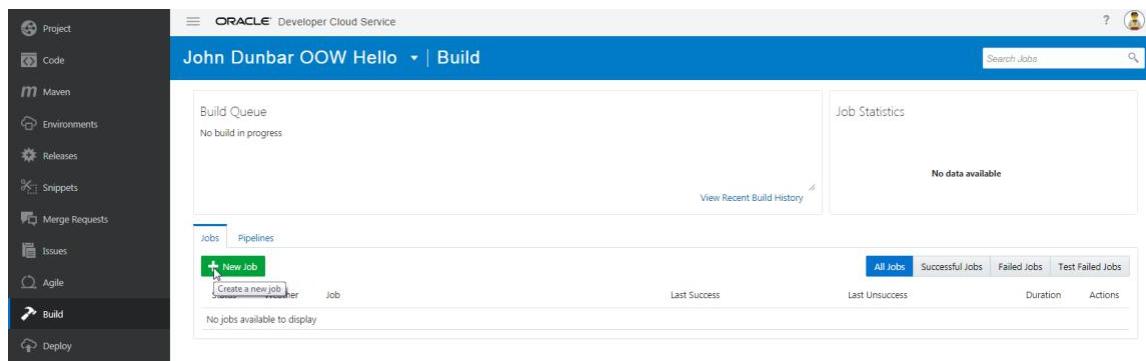


Milestone 6: Configure the Docker build job

In this section, we'll see how to use DevCS to build the Docker image for the Node.js REST application. We'll actually configure two build jobs, one for NodeJSDockerOCIR and one for NodeJSDockerHub.

Let's start with the NodeJSDockerOCIR build job.

1. Click **Build** in the left nav bar, then click **New Job**.



2. In the New Job dialog:

- a. Type **NodeJSDockerOCIR** in the **Job Name** field

- b. Type Build and push Docker image to OCIR in the **Description** field
- c. Select **DockerTemplate** from the **Software Template** drop-down
- d. Click **Create Job**.

New Job

* Job Name: NodeJSDockerOCIR

Description: Build and Push Docker image to OCIR

Use for Merge Request

Create New Copy existing job

* Software Template: OOB_Template

Create Job **Cancel**

- 3. From the **Add Source Control** dropdown, select Git.

John Dunbar OOW Hello | Build

Jobs Overview > NodeJSDockerOCIR > Configure

Job Configuration

Source Control Build Parameters Build Environment Builders Post Build

Configure Source Control

Add Source Control ▾

Git

Save Cancel

- 4. Select NodeJSDocker.git from the **Repository** drop-down.
- 5. Select master from the **Branch** drop-down.
- 6. Click the **Automatically perform build on SCM commit** check box.

7. Click the **Steps** tab.
8. From the **Add Builder** drop-down, select **Docker Builder->Docker login**.
 - a. Type `iad.ocir.io` in the **Registry Host** field
 - b. Type `gse00014089/api.user` in the **Username** field
 - c. Type `1}UHkK2o[L7nAij7nE{R` in the **Password** field.

9. From the **Add Builder** drop-down, select **Docker Builder->Docker build**.
 - a. Type `iad.ocir.io` in the **Registry Host** field
 - b. Type `gse00014089/oowhol/nodejsmicrojodu` in the **Image Name** field. Replace **jodu** with the first two letters of your first and last name, respectively.
 - c. In the **Source** radio buttons, click **Context root in Workspace**.

10. From the Add Builder drop-down, select Docker Builder->Docker push.

- a. Type iad.ocir.io in the Registry Host field
- b. Type Type gse00014089/oowhol/nodejsmicrojodu in the Image Name field. Replace **jodu** with the first two letters of your first and last name, respectively.

Docker build

* Registry Host: e.g. 'quay.io', for Docker Hub leave empty

* Image Name: e.g. 'john-doe/my-hello-world'

Version Tag: e.g. '1.7.0'

Full Image Name:

Options: options

* Source: Context Root in Workspace:
 Dockerfile Text:
 Remote Context Root:

Context Root in Workspace: e.g. 'target'

Dockerfile: e.g. 'build_1_7/Dockerfile2'; leave empty for 'Dockerfile' in build context root

Docker push

Options:

* Registry Host: e.g. 'quay.io', for Docker Hub leave empty

* Image Name: e.g. 'john-doe/my-hello-world'

Version Tag: e.g. '1.7.0'

Full Image Name:

11. Click Save.

John Dunbar OOW Hello | Build

Jobs Overview > NodeJS Docker OCIR > Configure

Job Configuration

Save Cancel

Source Control Build Parameters Build Environment Builders Post Build

Configure Builders Add Builder

Docker login

Docker logout will be performed automatically at the end of all build steps.

Registry Host: e.g. 'quay.io', for Docker Hub leave empty

* Username: e.g. 'johndoe'

* Password: Password

+ Link External Registry

You've finished configuring the first build job, so now let's move on to the NodeJSDockerHub build.

1. Click **Build** in the left nav bar.
2. Click **New Job**.

John Dunbar OOW Hello | Build

Build Queue
No build in progress.

Job Statistics
No data available

New Job

Status	Job	Last Success	Last Unsuccess	Duration	Actions
N/A	NodeJS DockerHub	N/A	N/A	< 1 Sec	

3. In the New Job dialog:

- Type **NodeJS DockerHub** in the **Job Name** field
- Type **Build and push Docker image to DockerHub** in the **Description** field
- Select **OOW_Template** from the **Software Template** drop-down

New Job

* Job Name	NodeJS DockerHub
Description	Build and Push Docker Image to <u>DockerHub</u>
<input type="checkbox"/> Use for Merge Request	
<input checked="" type="radio"/> Create New <input type="radio"/> Copy existing job	
* Software Template	OOW_Template
<input type="button" value="Create Job"/> <input type="button" value="Cancel"/>	

4. Click **Create Job**.
5. From the **Add Source Control** drop-down, select Git.

The screenshot shows the Jenkins interface for configuring a job named "John Dunbar OOW Hello". The "Configure" screen is displayed, with the "Source Control" tab selected. A dropdown menu titled "Add Source Control" is open, and the option "Git" is highlighted with a cursor icon.

6. Select NodeJSDocker.git in the **Repository** drop-down.
7. Select master from the **Branch** drop-down.

The screenshot shows the Jenkins interface for configuring a job named "John Dunbar OOW Hello". The "Configure" screen is displayed, with the "Source Control" tab selected. A "Git" configuration dialog is open, showing the repository "NodeJSDocker.git" and the branch "master". The "Automatically perform build on SCM commit" checkbox is unchecked.

8. Click the **Steps** tab.
9. From the **Add Step** drop-down, select **Docker ->Docker login**.
 - a. Type odcsexpo in the **Username** field
 - b. Type odcsexpo in the **Password** field.

10. From the **Add Step** drop-down, select **Docker ->Docker build**.

- Type **odcsrepo/nodejsmicrojodu** in the **Image Name** field. Replace **jodu** with the first two letters of your first and last name, respectively.
- In the **Source** radio buttons, click **Context root in Workspace**.

11. From the **Add Step** drop-down, select **Docker ->Docker Push**.

- Type **odcsrepo/nodejsmicrojodu** in the **Image Name** field. Replace **jodu** with the first two letters of your first and last name, respectively.

Docker build

* Registry Host: e.g. 'quay.io', for Docker Hub leave empty

* Image Name: odcsrcpo/nodejsmicrojobdu

Version Tag: e.g. '1.7.0'

Full Image Name: odcsrcpo/nodejsmicrojobdu

Options: options

* Source: Context Root in Workspace
 Dockerfile Text:
 Remote Context Root:

Context Root in Workspace: e.g. 'target'

Dockerfile: e.g. 'build_A_7/Dockerfile2'; leave empty for 'Dockerfile' in build context root

Docker push

Options:

* Registry Host: e.g. 'quay.io'; for Docker Hub leave empty

* Image Name: odcsrcpo/nodejsmicrojobdu

Version Tag: e.g. '1.7.0'

Full Image Name: odcsrcpo/nodejsmicrojobdu

12. Click **Save**.

Milestone 7: Create and execute the build pipeline

In this section, we'll create a pipeline to run the two build jobs we just created.

1. Click **Build** in the left nav, then click **Pipelines**:

The screenshot shows the 'Build Queue' section with the message 'No build in progress'. Below it is a navigation bar with tabs 'Jobs' and 'Pipelines', where 'Pipelines' is currently selected. A large green button labeled '+ New Pipeline' is visible. To the right is a 'Job Statistics' section featuring a green circular progress bar labeled '100%' and a small green square icon labeled 'Success'.

2. Click **New Pipeline**.
3. Let's use **DockerPipeline** as the name.
4. Select both check boxes, as shown here:

Create Pipeline

Name: **DockerPipeline**

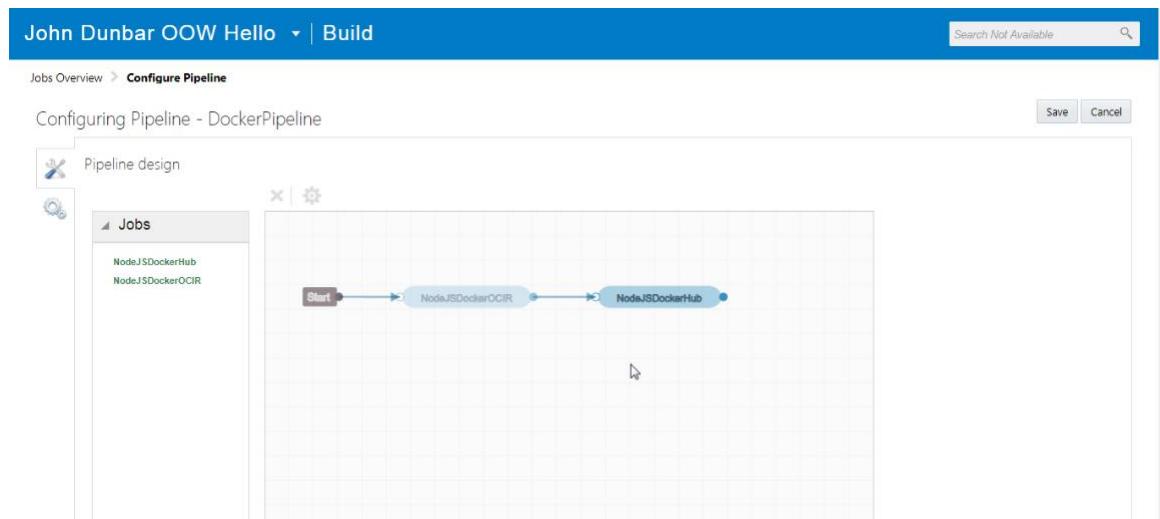
Description:

Auto start when pipeline jobs are built externally

Disallow pipeline jobs to build externally when the pipeline is building

Create **Cancel**

5. Click **Create**.
6. Drag NodeJSDockerOCIR to the gridded area, and drop it right after the **Start** bubble.
7. Drag NodeJSDockerHub and drop it right after NodeJSDockerOCIR.
8. User the cursor to draw areas between them, as shown here:

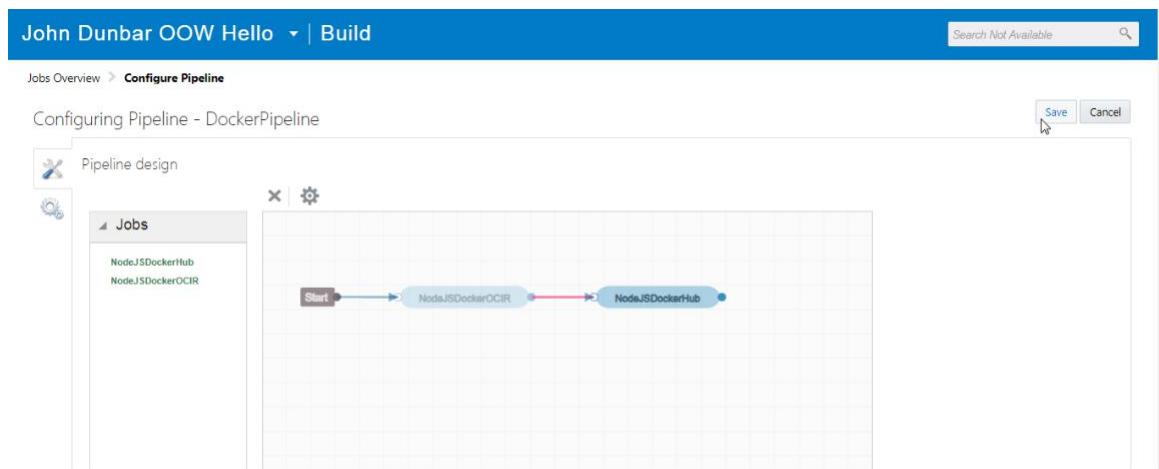


9. Double-click the line connecting the two build jobs.
10. In the editor, select Successful from the **Result Condition** drop-down:



11. Click **Apply**.

12. Click **Save**:



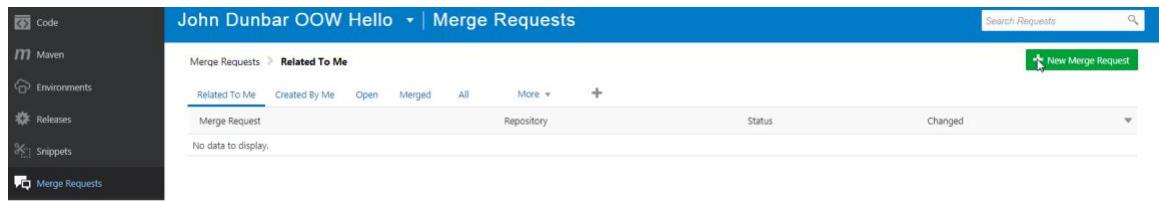
Milestone 8: Merge your feature branch with the master branch of the NodeJSDocker.git repository

In this section, we explore how to review and merge code in DevCS.

1. Click **Merge Requests** in the left nav bar:



2. Click New Merge Request:



3. In the New Merge Request dialog:

- Select NodeJsDocker.git from the **Repository** drop-down
- Select master as the **Target Branch**
- Select feature as the **Review Branch**
- Click the commit that appears for the review branch
- Click **Next**.

New Merge Request

Branch

Details

Description

* Repository NodeJS Docker.git

* Target Branch master

* Review Branch feature

alexadmin Today
8def1a5 Update Main.js

Back Next >

Create Cancel

4. On the Details page:

- a. Select yourself as the reviewer under **Reviewers**
- b. In the **Linked Issues** field, type Task until you see the task you created earlier.
- c. Click the task description
- d. Click **Next**.

New Merge Request

Back Branch Details Description Next >

Linked Issues: Task

Linked Builds: Task 1: Change the message in Main.js

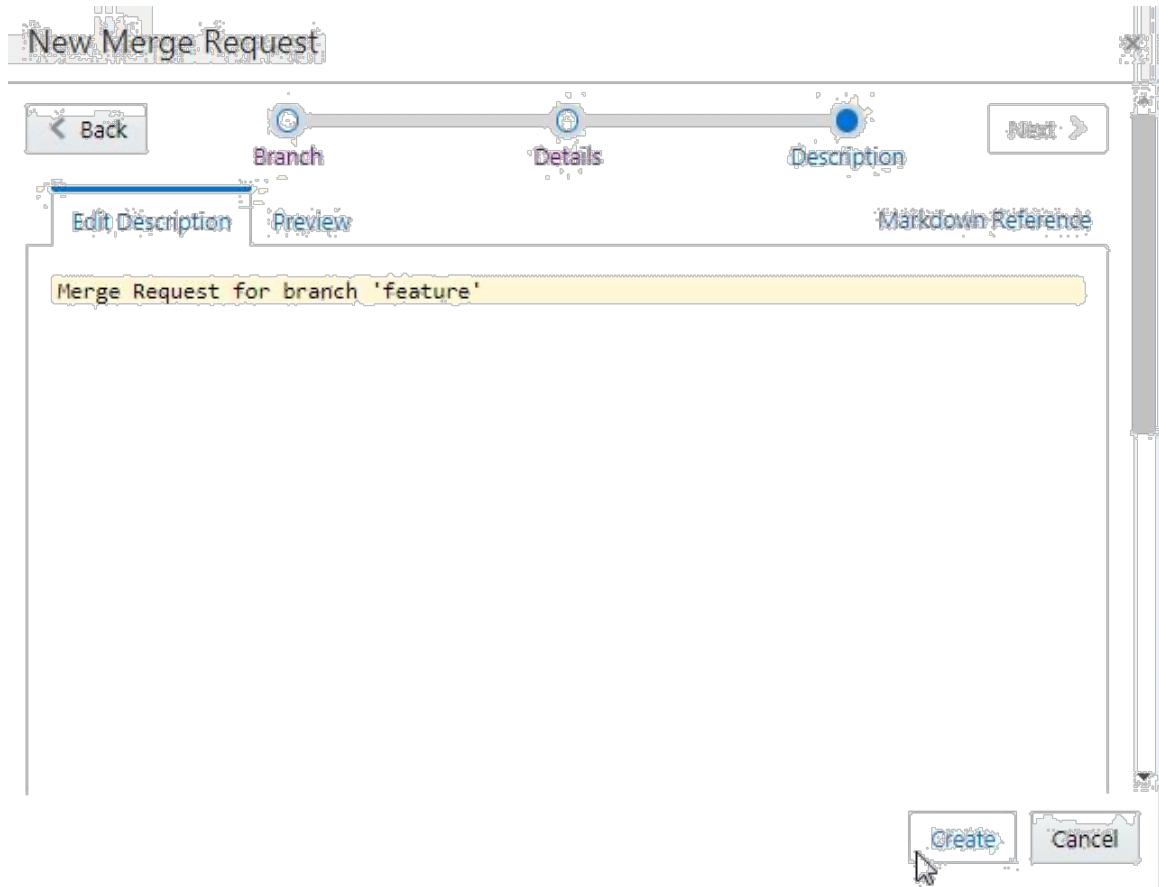
Tags: Select Tags

Summary: Merge Request for branch "feature"

* Reviewers: Alex Admin X

Create Cancel

5. Review the description and click **Create**:



6. Click the **Changed Files** tab and review the changes:

```

diff --git a/Main.js b/Main.js
index 867..174 100
--- a/Main.js
+++ b/Main.js
@@ -9,2 +9,2 @@
 9:     res.json({ "error": false, "message": "Hello OOW!"}); // Change OOW to your name here...
 9:     res.json({ "error": false, "message": "Hello Abhinav!"); // Change OOW to your name here...
@@ -18,2 +17,2 @@
18: });
19: router.post('/add',function(req,res){
@@ -17,2 +17,2 @@
17:     app.listen(80,function(){
18:         console.log("Listening at PORT 80");
19:     })
20: })

```

7. If the changes look correct, click **Approve**:

John Dunbar OOW Hello | Merge Requests

Merge Requests > 2 · Task 1 · Change the message in Main.js

OPEN  Alex Admin wants to merge 1+ commits to **master** from **feature** in **NodeJS Docker.git**

Conversation **Commits (1+)** **Changed Files (1)** **Unified Diff (1)** **Unmerged (0)** **1** **Filter** **Review Status**

Main.js (22 lines)

```

1  // Main.js (22 lines)
2
3  var router = express.Router();
4
5  router.get('/message',function(req,res){
6      res.json({"error": false, "message": "Hello OOW!"}); // Change OOW to your name here...
7
8      res.json({"error": false, "message": "Hello Abinav!"}); // Change OOW to your name here...
9
10 })
11
12 router.post('/add',function(req,res){
13
14     app.listen(80,function(){
15         console.log("Listening at PORT 80");
16
17     })
18
19 })
20
21
22

```

Merge                                                                 <img alt="Merge icon" data-bbox="828 5204 845

John Dunbar OOW Hello ➔ | Merge Requests

Merge Requests ➔ 2 Task 1 - Change the message in Main.js ➔

OPEN  Alex Admin wants to merge 1+ commits to **master** from **feature** in [NodeJS Docker git](#)

Conversation Community (1+) Changed Files (0) Listed Issues (0) Listed Pulls (0)

0:1 1:2 2:2

Main.js ↗ 2:2

1:1 2:2 3:3 4:4 5:5 6:6 7:7 8:8 9:9 10:10 11:11 12:12 13:13 14:14 15:15 16:16 17:17 18:18 19:19 20:20

```
= 0:1 -0:7 +0:7 @@  
1:1 var router = express.Router();  
2:2  
3:3 router.get('/message',function(req,res){  
4:4 res.json({error : false, message : "Hello OOW!"}); // Change OOW to your name here...  
5:5 res.json({error : false, message : "Hello Abhinav!"}); // Change OOW to your name here...  
6:6 } );  
7:7  
8:8 router.post('/add',function(req,res){  
9:9 res.json({error : false, message : "Data added successfully"});  
10:10 } );  
11:11 app.listen(80,function(){  
12:12 console.log("Listening at PORT 80");  
13:13 } );  
14:14 } );
```

Merge 
Approve 
Reject 
Close 

Review Status

APPROVE   Alex Admin 

10. Keep the defaults, then click **Create a Merge Commit**.

The screenshot shows a 'Merge' dialog box. At the top left is the title 'Merge'. On the right side are two buttons: 'Remember my choice' (unchecked) and 'Cancel'. The main area is titled 'Merge Options' and contains a section 'Create a Merge Commit' with a note: 'The 1 commits from branch 'feature' will be added to the branch 'master' via a merge commit. Merge commit with two parents will be created (non-fast-forward)'. Below this is a section titled 'Merge Request 2 from 'feature' into 'master''. It lists 'Resolved issues:' followed by '1) Task 1 - Change the message in Main.js.' At the bottom left is a 'Post Merge Actions' section with two checked checkboxes: 'Delete branch' and 'Resolve linked issues'. The 'Resolve linked issues' checkbox has a sub-item 'Task 1 - Change the message in Main.js' listed under it. At the bottom right are two buttons: 'Create a Merge Commit' (highlighted with a cursor arrow) and 'Cancel'.

You should now see the Review Status as Approve, with a green tick.

This merge will trigger the execution of the **NodeJSDockerOCIR** build job, which in turn will trigger the execution of the **DockerPipeline**.

The screenshot shows a 'Merge Requests' page with a single merged pull request. The pull request details are as follows:

- Merge Requests: 2 Task 1 - Change the message in Main.js
- MERGED acc52a3 · Alex Admin merged 1+ commits to master from feature in NodeJSDecker.git
- Conversation: 1 message
- Commits (1+)
- Changed Files (3)
- Linked Issues (0)
- United Labels (0)
- Review Status: APPROVED by Alex Admin

The commit history shows:

- Alex Admin applied 1 commits 48 minutes ago: Update Main.js
- Alex Admin applied 1 commits 48 minutes ago: ok
- Alex Admin merged branch 'feature' into 'master' just now: Merge Request 2 from 'feature' into 'master'

11. Click **Build** in the left nav bar.

12. Click **Jobs**:

The 'Build' page displays the 'Job Queue' section with two pending jobs:

Job	Build	Scheduled	Progress
NodeJSDeckerOCIR	#6	Just now	Waiting for Executor

On the right, the 'Job Statistics' section shows a 100% success rate with a green circle.

The main job list table has the following columns:

Jobs	Pipelines	All Jobs	Successful Jobs	Failed Jobs	Test Failed Jobs	
New Job						
Status	Weather	Job	Last Success	Last Unsuccess	Duration	Actions
✓	☀️	NodeJSDeckerHub	#3 10 min	13 hrs	30 secs	↻ ⚙️ ☰ ✖️
✓	☁️	NodeJSDeckerOCIR	#5 11 min	37 min	1 mins 33 secs	↻ ⚙️ ☰ ✖️

You should see your two build jobs, waiting to be built.

13. Click the **NodeJSDeckerOCIR** build job.

14. Click **Build Log**, as shown here:

John Dunbar OOW Hello | Build

Jobs Overview > NodeJS Docker OCIR

Job Details

Build and Push Docker Image to OCIR

Notifications: On CC Me

Build History

By	Status	Build	Started	Duration	Actions
	Success	#41	12 minutes ago	1m 38s	
	Success	#42	14 minutes ago	18s	

Build Trend

Build Number: 5

Build Duration: 1m 38s

Build Status: Success

15. Take a look at the log for the NodeJS DockerOCIR build job.

```

----> 404cf4a823ae
Successfully built 404cf4a823ae
Using docker://bin/docker
/bin/docker push iad.ocir.io/devcstest/oowhol/nodejsmicrojobu
The push refers to a repository [iad.ocir.io/devcstest/oowhol/nodejsmicrojobu]
db41ee45fcfb9: Preparing
74371bfbfe3e: Preparing
2343c7967ba4: Preparing
6c9a1bc6b767: Preparing
f059848e1802: Preparing
2793dc0607dd: Preparing
74800c25aa8c: Preparing
ba504a540674: Preparing
ba504a540674: Preparing
81101ce649d5: Preparing
daf45b2cad9a: Preparing
8c466bf4ca6f: Preparing
2793dc0607dd: Waiting
74800c25aa8c: Waiting
ba504a540674: Waiting
81101ce649d5: Waiting
daf45b2cad9a: Waiting
8c466bf4ca6f: Waiting
74371bfbfe3e: Layer already exists
f059848e1802: Layer already exists
db41ee45fcfb9: Layer already exists
6c9a1bc6b767: Layer already exists
2793dc0607dd: Layer already exists
ba504a540674: Layer already exists
74800c25aa8c: Layer already exists
81101ce649d5: Layer already exists
daf45b2cad9a: Layer already exists
8c466bf4ca6f: Layer already exists
2343c7967ba4: Layer already exists
latest digest: sha256:31b304cc366a64f48032282918dc7c9e5a8b670db18463cbea2b8d46860447dd size: 2632
/bin/docker logout iad.ocir.io
Remove login credentials for iad.ocir.io
Slave log size 2.1 KB (2,141)

Build completed.
Status: DONE Result: SUCCESSFUL Duration: 26 sec

```

16. Click the NodeJS DockerHub build job and take a look at its build log too:

Job Details

Build and Push Docker Image to DockerHub

Notifications: On

Build History

By	Status	Build	Started	Duration	Actions
John Dunbar	Success	#1	49 minutes ago	50 s	View Log Delete
John Dunbar	Success	#2	29 minutes ago	24 s	View Log Delete
John Dunbar	Success	#3	29 minutes ago	44 s	View Log Delete
John Dunbar	Success	#4	49 minutes ago	44 s	View Log Delete

Build Trend

Duration (s)

Build Number

Legend: Canceled (Red), Failed (Orange), Test Failed (Yellow), Success (Green)

17. Inspect the build job:

John Dunbar OOW Hello | Build

Jobs Overview > NodeJS DockerHub > #4 > Log

Build Log

Download Log

```

Build scheduled. Build started by pipeline DockerPipeline [Upstream Job: NodeJS DockerOCIR Build: #6]
Build task id: 129ee178-8255-4128-8023-f5abd512de95
Build execution started.
Building on slave OOW_Template
Git: Checkout directory is the workspace root.
Git: Fetching from remote repository https://devinstance4wd8us2-wd4devcs8us2.uscom-central-1.oraclecloud.com/devinstance4wd8us2-wd4devcs8us2/s/devinstance4wd8us2-wd4devcs8us2_john-dunbar-oow-
hello_52638cm/NodeJS/Docker.git
Git: Checking out branch master
Git: Done
Using docker /bin/docker
/bin/docker login -u odcsexpo -p **** https://index.docker.io/v1/
Flag --email has been deprecated, will be removed in 1.13.
Login Succeeded
Using docker /bin/docker
/bin/docker build -t odcsexpo/nodejsmicrojodu .
Sending build context to Docker daemon 74.24 kB

Step 1 : FROM node:6
--> f64a15ead359
Step 2 : ADD Main.js .
--> Using cache
--> 004b36f9e212
Step 3 : ADD package.json .
--> Using cache
--> 97af4b87d5d3
Step 4 : RUN npm install
--> Using cache

```

18. Open Google Chrome and go

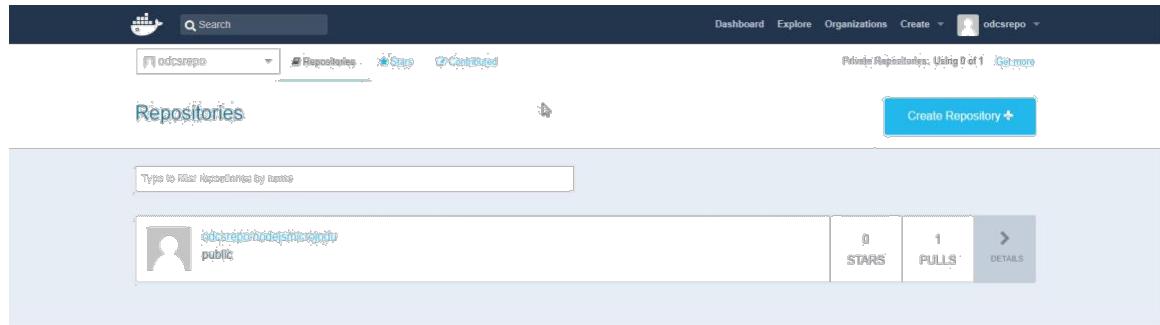
to: <https://hub.docker.com>

19. Sign in using these credentials:

Username: odcsexpo

Password: odcsexpo

You should see your Docker image in the repository!



Summary

Now that you've finished this lab, you should know how to build a pipeline to build and push a Docker image to Oracle Cloud Infrastructure Registry and DockerHub registry using Developer Cloud. Specifically, you learned how to:

- Work with Oracle Developer Cloud Service and its most important features
- Create a project and pull the code from an existing GitHub repository to a DevCS-hosted Git repository
- Create a task within DevCS and an Agile board to track it
- Complete the task by branching the code, editing it using the online editor, and merging the code
- Build the Docker image with Continuous Integration in DevCS
- Push the Docker image to the Oracle Cloud Infrastructure Registry and DockerHub.