Cloud Developer Certification Preparation

Exercise 5.4:

Managing source code for projects in IBM Bluemix DevOps Services

Exercise 5.4: Prerequisites

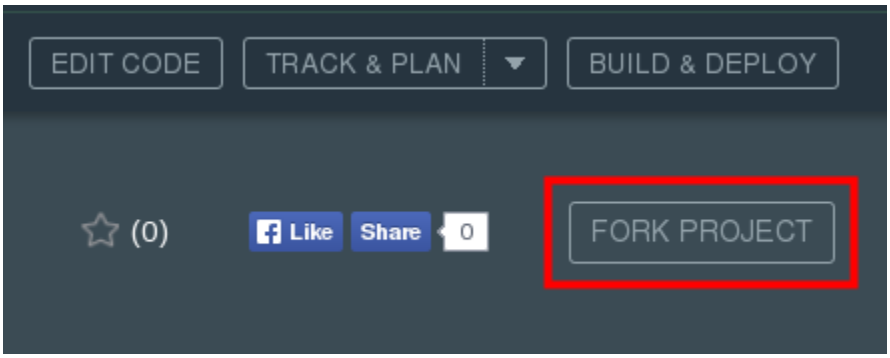
Sign up for a 30-day free trial IBM Bluemix account if you don't already have one. Also, create a DevOps account: https://hub.jazz.net/. You should use the same credentials for both the Bluemix and DevOps accounts.

You also need the following software:

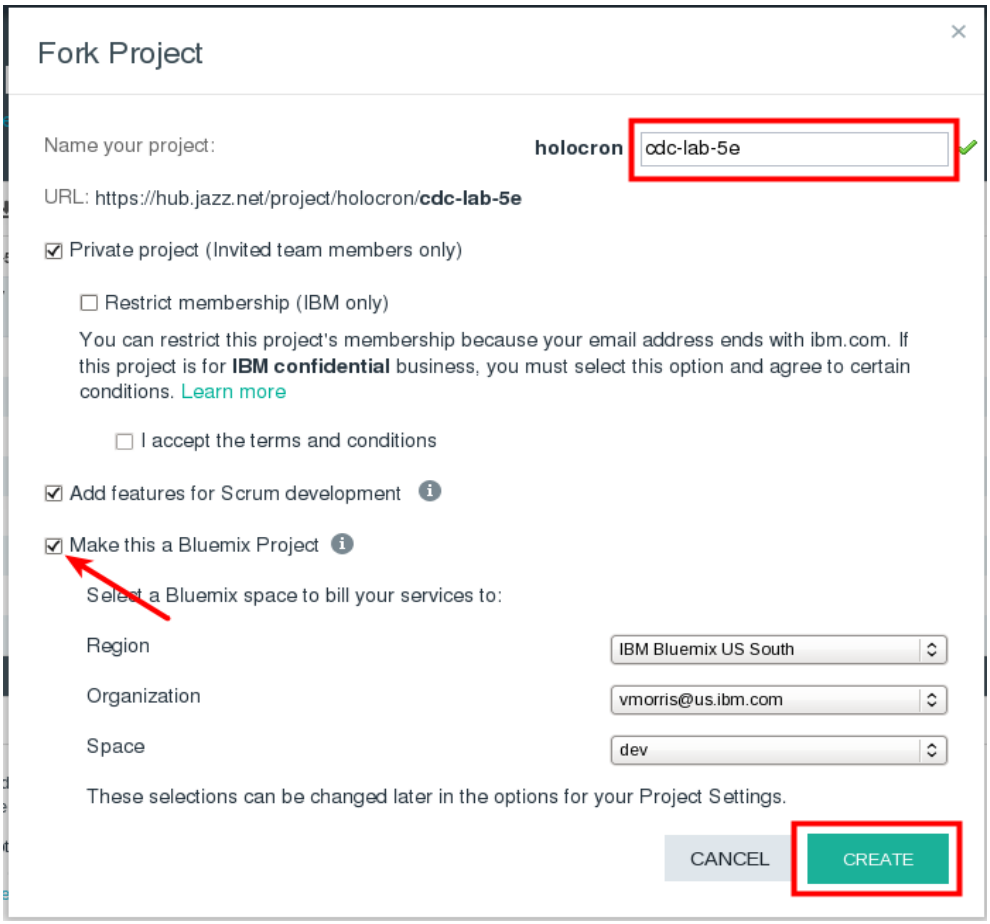
- A web browser supported by Bluemix:
    - Chrome: the latest version for your operating system
    - Firefox: the latest version for your operating system and ESR 31 or ESR 38
    - Internet Explorer: version 10 or 11

Exercise 5.4.1: Forking a project

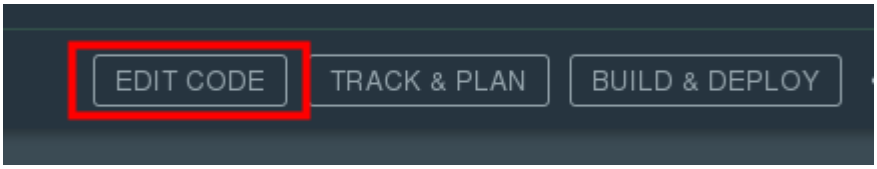
1. Log in to IBM DevOps Services and fork the project located at https://hub.jazz.net/project/ecosysdevcnc/cdc-lab-5/overview into a new project.



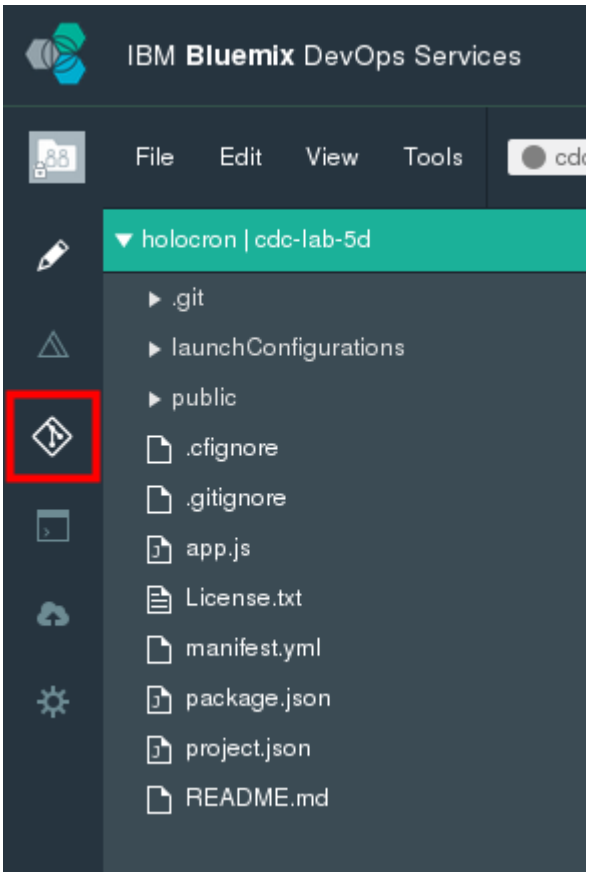
2. Give the new project a unique name, select all the check boxes, and choose an appropriate Bluemix runtime configuration. Click **CREATE**.



3. After you see the message about successfully creating your project, click **EDIT CODE**.

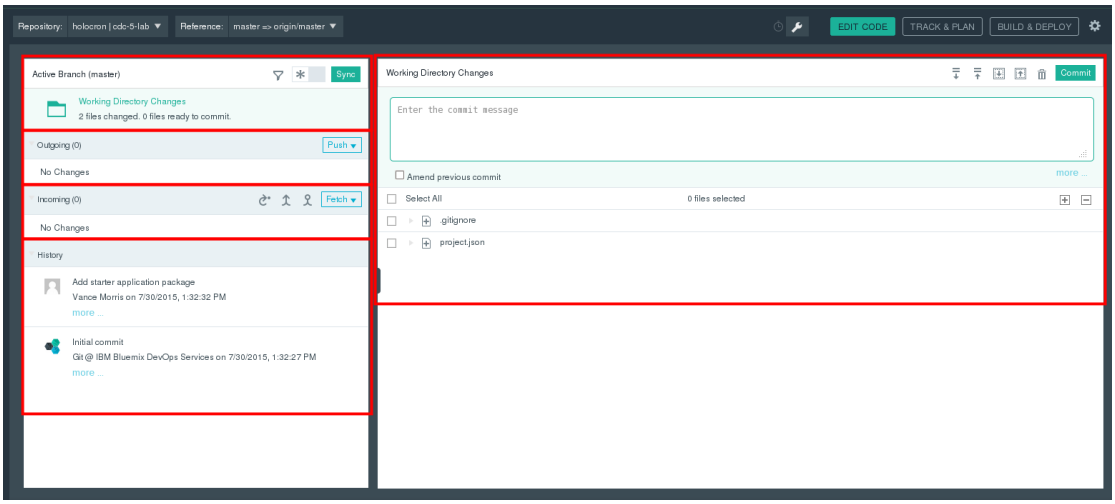


4. Wait a moment for the workspace to be set up. When the project is ready, click the **Git** icon located on the left column.

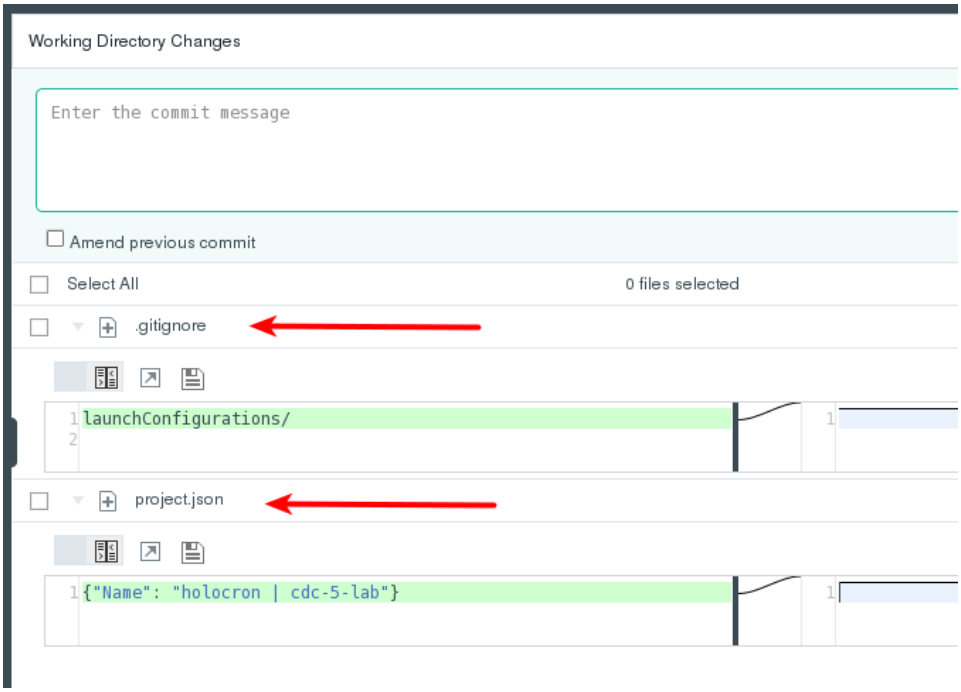


Exercise 5.4.2: Adding local changes to the Git repository

The Git Repository view, or "Git view" is organized into two panes. The left pane shows any staged outgoing changes, any incoming changes, and the active branch's history. The right pane shows any working directory changes and provides the functionality to select changed files, enter a commit message, and stage the files for a push.



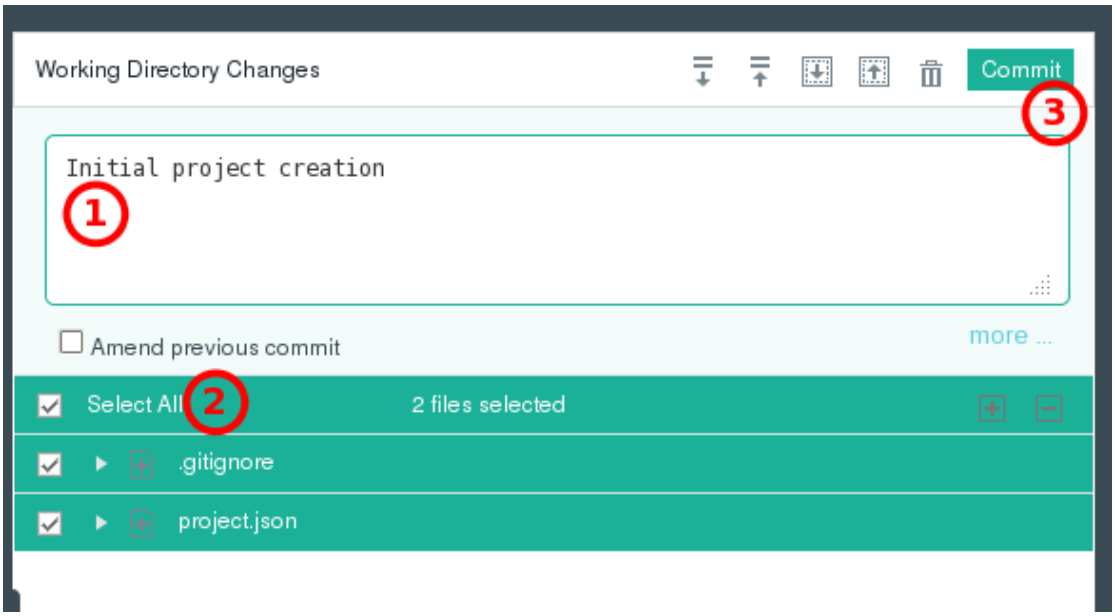
The process of forking a project causes changes to occur in the files. Notice that there are two files in the right pane. Click each one to get a view of what has changed.



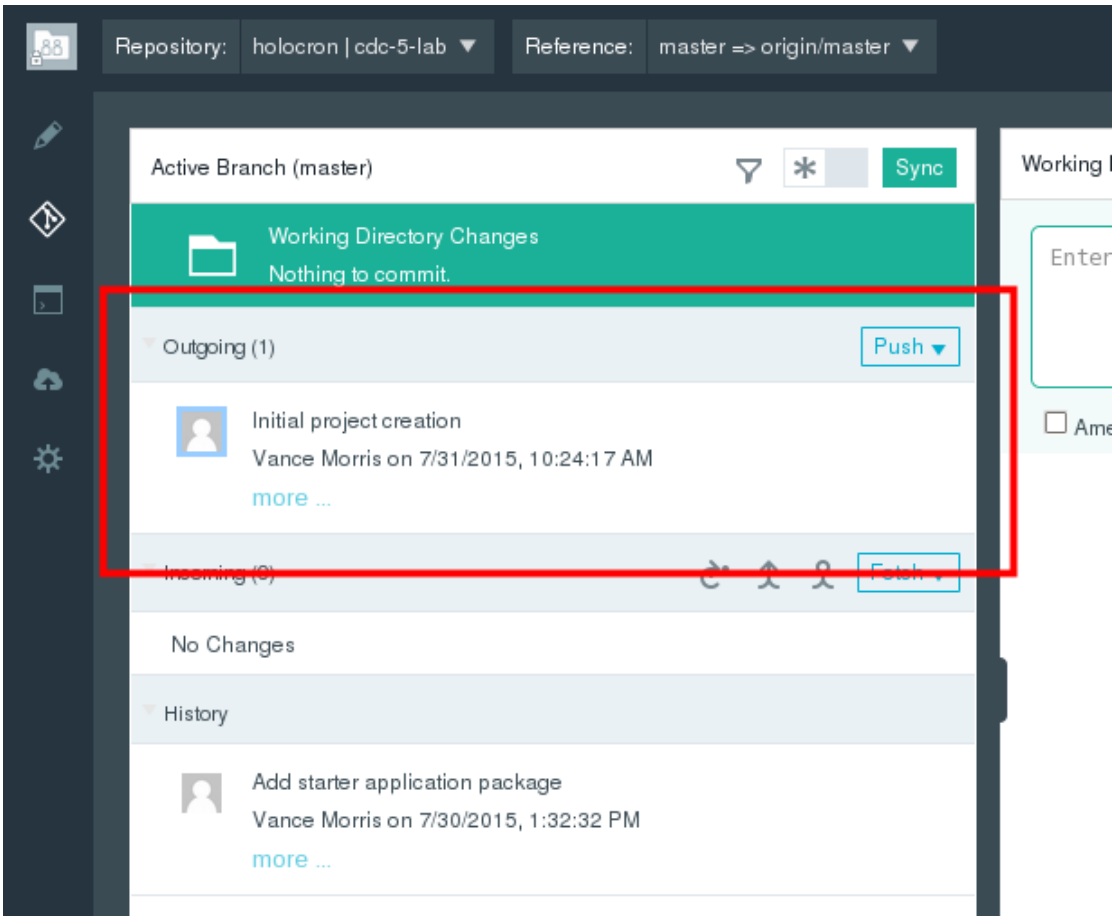
The .gitignore is a file used by Git to list all the files and directories in the project that should be ignored by Git. Items such as Bluemix launch configurations and build artifacts generally do not need to be tracked by Git and should not be committed to the master branch.

The project.json is a file used by DevOps Services to identify your project. It should remain unchanged and can be committed to the master branch.

1. Enter a commit message, such as `Initial project creation`, select **Select All**, and then click **Commit** in the top right.

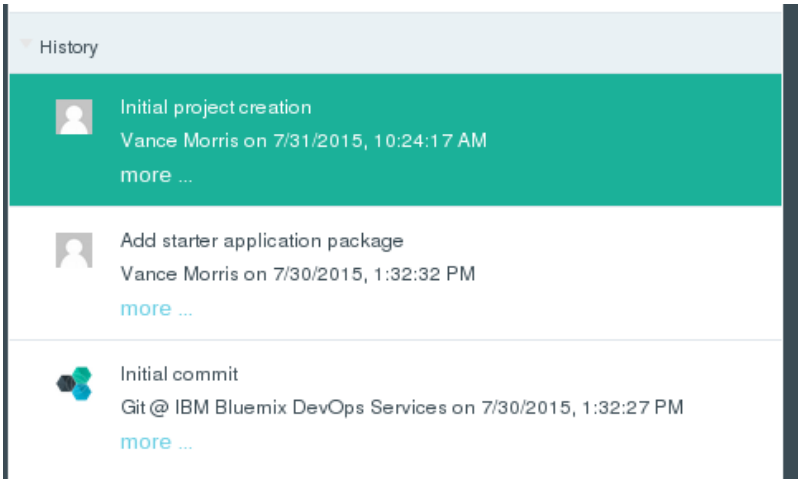


Notice that the working directory changes pane is now empty of any files, and a new entry is made in the Outgoing section of the left pane. This is called "staging" a commit.



The master branch is still unchanged and will remain so until you push the outgoing commit. If there were any incoming changes from other branches, you might need to merge them together with your outgoing changes, and you will have the opportunity to do so before the push completed.

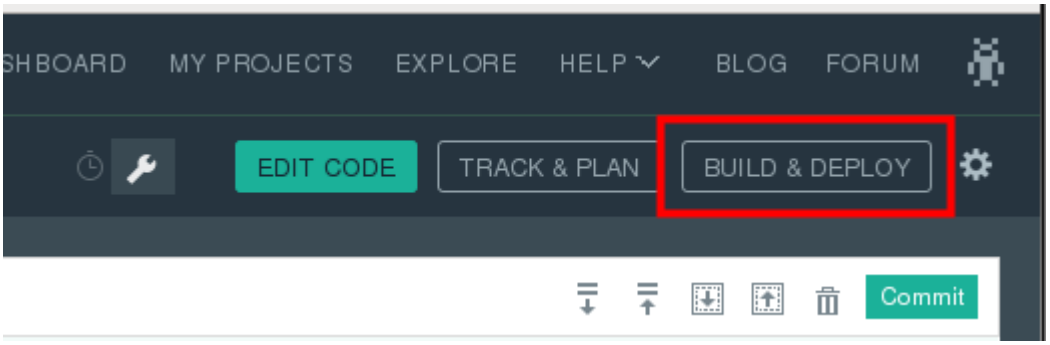
2. Click **Push** and note that the History is updated to include your commit. Any other branches will be able to pull this commit and its changes into their local working copy.



Exercise 5.4.3: Verifying the integrity of code delivered to the repository with a build

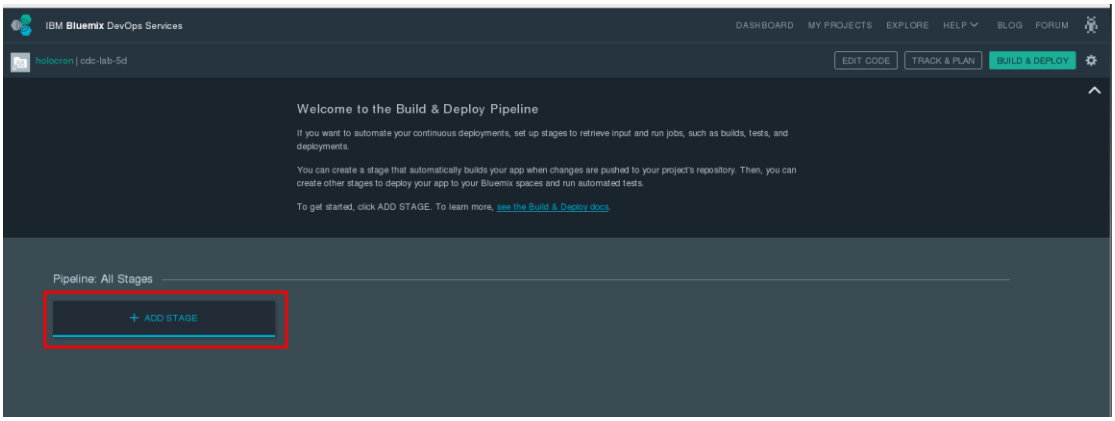
Next we will configure DevOps Services to perform a build of the code and have the build process trigger a successful push to the master branch. The builder will watch the master branch for any new pushes, and when detected, it will automatically build the project into a deployable package.

1. Click **BUILD & DEPLOY** in the top right.



The build and deploy pipeline view gives a high level view of the various stages that are configured in the pipeline.

2. Click **ADD STAGE**.

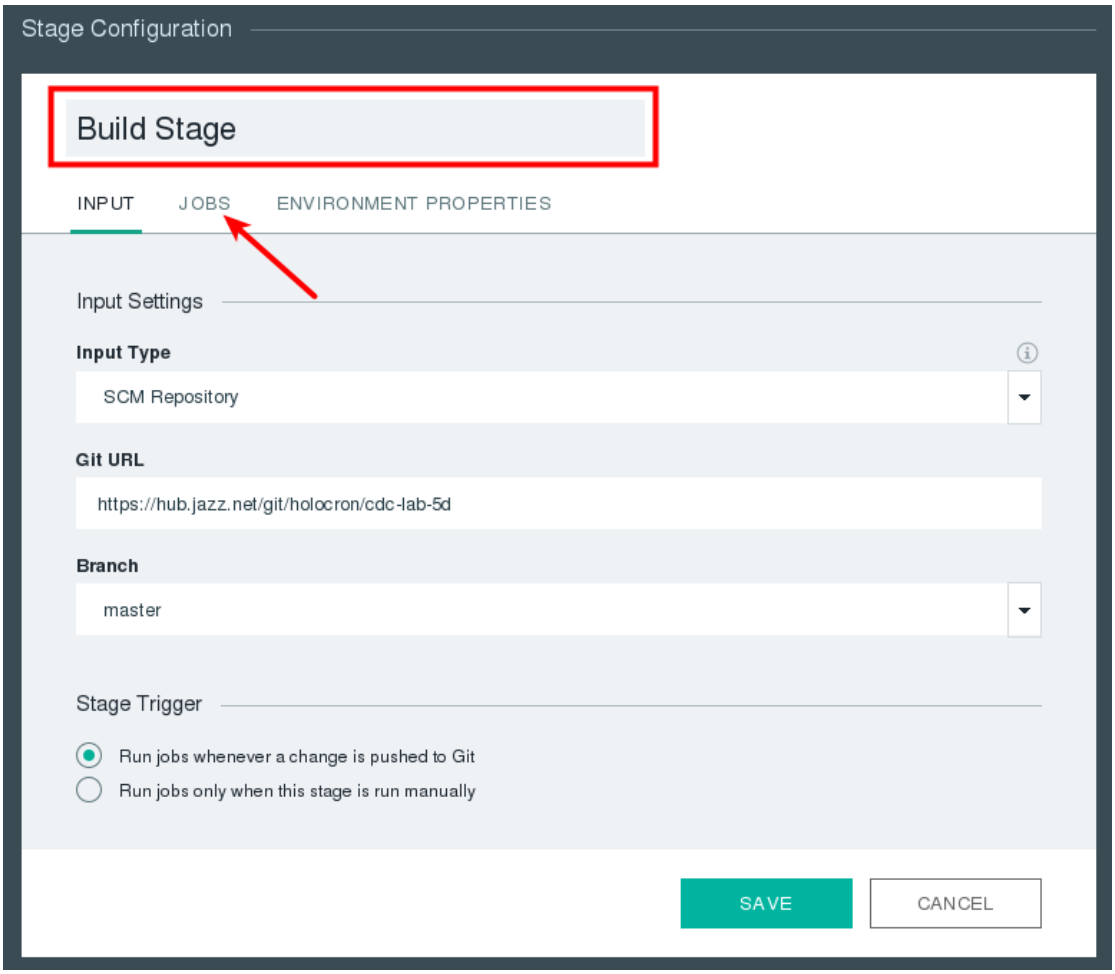


A stage consists of a series of jobs that run in sequence and the input for those jobs. Stages can be configured to run automatically based on triggers.

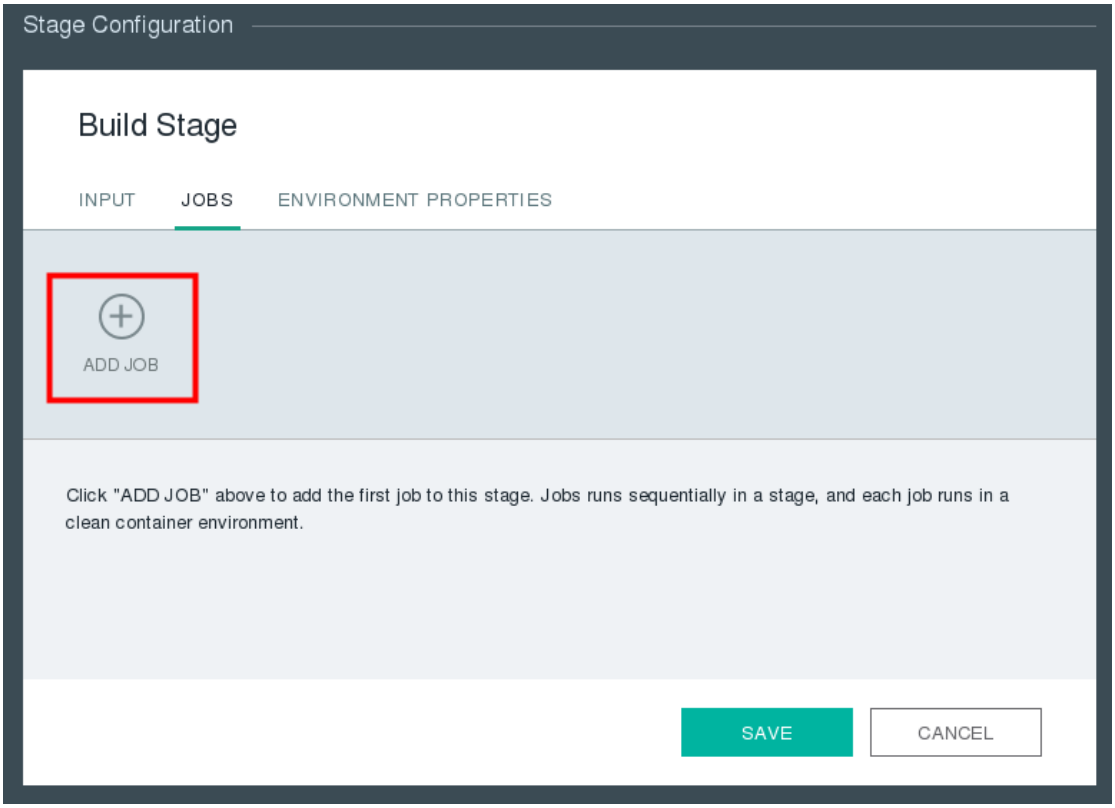
The input can be either the project SCM repository or build artifacts from a preceding stage. Input applies to all jobs in the stage. When a stage is run, the input is fetched. The files are placed in the working directory before each job starts.

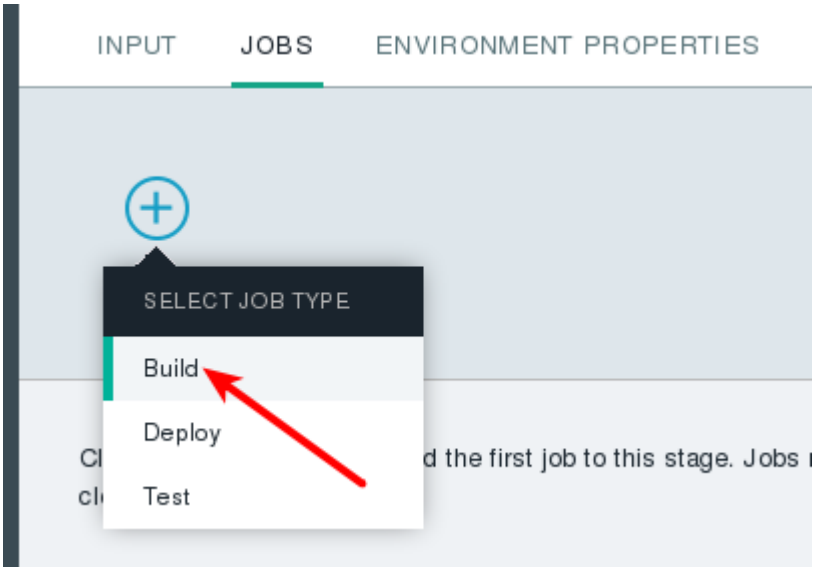
Jobs perform the work for a stage, such as building, deploying, and testing. The jobs in a stage run sequentially, and each job runs in a clean container environment. Files do not persist across job executions, so any dependencies required by the jobs must be provided in the stage input or installed as part of the job.

3. Give the stage a name to identify it. Then, click the **JOBS** tab. The input for the stage is the SCM repository that contains the master Git branch of your project. Also, note that the default has the stage run whenever any change is pushed to Git.

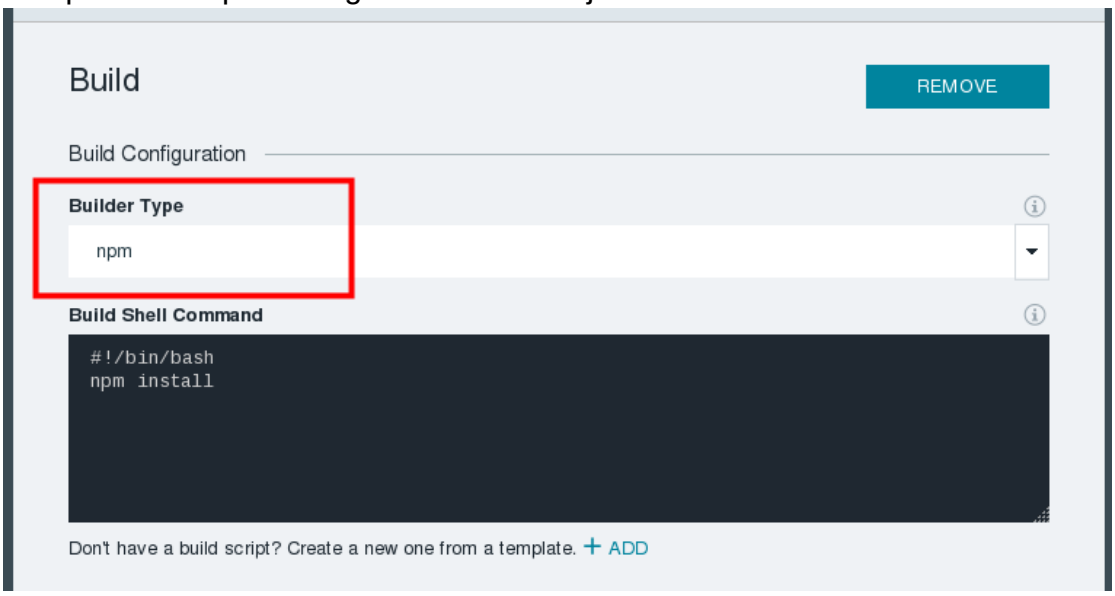


4. In the **JOBS** tab, click the **+** sign to add a job. Then, select **Build**.

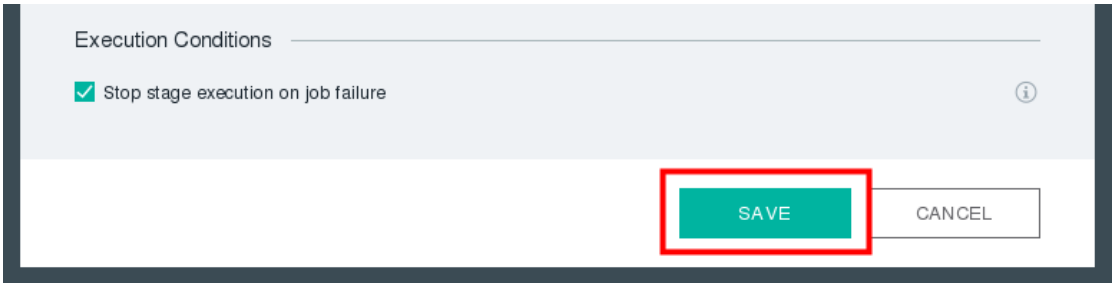




5. Under **Builder Type**, select **npm**. Many builder types are available to facilitate your project's needs, and other configuration options include a working directory, build archive directory, and an option to stop the stage execution if a job fails.



6. Click **SAVE**.

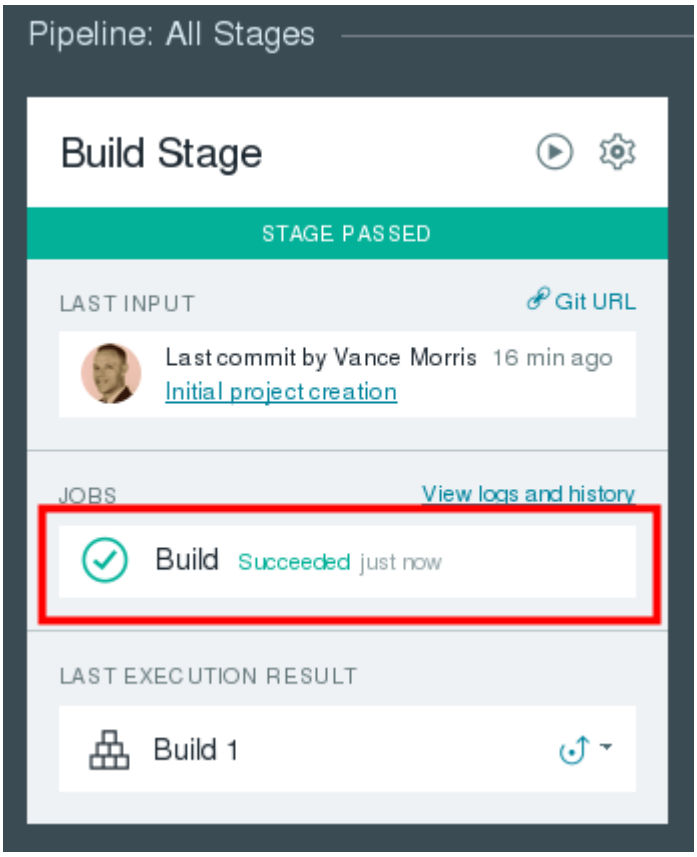


You are automatically returned to the build and deploy pipeline overview and the new stage is displayed.

7. Start a build automatically by clicking the **Play** button within the stage.

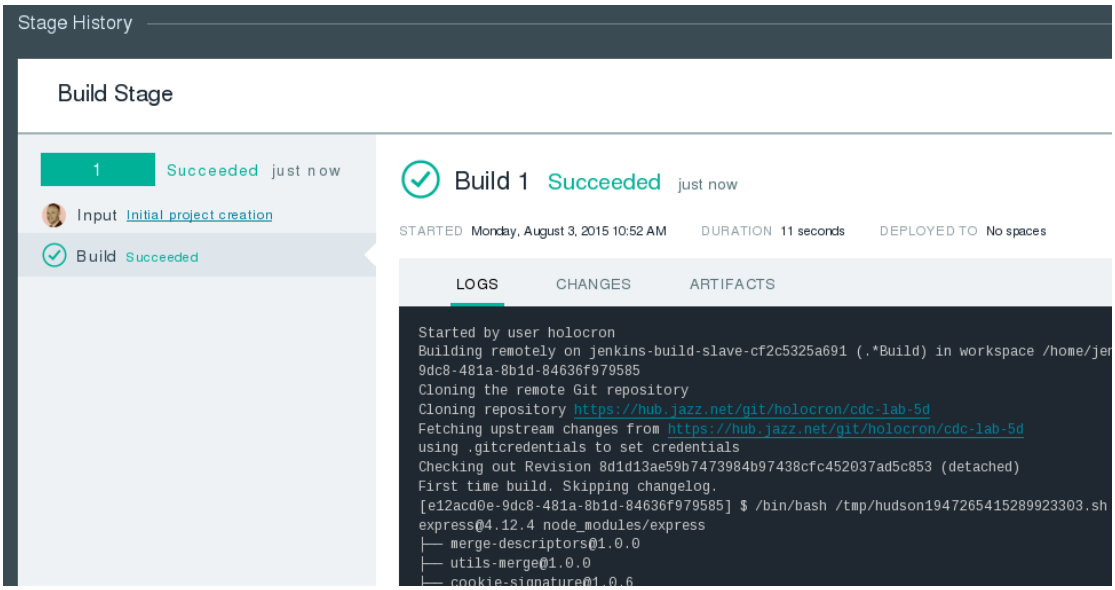
The stage starts running, and the build job status changes from Pending, to Queued, to Running, to Succeeded.

8. To view the logs from the build, click the job status message.



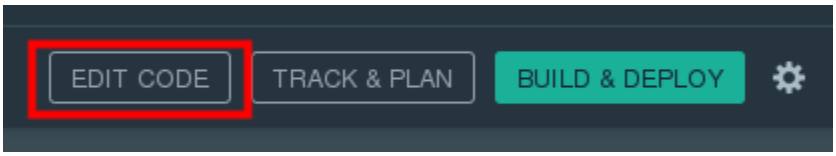
9. From the Stage History view, you can see the history of all previous executions of this stage, and download any artifacts generated as a result of a particular stage execution.

Exercise 5.4: Managing source code for projects

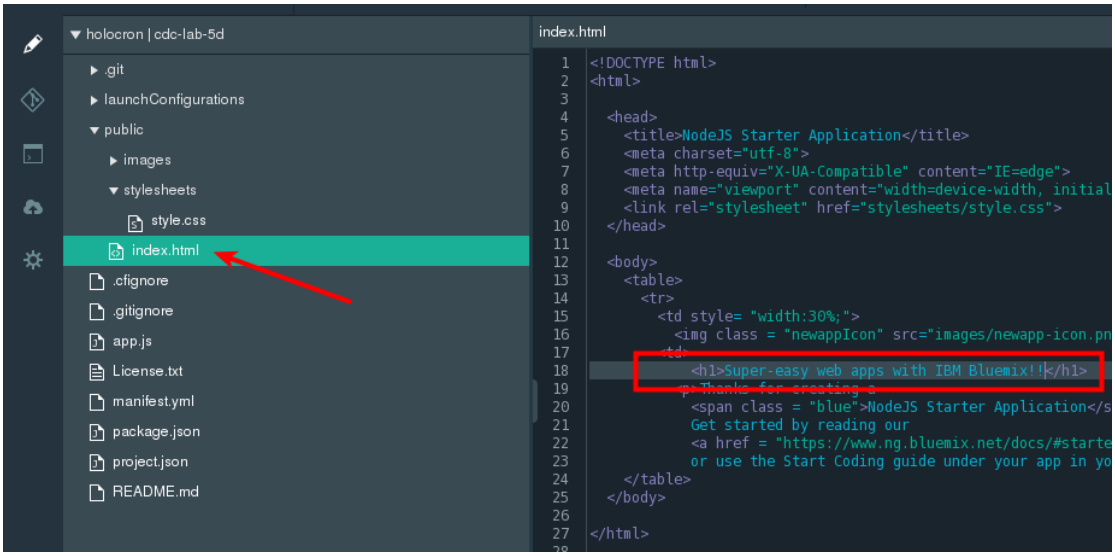


Exercise 5.4.4: Triggering the build stage by pushing to the Git repository

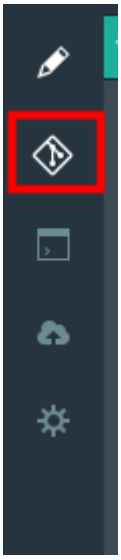
1. Return to the Web GUI editor by clicking **EDIT CODE**.



2. Alter the project by opening the public/index.html file and changing the <h1> tag to something new.



3. Switch to the Git repository view by clicking the **Git** icon in the left column.

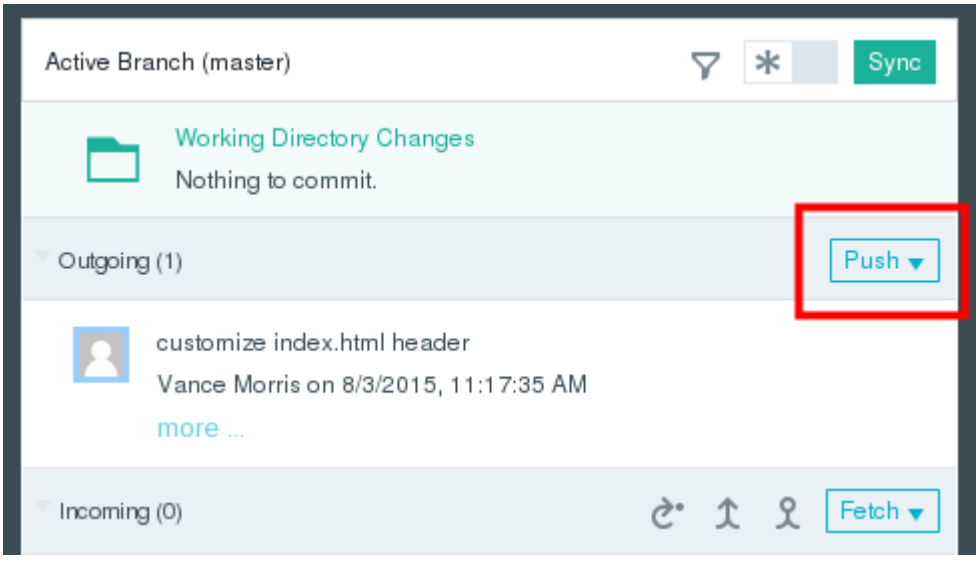


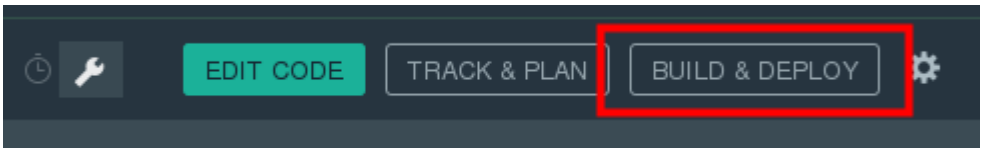
Note that the SCM detected a change in the working directory.

4. Enter an informative commit message, select the check box next to **index.html**, and then click **COMMIT**. The outgoing commit is now staged, and ready to push.

5. Immediately after pushing the commit, switch over to the **BUILD AND DEPLOY** view. You will need to be quick to see the builder stage automatically start, but if you miss it, you can always view the details of the build in the history view.

6. Click **Push** and then immediately click **BUILD AND DEPLOY**.





Observe the automatic execution of the build stage.