

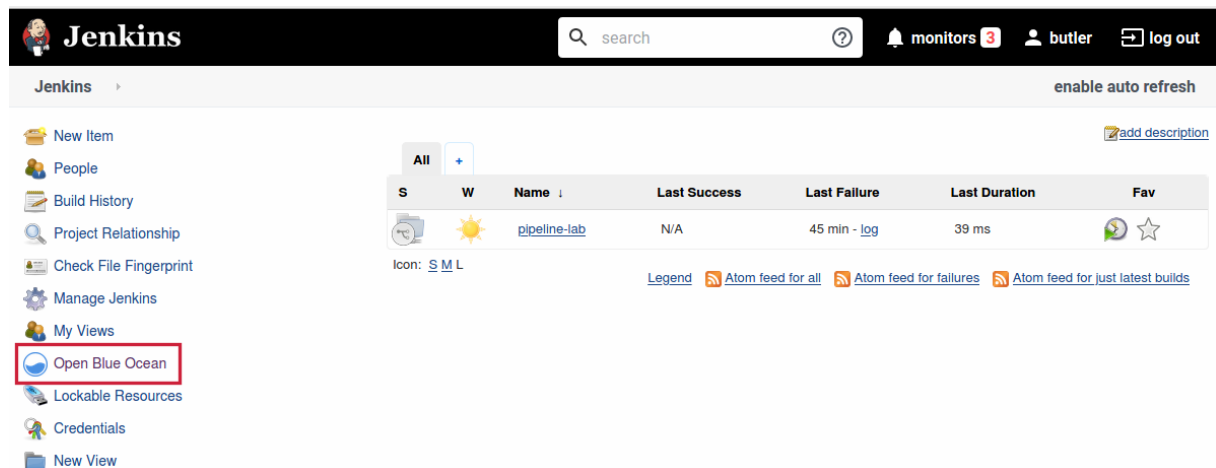
Lab: Create and edit an example pipeline

First pipeline

Our first project is a skeletal pipeline that builds, tests and deploys the software. We will use Blue Ocean to create and run that pipeline, then save it to the SCM repository.

Initialize Blue Ocean

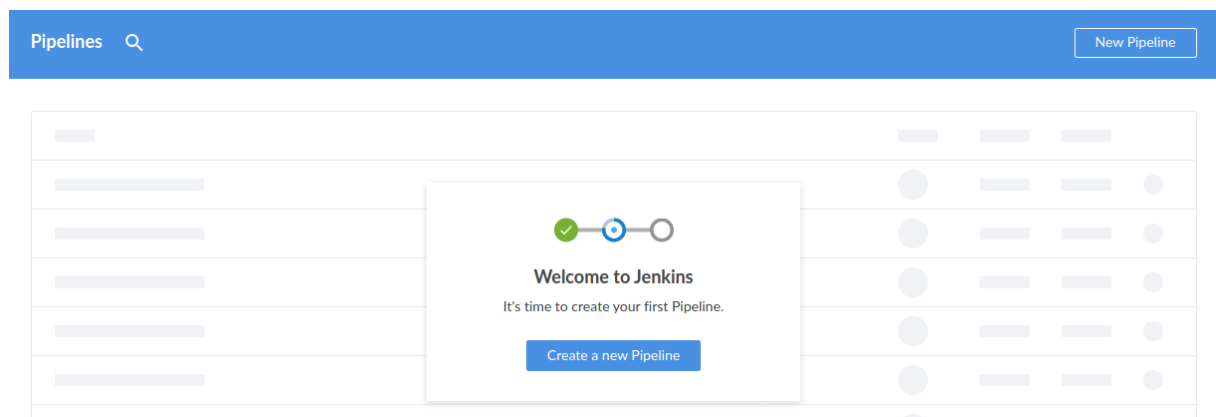
When you open Jenkins, the console is displayed:



Click on **Open Blue Ocean** in the left sidebar of the Jenkins console

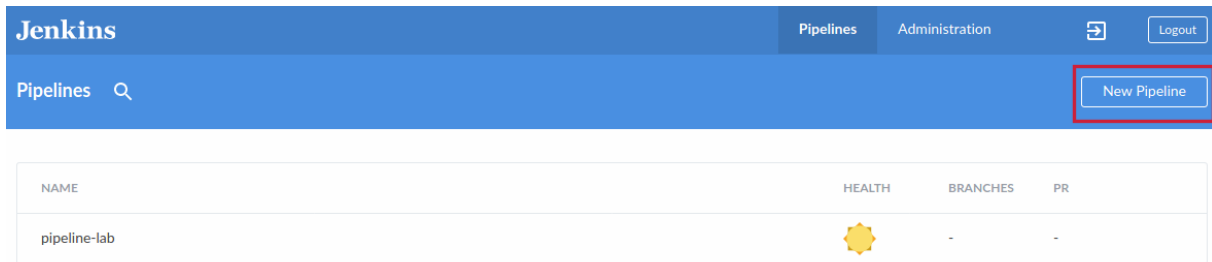
We will come back to the Web UI as we go through this class. You can do lots of things from this page but we are going to do most of our initial work using the Blue Ocean development environment.

The following screen appears the first time you open Blue Ocean:



Click **Create a new Pipeline**.

You can also start a new Pipeline by clicking the **New Pipeline** box on the Blue Ocean Navigation Bar at the top of the display:



Before we can start creating a Pipeline, Blue Ocean must be registered with the SCM repository. You are automatically put into the Blue Ocean screen that enables you to do this.

Register with Source Code Management

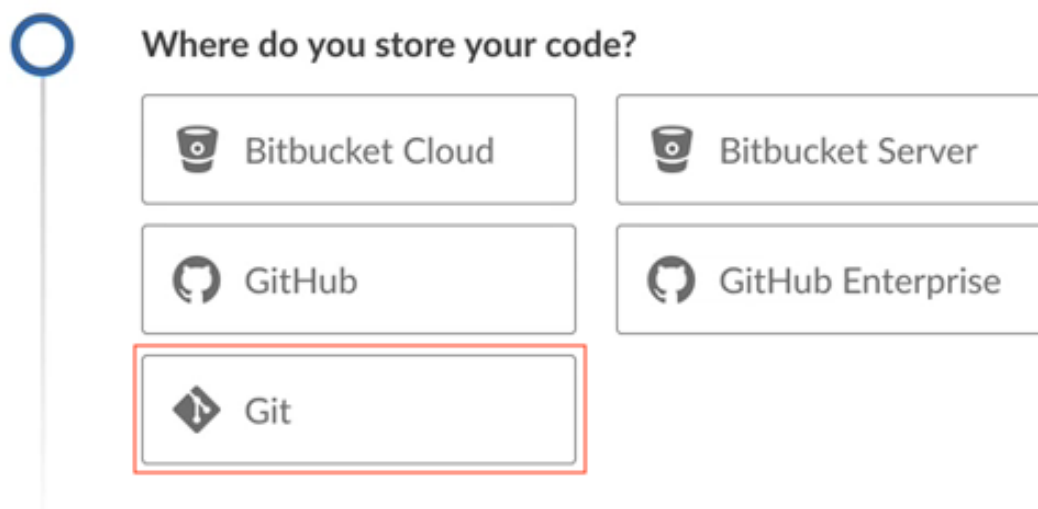
Before you can create a Pipeline, you must register Blue Ocean with the Source Code Management repository.

Blue Ocean provides tools to simplify this process. Your lab is set up to use Git, with Gitea as the host. This gives you a local advanced Git server with a web interface to browse repositories, authenticate, do pull requests and reviews.

Configure Blue Ocean for your SCM

The first time you create a Pipeline, you must configure Blue Ocean to work with your Source Code Management system.

Select **Git**.



Jenkins can work with almost any Source Code Management system. An advanced Git server such as Bitbucket or GitHub is required for pull/merge request integration in Jenkins, which validates the Pipeline code by Pull Request. Examples in this course use Gitea to host

a Git SCM so you can use standard Git commands without having any external service access

For your production environment, we strongly recommend using your own advanced Git server, or a hosted Git Service like Bitbucket or GitHub. The pull/merge request integration is an important feature; we will discuss this more later.

Connect repository

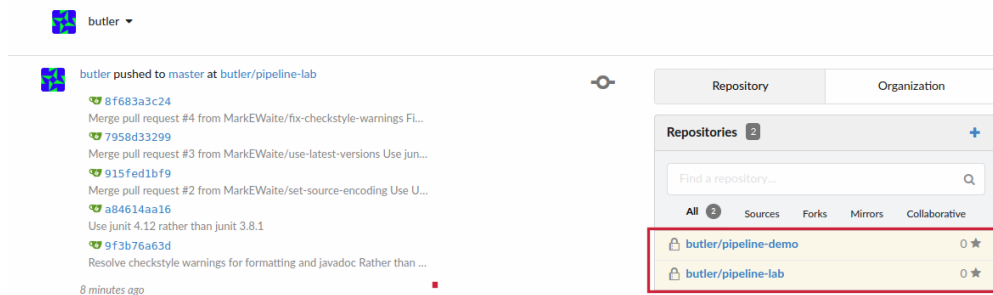
When you select **Git**, Blue Ocean provides a box where you can put the URL of the Git repository. The next slides show you how to get the URL that you can copy into this box.

The repository in your lab environment includes some executables and other resources that we will use for our Pipeline projects; in real life, you will have your repo that contains the code for your application and the software that is required to build it. As you build your Pipelines, you will be committing your Pipeline back to this same repo.

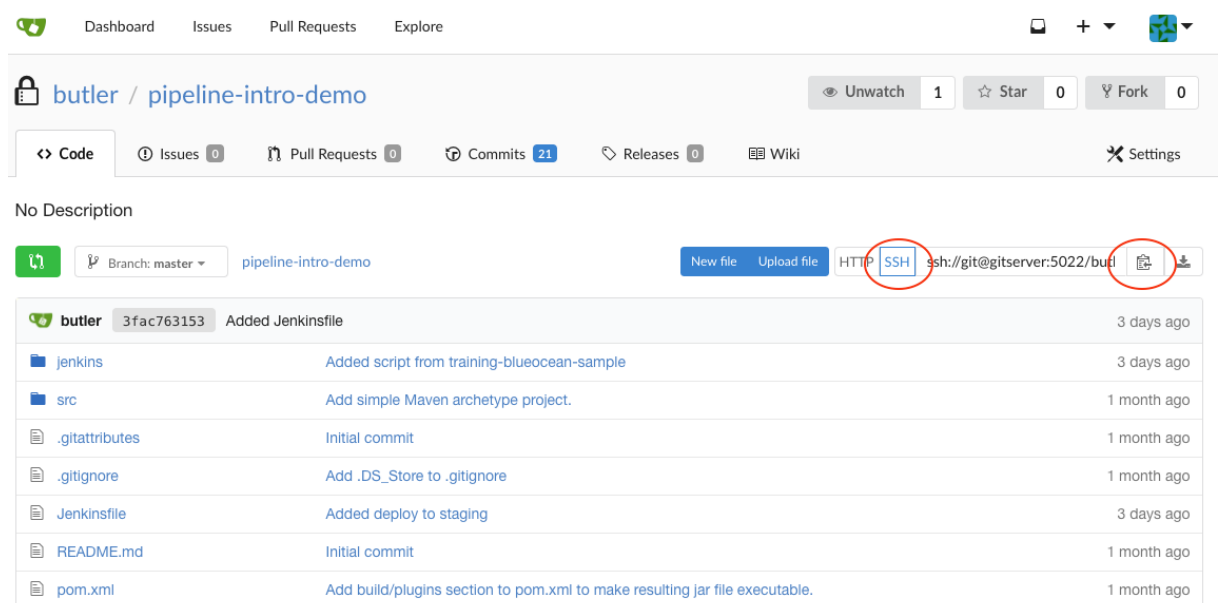
Pick repository

To provide the repository address (URL), access the Git repository page for the demo project:

1. On your **CloudBees Lab** home page, click on the **> Gitserver** link.
2. Click on **Sign In** in the upper right of the screen.
3. After you sign in, find the **Repository** box in the right portion of the display:



4. Click on the repository you want to use; in this case, choose **butler/pipeline-demo**
5. On the right end of the repository description bar, select the SSH button to display the SSH address.
6. Click on the copy icon at the right end of the box to copy the SSH address.



7. Go back to your Blue Ocean page and paste the URL into the box on the screen.

Autogenerated SSH key

When supplied with a valid URL, Jenkins detects that the repository is available for SSH access and auto-generates an SSH key for you. You must put the public part of this SSH key in the remote project. Gitea, the local Git Server in your lab instance, supports SSH out of the box, as do GitHub, Bitbucket, and GitLab.

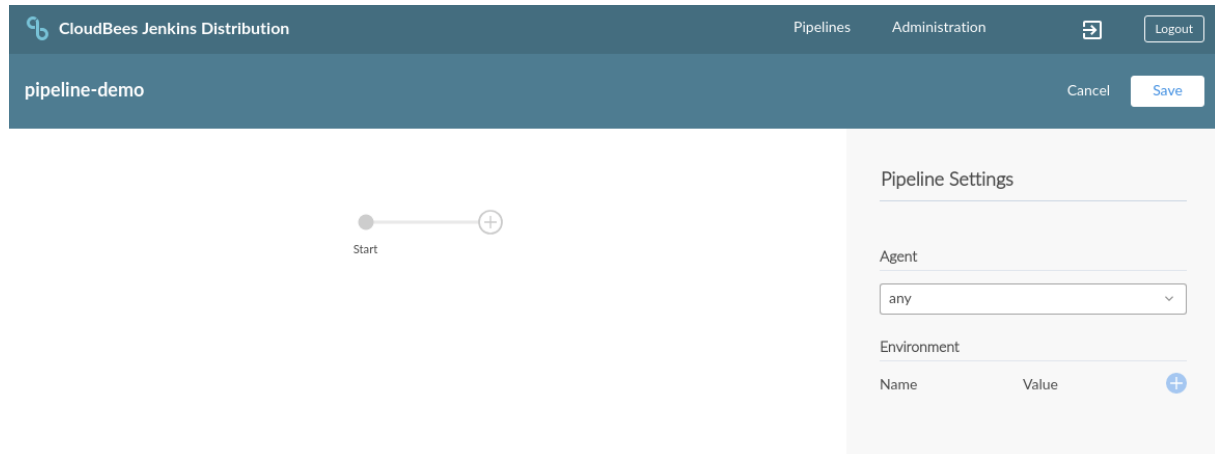
Blue Ocean displays the generated SSH key for the user `butler`. This key is used by Jenkins to authenticate on the Gitea Server so Blue Ocean can access the source code.

Your environment is already configured: the SSH key is already loaded in the Gitea Server. In Blue Ocean, you can click on the **Create Pipeline** button at the bottom of the screen.

Your access to the Git server is established. Blue Ocean is now connected to your Git server and we can create our first Pipeline. To get started, click on the **Create Pipeline** button at the bottom of the screen.

Creating your first (skeletal) Pipeline

When you click on the **Create Pipeline** button after registering your SSH key with Gitea, the following screen appears:



Notice the Navigation Bar at the top of the screen. The contents of this bar changes as appropriate for various tasks you run. The color of this bar changes to represent the status of your Pipeline.

Stages and steps

A Pipeline is divided into sections each of which is called a `stage`. A `stage` defines a chunk of work to be done; the stages are executed sequentially. Typically, a `stage` corresponds to one of the parts of the Continuous Delivery process such as build, unit test, regression test, or deploy. You can give a stage any name that is meaningful for your application.

A `stage` includes `steps`, which define actual tasks such as executing a script or program.

Any non-setup work should occur within a `stage` block. The Pipeline code itself executes on the Jenkins Master node. Most of the code inside `stage` blocks executes on agent nodes.

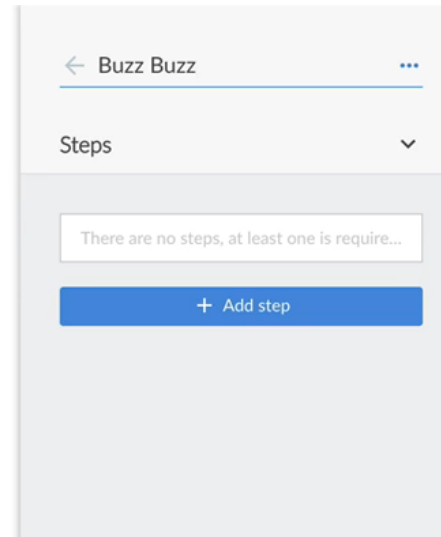
Add Buzz Buzz stage

First, we will create a `stage` called **Buzz Buzz**.

The first screen you see shows you a Pipeline, represented by a circle named **Start** and a circle with a + in it; A note at the bottom of the screen says that you need a stage.

- Click on the circle with the + sign to create a `stage`. Notice that the visualization for the Pipeline changes to include a blue circle (indicating that this stage is selected).
- The right frame of the screen displays a space where you can name this `stage`. Type in the name **Buzz Buzz** and notice that your Pipeline visualization now displays the name of the stage.

- Each stage executes on a specified `agent`. You can assign a single agent to use for all stages in the Pipeline but usually you want to specify a specific agent that has the operating system, JDK version, and any tools you need. We will discuss this more later. For now, use the agent box to set the **agent** to **any**. This says that this `stage` can run in any environment.
- You can also define environment variables here but we will not do that right now.



Notice that, if you click away from the left frame, Blue Ocean flags an error, both with a red triangle on the visual for the Pipeline and the **At least one step is required** message in the box in the left frame.

We could just delete this stage and start over. Hover the cursor over the three dots (`...`) in the upper right of the right frame to see the **Delete** box. Were you to click that box, the stage would disappear from the graphic and the **A stage is required** message would be displayed at the bottom of the screen. Blue Ocean requires that at least one stage be defined in order to save the Pipeline

In our case, what we have done is correct but we need to add a step to the stage.

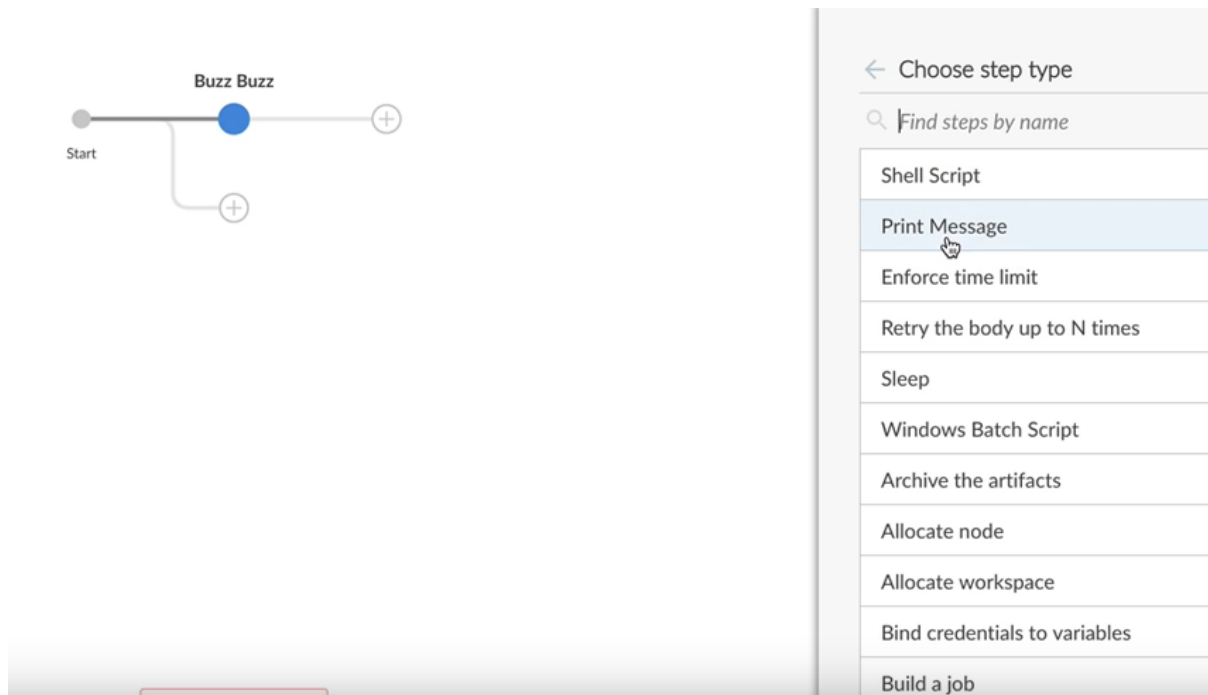
Add step to stage

A `step` specifies an executable to run to accomplish a task. We will add a very simple step to just print a message. To do this:

- Click on + **Add step** in the right frame. Blue Ocean lists the type of steps you can use.
- For this skeletal pipeline, choose **Print message** from the list.
- A box is displayed; if you click away from this box, Blue Ocean tells you that you need to supply some text here.
- In the box that is displayed, type:

Bees Buzz!

We now have a valid Pipeline That runs on **agent any** and prints the `Bees Buzz` message when it runs.



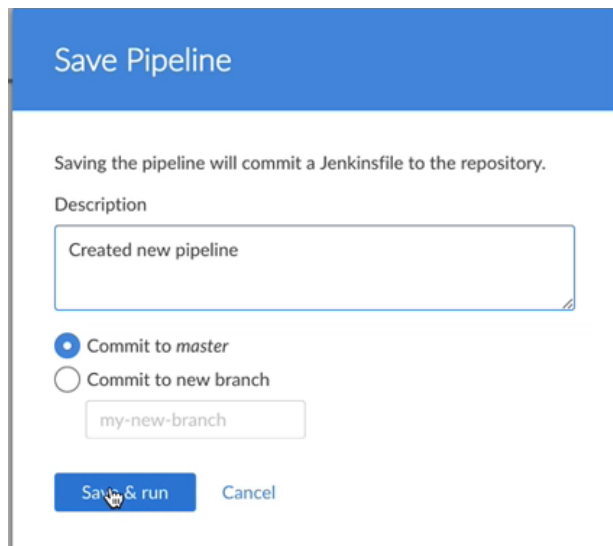
Now we can save and run the Pipeline.

Save Pipeline

To save and run our Pipeline:

- Click the **Save** button at the top right.
- In the **Save Pipeline** dialog box, type a description of what changed in the **Description** box: **Created new pipeline**
 - This is the `commit` message, if you who are familiar with Git.
 - Give a meaningful description of your work, including perhaps an explanation of why you did it.
 - You can put a brief (1 line, 80 characters) description on the first line, followed by a blank line and then as many lines as necessary to fully explain the change, just as you would when making a regular Git commit.
- Leave the default **Commit to master** set; we will discuss this more later.
- Click on **Save and Run**.

Notice that the screen seems to sit there, but, if you look closely, you see a grey progress bar moving across the top of the screen. This shows progress for doing the actual commit to Git.



Save Pipeline

Saving the pipeline will commit a Jenkinsfile to the repository.

Description

Created new pipeline

☒ Commit to master


☐ Commit to new branch

my-new-branch

Save & run Cancel

Watch Pipeline run


Once the commit to Git completes, you see the progress of the Pipeline as it runs. In this case, the Pipeline runs so fast that it completes almost immediately.




CloudBees Jenkins Distribution


Pipelines


Administration



Logout

 pipeline-demo






Activity

Branches

Pull Requests

STATUS	RUN	COMMIT	BRANCH	MESSAGE	DURATION	COMPLETED
	1	232ca79	master	Branch indexing	10s	a minute ago

- The green circle with a check in it indicates that the Pipeline ran successfully. For Pipelines that take longer to run, you will see a swirling blue circle on the left while it is running.
- You see that this is the first time to run this Pipeline. You see the Git commit number and the branch to which this edit was written.
 - Because this is the first time this Pipeline was committed, it shows **Branch indexing** as the message; in subsequent runs, you will see your commit message here.
- The display also shows how long it took to run the Pipeline and when it finished.
- Click on `Branch indexing` to see more details about the run.

View Pipeline Run Details

The Pipeline Run Details page shows a visual representation of the Pipeline, which currently consists of one stage, named **Buzz Buzz**:

✓ pipeline-demo 1

Branch: master [link](#) 11s No changes
Commit: 3922b95 10 minutes ago Branch indexing

Start Buzz Buzz End

Buzz Buzz - <1s [Restart Buzz Buzz](#) [link](#) [download](#)

✓	> Check out from version control	5s
✓	> Bees Buzz! - Print Message	<1s

- The green circle with the checkmark in it indicates that it ran successfully.
- Below this, it lists the output of each stage.
 - Click on **Check out from version control** to see the actual steps taken to commit this edit to Git.
 - Click on the **Buzz Buzz** stage to see details for the execution of the one step we put in this stage, which prints out **Bees Buzz!**

Note that the color of the header of this screen is significant:

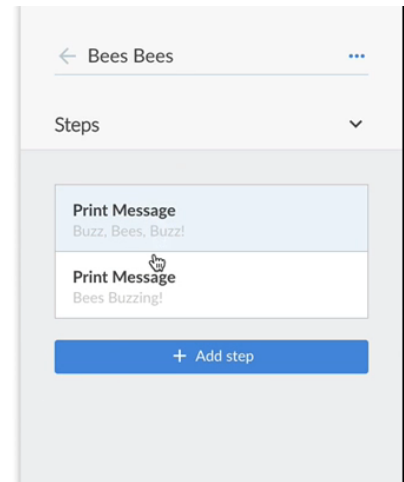
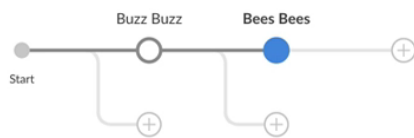
- blue indicates that this version of the pipeline is under development or running
- green indicates that the Pipeline has been run successfully.
- red indicates that the Pipeline run failed.
- grey indicates that the Pipeline run was interrupted before it completed

Let's look at some other information that is available:

- Click on **Changes**. This is the first time we committed this Pipeline so there are no changes.
- Click on **Tests**; we have not (yet) added any tests to the Pipeline so it tells us that no tests were run.
- Click on **Artifacts**. An *artifact* is a file produced as a result of a Jenkins build. We will discuss artifacts more in the next section. For now, you see that we only have one artifact, which is the `pipeline.log` file. This is a raw log of the Pipeline that is produced by every Pipeline run. It is not a **user-friendly** file but it does include detailed information that is sometimes helpful when you are debugging complex Pipelines.

Add Bees Bees stage

Now we will edit the Pipeline to add a second stage. To edit this Pipeline, either click the pencil icon in the header or hover the cursor at the right end of the status line on the status info screen for this Pipeline.



Note that the header of this screen is now blue to indicate that the Pipeline is being edited or processed.

To add another stage:

- Click on the circle to the right of the **Buzz Buzz** stage. Name this stage **Bees Bees**.
- Add a step of type **Print Message**
 - A box is displayed where you type the content of the message; type **Buzz, Bees, Buzz!**
- You can add more steps to this stage, of any kind, just by clicking on **Add step** and selecting the step you want from the list.
 - Add another **Print Message** step with the text **Bees Buzzing!**
 - Add another **Print Message** step with the text **Bees Buzzing Again**
 - Now let us delete that third step:
 - Select the message
 - Click on the three dots at the upper right
 - Click on **Delete**

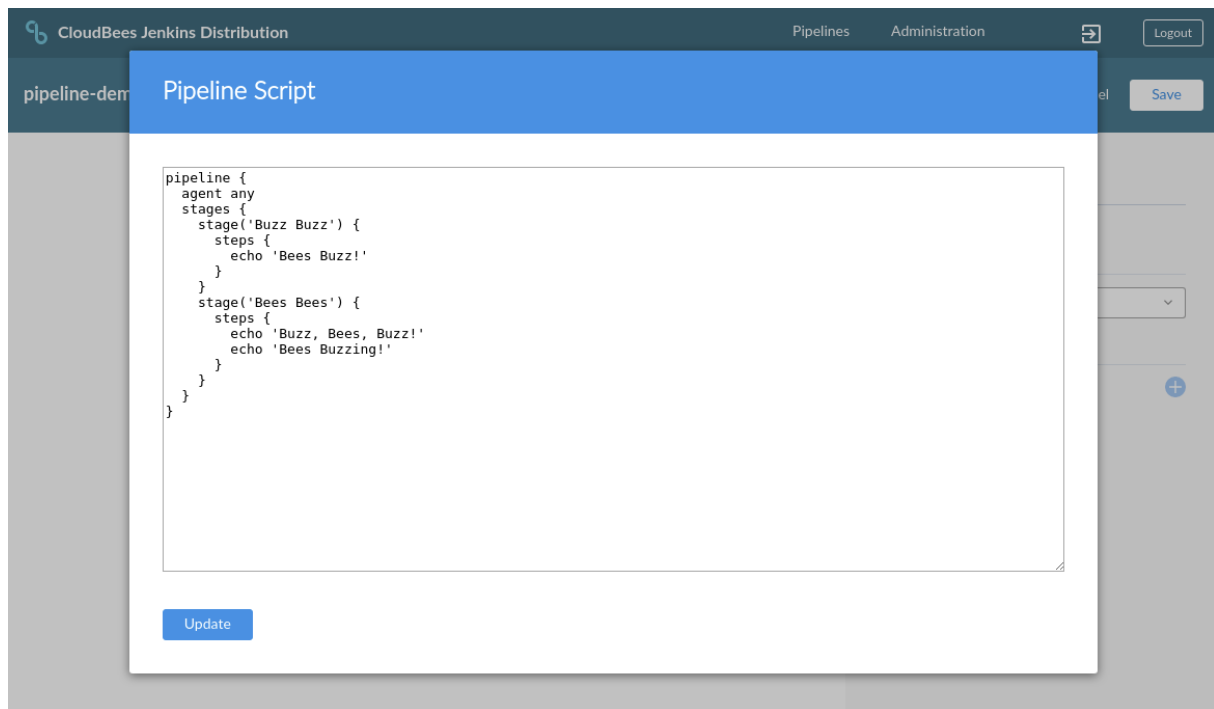
The Pipeline now contains the **Buzz Buzz** stage that has one step and the **Bees Bees** stage that has two steps.

Before we save this, let's look at the actual Jenkinsfile that defines the Pipeline.

[View Jenkinsfile](#)

Blue Ocean includes a Code Editor window in which you can view or modify the *Jenkinsfile*, which is your actual Pipeline (Pipeline-as-code). Later, when you learn to develop Pipelines without Blue Ocean, you can access and edit this same *Jenkinsfile* with the text editor of your choice.

To access the Code Editor, click ctrl-S (Linux or Windows) or cmd-S (macOS). This pops the edit window that displays an editable version of the current Jenkinsfile. Here you can clearly see the structure of your Declarative Pipeline:



Jenkinsfile structure

For Declarative Pipeline, the first line must be `pipeline` to open a pipeline block in which all the rest of the code is included. Note the following elements:

- **agent** — Specifies where the Pipeline or a specific **stage** executes
 - Declarative Pipeline requires a global declaration of **agent** as the first line in the `pipeline` block. It is possible to use different agents for different stages but the global `agent` statement is always required. For now, we are using `agent any`, meaning that this Pipeline can run on any available agent.
- **stage** — A conceptually distinct subset of the Pipeline, such as **MyBuild**, **MyTest**, or **MyDeploy**. Note the two `stages` that we have created, each of which contains `steps`.
 - Used to present Pipeline status/progress.
 - Stage labels should be significant for your application.
- **steps** — A series of distinct tasks inside a stage. You see here that the **Print Message** steps we created have been converted to standard `echo` commands.

Real world Jenkinsfiles are, of course, much more complex — they can also include Sections, Directives, Options and so forth — but will always have this same basic structure.

For full information about Pipeline syntax, see jenkins.io/doc/book/pipeline/syntax

Edit Jenkinsfile in the Code Editor

You can edit the Jenkinsfile directly in this Code Editor window; just make the changes then click the **Update** button.

For example, let's change the name of the second stage to be **Bees Bees Bees**:

- Click the **Update** button
- See that the UI now shows the new stage name
- Now go back to the Code window and set the name of the second stage back to **Bees Bees**
- Click the **Update** button, and see that the stage name is back to **Bees Bees**.

Save and run pipeline

Now we can save and run our revised Pipeline:

- For Description, type **Added Bees Bees Stage**
- Click **Save & Run**
- Watch the grey progress bar across the top of the screen

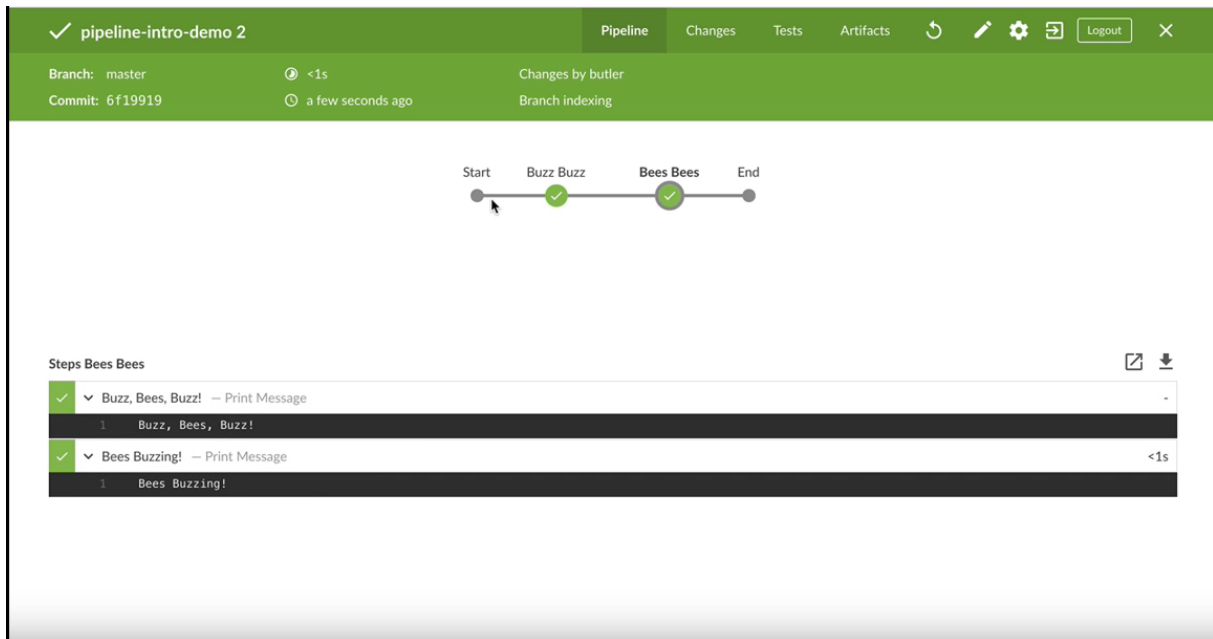
Watch Revised Pipeline Run

Now you see that we have two runs listed. Again, both run so quickly that they are completed by the time the display comes up; this will not always be the case! Notice that the **Description** you supplied for the revised Pipeline is now displayed under **Message**

CloudBees Jenkins Distribution							
pipeline-demo				Administration			
Activity				Branches		Pull Requests	
STATUS	RUN	COMMIT	BRANCH	MESSAGE		DURATION	COMPLETED
✓	2	625c7f4	master	Add Bees Buzz stage		2s	2 minutes ago
✓	1	232ca79	master	Branch indexing		10s	an hour ago

[View Revised Pipeline run details](#)

Our Pipeline now has two stages:



- Click on the **Buzz Buzz** stage. Like before, it has an SCM step that displays details of the commit to Git
- Click on the Bees Bees stage. You see that it has the two steps we defined.
- Click on each step to see that the output of the stage is displayed.

So now you have created a skeletal Pipeline. In the next section, we will create a simple Pipeline that does some more interesting things.

Do it your self lab

In this exercise you will:

- Create a new Pipeline job.
- Create and run a new Pipeline in the Blue Ocean Visual Editor.
- Review Pipeline run details.
- Edit a Pipeline.
- Use the Blue Ocean Code Editor to update your Declarative Pipeline code.

Task: Create a new Pipeline job

In the first lab exercise, you will create a simple pipeline with Declarative Syntax. This pipeline has three distinct stages and a post step that always runs.

If you are using CloudBees University Standard, start your lab VM.

If you are using CloudBees University Premium, follow these instructions to start your lab environment:

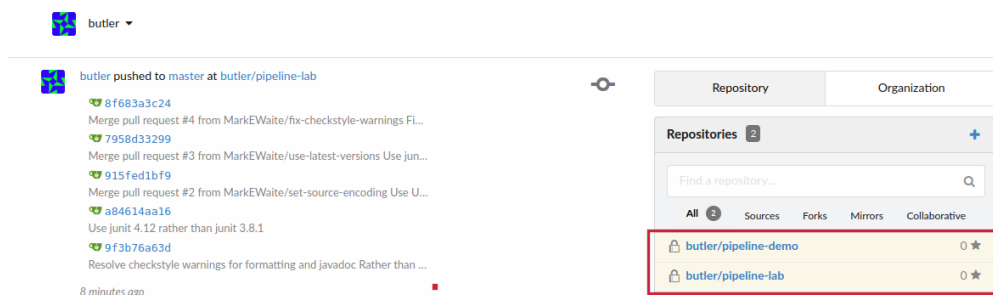
1. From your Pipeline Essentials Lab VMs, choose **Pipeline Essentials UI resources**.
2. Double-click the Google Chrome icon.
3. From the Welcome to your CloudBees Lab Instance home page, select the **Jenkins LTS (Long-Term Support)** link.

The username *and* password for all of your training environments is `butler`.

4. From the Jenkins Dashboard, select **Open Blue Ocean** in the left navigation bar.

Task: Create a new Pipeline

1. Return to your CloudBees Lab Instance home page and then select the **Gitserver** link.
2. If you are not already signed in, select **Sign In** in the upper-right corner of the screen.
3. After you sign in, find the Repository box in the right portion of the display.



4. Select the repository you want to use. In this case, choose `butler/pipeline-lab`.
5. On the right end of the repository description bar, select **SSH** to display the SSH address.

6. Select the **Copy** icon at the right end of the box to copy the SSH address:
`ssh://git@gitserver:5022/butler/pipeline-lab.git.`
7. Return to Blue Ocean and select **New Pipeline**.
8. Choose **Git** when asked where to store your code.
9. Paste the SSH address for `pipeline-lab`.
10. Select **Create Pipeline**. The `pipeline-lab` Pipeline editing page appears.
11. Create three new stages:
 - a. Add a Fluffy Build, a Fluffy Test, and a Fluffy Deploy stage. Notice that these stages show errors because they do not have steps yet.
 - b. Add a **Print Message** step to each stage with the text `Placeholder` for the message.
12. Save the pipeline:
 - a. In the **Save** dialog, provide the description `Add first pipeline`.
 - b. Select **Commit to master**.
 - c. Select **Save & run** to save to the default branch (`master` branch).
13. Review the pipeline run details.

Task: Edit a Pipeline

Edit the pipeline you just created:

1. Select the edit icon.
2. Remove the **Print Message** step from the Fluffy Test stage and add two steps:
 - a. **Shell Script**: `sleep 5`
 - b. **Shell Script**: `echo Success!`
3. In the Fluffy Build stage, add the following step:
 - a. **Shell Script**: `echo Another Placeholder`
4. Save the pipeline. In the **Save** dialog, provide the description `Update first pipeline` and save to the default branch (`master` branch).

Task: Use the Blue Ocean Code Editor

1. Select the `Update first pipeline` build and then select the edit icon.
2. To edit the pipeline, open the Blue Ocean Code Editor with `Ctrl+S` or `Cmd+S`.
3. Change the string in the Fluffy Build stage from `echo Another Placeholder` to `echo Edited Placeholder`.
4. Select **Update**.
5. View the steps in your pipeline to ensure the text has changed.
6. Save the pipeline. In the **Save** dialog, provide the description `Update first pipeline` and save to the default branch (`master` branch).

Solution

Jenkinsfile (final)

```
pipeline {
  agent any
  stages {
    stage('Fluffy Build') {
      steps {
        echo 'Placeholder'
        sh 'echo Edited Placeholder.'
      }
    }
    stage('Fluffy Test') {
      steps {
        sh 'sleep 5'
        sh 'echo Success!'
      }
    }
    stage('Fluffy Deploy') {
      steps {
        echo 'Placeholder'
      }
    }
  }
}
```