Getting Started with Jenkins X

Class Labs: Revision 1.5 - 6/22/20

Brent Laster

Important Note: Prior to starting with this document, you should have the ova file for the class (the virtual machine already loaded into Virtual Box and have ensured that it is startable. See the setup doc jxi-setup.pdf in http://github.com/brentlaster/safaridocs for instructions and other things to be aware of. You should also have a GitHub account.

Lab 1: Creating a Cluster

Purpose: In this lab, we'll see how to create a Kubernetes cluster starting without one on our minikube system.

1. First, we'll create a cluster. Since the Jenkins X create cluster option has been deprecated and we don't have a terraform or other means, we'll use the recommended minikube start command. Enter the command below in a terminal emulator to create the cluster. Note there is a space between "—vm-driver" and "none".

```
sudo minikube start --memory 12000 --cpus 5 --disk-size 35GB --vm-driver
none  --bootstrapper=kubeadm --kubernetes-version v1.15.0
```

2. This should then create the cluster. **It will take a few minutes to run.** There are some permissions that need to be changed and some performance improvement we can get via a tweak to the DNS pods. Run the command in the home directory:

```
sudo ./fixdns.sh
```

3. Now, let's start up the Kubernetes dashboard and have a look around at our cluster. Do this by running the command below.

```
minikube dashboard &
```

4. If you want, you can explore the cluster by looking at the various elements selectable via the dashboard.

Lab 2: Installing Jenkins X on the Cluster

Purpose: In this lab, we'll see how to install Jenkins X (jx) on the cluster we created in lab 1.

1. To install Jenkins X on the cluster, we'll use the jx install command. Like the jx create cluster command, we have some options to select on this. Start by invoking the command below. (The --default-admin-password option is to avoid having a long, difficult system-generated password for our Jenkins instance). (If you are not back at the prompt in your terminal session, you can just Enter before putting in the command below.)

```
jx install --default-admin-password=admin
```

2. For this initial example, we'll leverage the "**Static Jenkins Server and Jenkinsfile**" implementation. At the "Select Jenkins Installation Type" prompt, arrow down and select that option and hit Enter. (This may run for a bit as it creates the jx namespace and sets the default namespace context to it.)

3. At the prompt for "Please enter the name you wish to use with git:", enter your name – usually in the form:

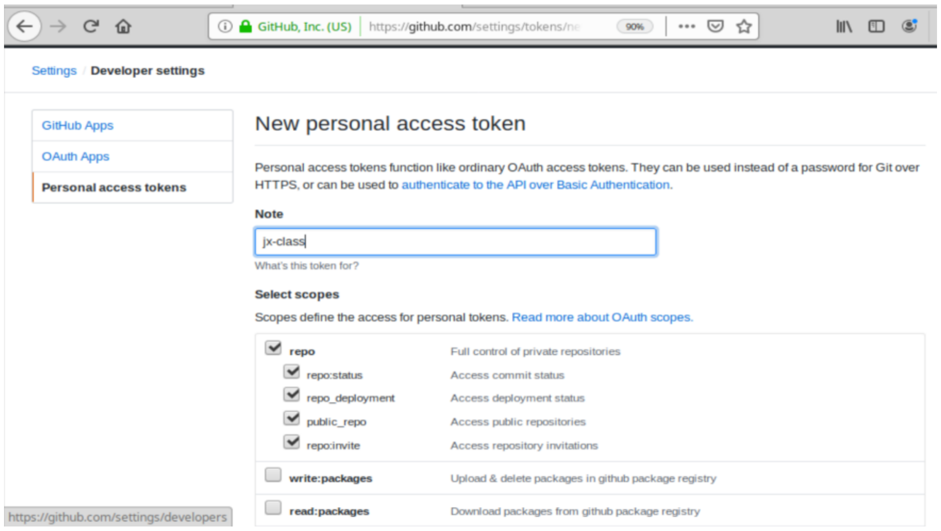
   First-name Last-name

4. At the prompt for "Please enter the email address you wish to use with git:", enter your email address.

5. At the prompt for "GitHub username:", enter your GitHub username.

6. You'll then need to supply a GitHub API token to have access to GitHub. jx will generate a URL for you. Right click on that URL and select "Open Link". (Don't worry if you see an error in the background from Firefox.)

7. Log in to GitHub and you'll be on the right screen to generate a token for access. In the Note field, you can just enter any kind of identifier/name, such as "jx-class".



GitHub, Inc. (US) https://github.com/settings/tokens/ne 90%

Settings / Developer settings

GitHub Apps
OAuth Apps
Personal access tokens

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

jx-class

What's this token for?

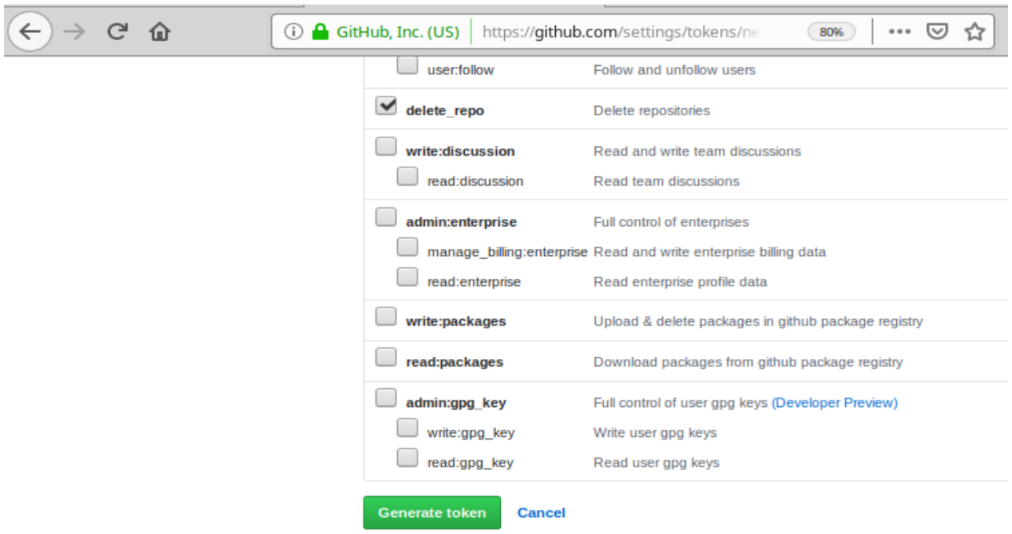
Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

✓ repo	Full control of private repositories
✓ repo:status	Access commit status
✓ repo_deployment	Access deployment status
✓ public_repo	Access public repositories
✓ repo:invite	Access repository invitations
☐ write:packages	Upload & delete packages in github package registry
☐ read:packages	Download packages from github package registry

https://github.com/settings/developers

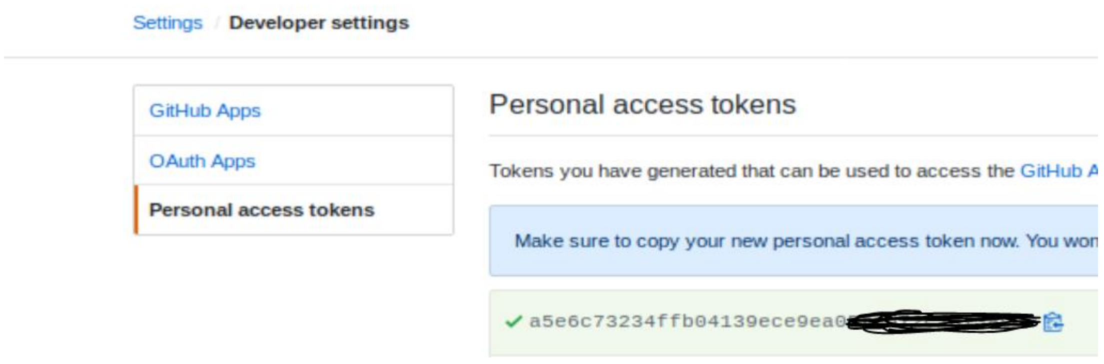
8. Scroll down to the bottom and click on the green "Generate token" button.



GitHub, Inc. (US) https://github.com/settings/tokens/ne 80%

☐ user:follow	Follow and unfollow users
✓ delete_repo	Delete repositories
☐ write:discussion	Read and write team discussions
☐ read:discussion	Read team discussions
☐ admin:enterprise	Full control of enterprises
☐ manage_billing:enterprise	Read and write enterprise billing data
☐ read:enterprise	Read enterprise profile data
☐ write:packages	Upload & delete packages in github package registry
☐ read:packages	Download packages from github package registry
☐ admin:gpg_key	Full control of user gpg keys (Developer Preview)
☐ write:gpg_key	Write user gpg keys
☐ read:gpg_key	Read user gpg keys

9. You'll then see your new personal access token displayed. Select it and copy it (or just click the icon on the end to copy).


Settings / **Developer settings**

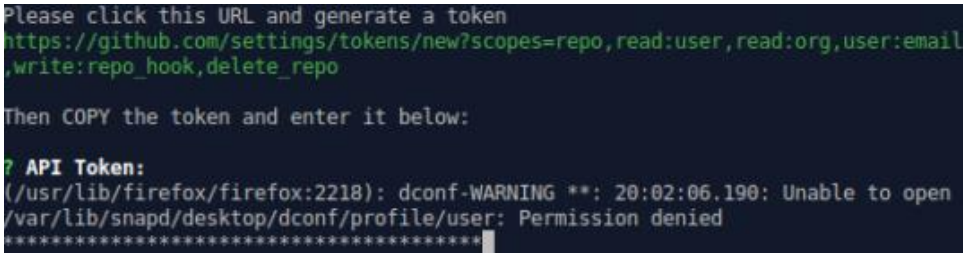
Personal access tokens

Tokens you have generated that can be used to access the GitHub A

Make sure to copy your new personal access token now. You won

✓ a5e6c73234ffb04139ece9ea0[redacted]

10. **Paste the copied token** (right-click and select Paste) back at the "API Token" prompt in the terminal session and **hit Enter**. See screenshot below. Paste it under the "Permission denied" line.


```
Please click this URL and generate a token
https://github.com/settings/tokens/new?scopes=repo,read:user,read:org,user:email
,write:repo_hook,delete_repo

Then COPY the token and enter it below:

? API Token:
(/usr/lib/firefox/firefox:2218): dconf-WARNING **: 20:02:06.190: Unable to open
/var/lib/snapd/desktop/dconf/profile/user: Permission denied
****************************************
```

(Note: You may want to save the token string in a file somewhere or email it to yourself as you won't be able to get to the actual string again.)

11. At the prompt for "Do you wish to use GitHub as the pipelines Git server: (Y/n), just hit Enter to select the default Yes answer.

12. You'll be prompted again for a GitHub API Token for a Git user for the pipeline processes (the jx bot). We can use the same token as before, so just paste it again and hit Enter.

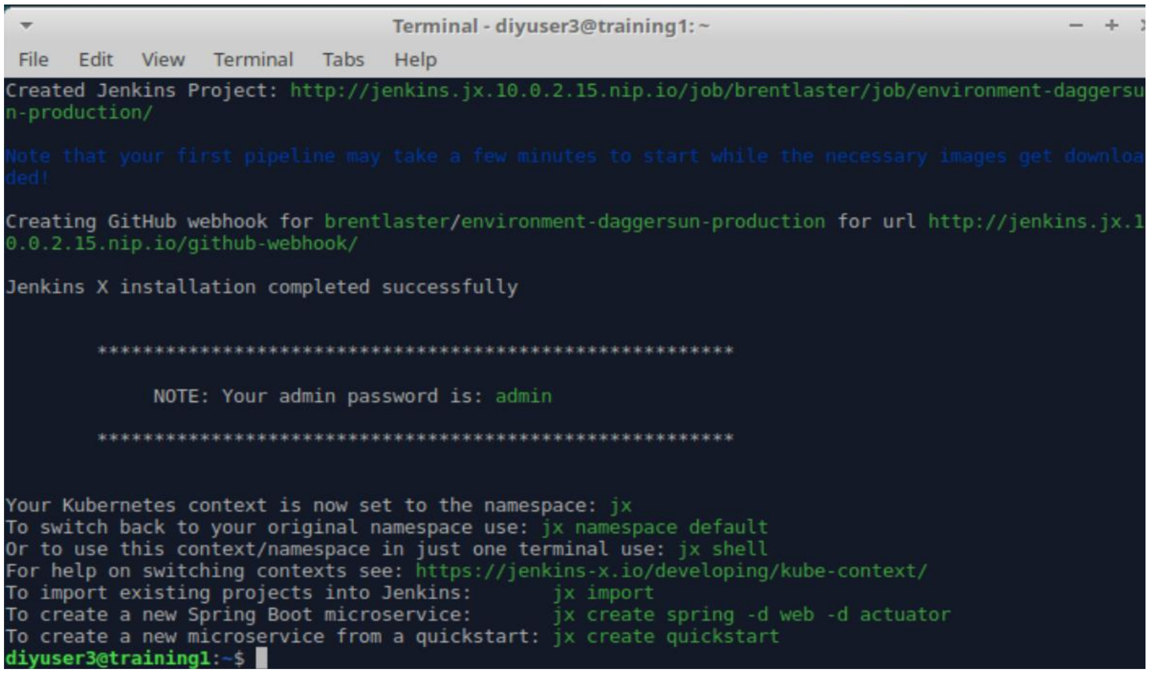
13. This will take a while. You can leave it running while we cover the next section. (You can also ignore the WARNING about vault.)

Lab 3: Creating a Quickstart Project

Purpose: In this lab, we'll finish setting up our jx install and then create a Quickstart project with it.

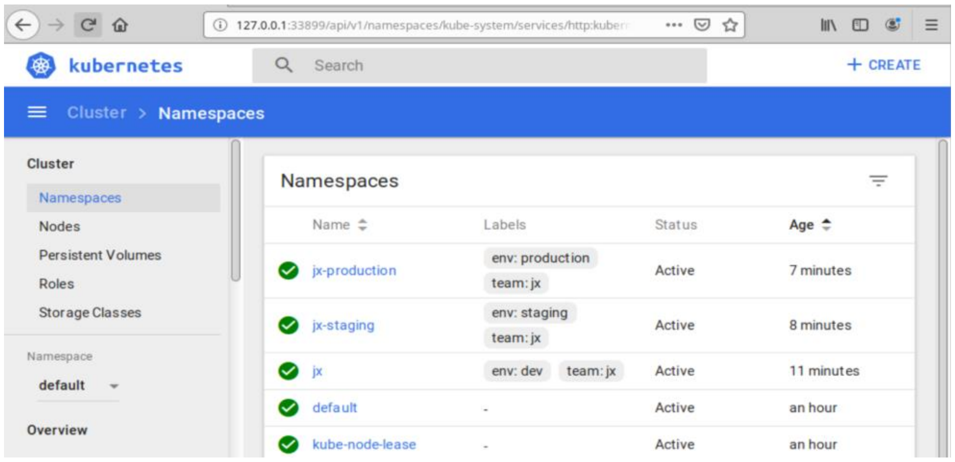
1. At this point, your install will probably be at the point of asking you to "Select the organization where you want to create the environment repository:". It should also be displaying your GitHub userid as the suggested response. (If you have multiple organizations, choose the one you want to use.) Go ahead and hit Enter to accept that.

2. After a few moments, you should see a completion message saying "Jenkins X installation completed successfully. It will also show your admin password and other information.


Terminal - diyuser3@training1: ~
File Edit View Terminal Tabs Help

```
Created Jenkins Project: http://jenkins.jx.10.0.2.15.nip.io/job/brentlaster/job/environment-daggersu
n-production/

Note that your first pipeline may take a few minutes to start while the necessary images get downloa
ded!

Creating GitHub webhook for brentlaster/environment-daggersun-production for url http://jenkins.jx.1
0.0.2.15.nip.io/github-webhook/

Jenkins X installation completed successfully


        ********************************************************

             NOTE: Your admin password is: admin

        ********************************************************


Your Kubernetes context is now set to the namespace: jx
To switch back to your original namespace use: jx namespace default
Or to use this context/namespace in just one terminal use: jx shell
For help on switching contexts see: https://jenkins-x.io/developing/kube-context/
To import existing projects into Jenkins:       jx import
To create a new Spring Boot microservice:       jx create spring -d web -d actuator
To create a new microservice from a quickstart: jx create quickstart
diyuser3@training1:~$
```

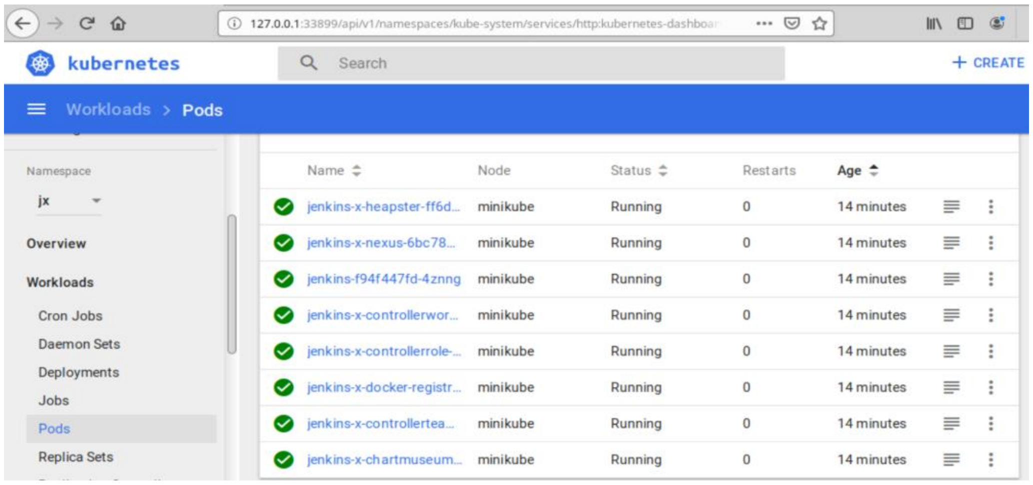
3. Now, let's take a look around and see what Jenkins created for us. Switch back to the Kubernetes dashboard in your browser. Click on the "Namespaces" link under the "Cluster" link in the left menu. Notice that you now have "jx", "jx-staging" and "jx-production" namespaces.



Namespaces

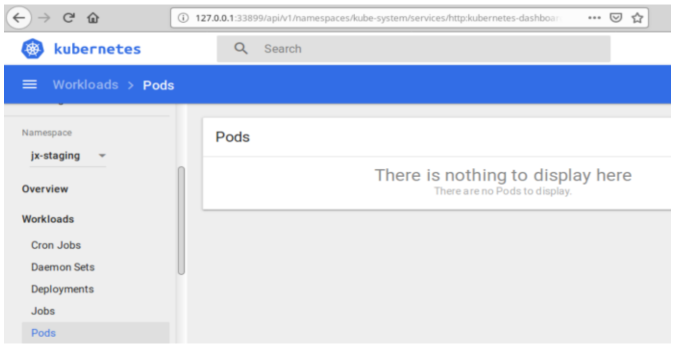
Name	Labels	Status	Age
✓ jx-production	env: production team: jx	Active	7 minutes
✓ jx-staging	env: staging team: jx	Active	8 minutes
✓ jx	env: dev team: jx	Active	11 minutes
✓ default	-	Active	an hour
✓ kube-node-lease	-	Active	an hour

4. Let's look at all the different pods that were created for the different applications in the jx namespace. In the left menu, just below the horizontal bar, under "Namespace", click the down arrow next to "default" and change it to "jx". Then under "Workloads", click the "Pods" link to see the various applications running in pods in this workspace. (You can expand your browser if you can't see the whole name or hover over it.



Name	Node	Status	Restarts	Age
jenkins-x-heapster-ff6d...	minikube	Running	0	14 minutes
jenkins-x-nexus-6bc78...	minikube	Running	0	14 minutes
jenkins-f94f447fd-4znng	minikube	Running	0	14 minutes
jenkins-x-controllerwor...	minikube	Running	0	14 minutes
jenkins-x-controllerrole-...	minikube	Running	0	14 minutes
jenkins-x-docker-registr...	minikube	Running	0	14 minutes
jenkins-x-controllertea...	minikube	Running	0	14 minutes
jenkins-x-chartmuseum...	minikube	Running	0	14 minutes

5. Set the "Namespace" to the "jx-staging" one and notice that there are no pods here yet. (If you don't see it, you may need to force a refresh by hitting F5.)



Pods

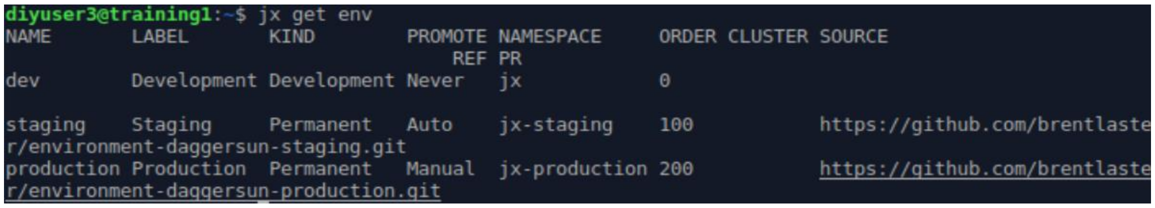
There is nothing to display here

There are no Pods to display.

6. One of the things this process did was create a staging GitHub repository and a production GitHub repository for our use. We'll look at the production one first. In the terminal window, run the command below.

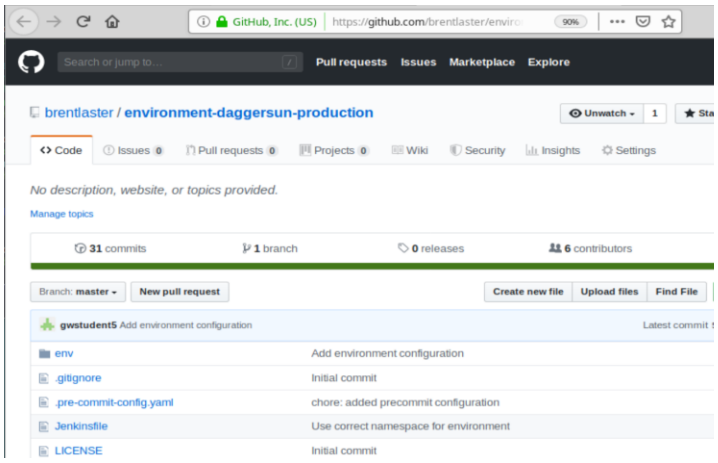
jx get env

7. Hover over the link for the "production" one, right click and select "Open Link".



```
diyuser3@training1:~$ jx get env
NAME       LABEL      KIND        PROMOTE NAMESPACE     ORDER CLUSTER SOURCE
                                      REF PR
dev        Development Development Never   jx            0

staging    Staging    Permanent   Auto    jx-staging    100           https://github.com/brentlaste
r/environment-daggersun-staging.git
production Production Permanent   Manual  jx-production 200           https://github.com/brentlaste
r/environment-daggersun-production.git
```

8. You should be taken to the GitHub location for your production repository.


GitHub, Inc. (US) https://github.com/brentlaster/enviro 90%

Search or jump to... Pull requests Issues Marketplace Explore

brentlaster / environment-daggersun-production Unwatch 1 Sta

<> Code Issues 0 Pull requests 0 Projects 0 Wiki Security Insights Settings

No description, website, or topics provided.

Manage topics

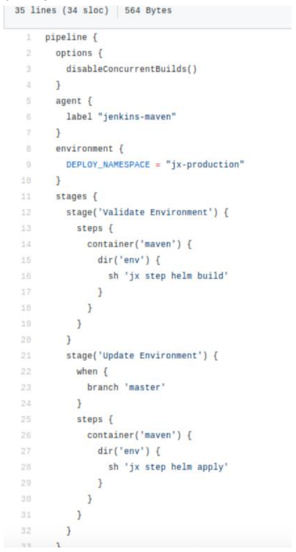
31 commits 1 branch 0 releases 6 contributors

Branch: master New pull request Create new file Upload files Find File

gwstudent5 Add environment configuration Latest commit

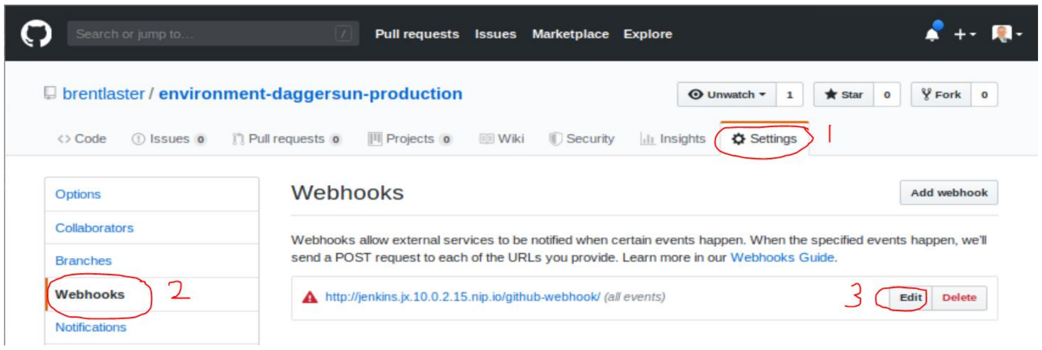
env	Add environment configuration
.gitignore	Initial commit
.pre-commit-config.yaml	chore: added precommit configuration
Jenkinsfile	Use correct namespace for environment
LICENSE	Initial commit

9. While we're here, let's look at the basic Jenkinsfile that was created for Jenkins to run our project. Click on the "Jenkinsfile" link. Take a look at what the Jenkinsfile is doing.


35 lines (34 sloc) | 564 Bytes

```
pipeline {
  options {
    disableConcurrentBuilds()
  }
  agent {
    label "jenkins-maven"
  }
  environment {
    DEPLOY_NAMESPACE = "jx-production"
  }
  stages {
    stage('Validate Environment') {
      steps {
        container('maven') {
          dir('env') {
            sh 'jx step helm build'
          }
        }
      }
    }
    stage('Update Environment') {
      when {
        branch 'master'
      }
      steps {
        container('maven') {
          dir('env') {
            sh 'jx step helm apply'
          }
        }
      }
    }
  }
```

10. While we are here, we need to modify the webhook that was setup for us so it will work with our local instance. Go back one page in the browser. Near the top of the screen, in the line of tabs that starts with "<> Code", click on the **Settings** tab (at the end). Then in the left menu, click on **Webhooks**. Then click on the **Edit** button on the right.



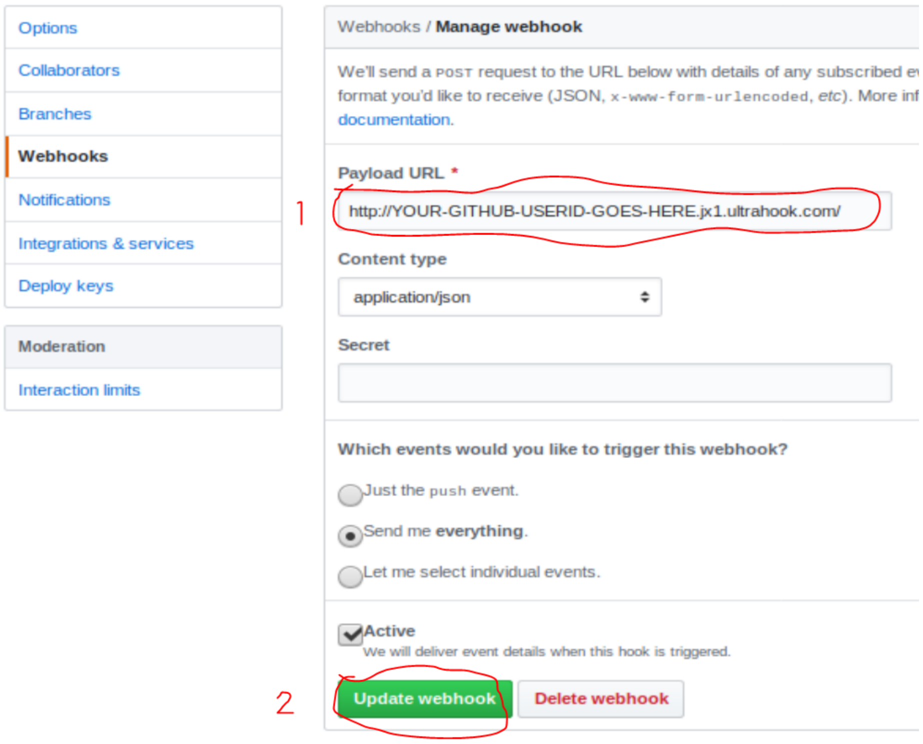
Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

http://jenkins.jx.10.0.2.15.nip.io/github-webhook/ (all events)

11. In the **Webhooks/Manage** webhook screen, change the **Payload URL** field to be the link below (substituting your github userid for "<your github userid>"). That's the only thing you need to change. After you make the change, click on the **Update webhook** button. (You may then need to enter your password.)

http://**<your github userid>**.jx1.ultrahook.com/



We'll send a POST request to the URL below with details of any subscribed e
format you'd like to receive (JSON, x-www-form-urlencoded, etc). More inf
documentation.

Payload URL *
http://YOUR-GITHUB-USERID-GOES-HERE.jx1.ultrahook.com/

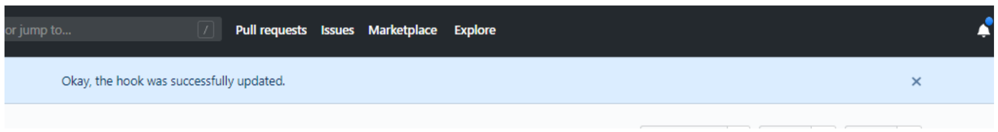
Content type
application/json

Secret

Which events would you like to trigger this webhook?
Just the push event.
[✓ Send me everything.]
Let me select individual events.

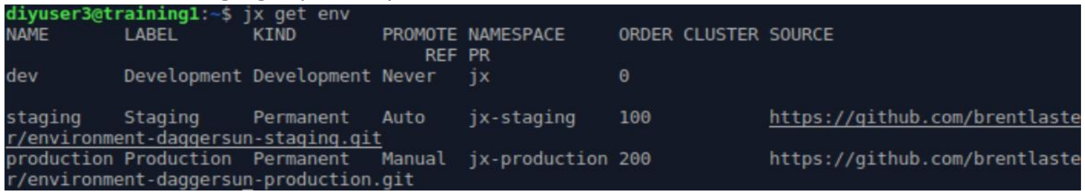
[✓ Active]
We will deliver event details when this hook is triggered.

12. If all went well, you should see a message at the top that tells you the hook was successfully updated.



Okay, the hook was successfully updated.

13. **<u>Make the same webhook change for the staging environment</u>.** You can get that link back in the terminal window with the list of environments. After you open up that link, repeat steps 10-12 above for the -staging repository.

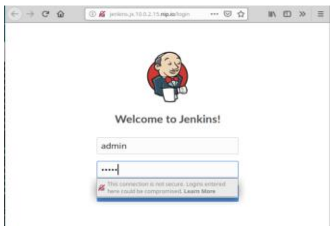


```
diyuser3@training1:~$ jx get env
NAME       LABEL      KIND        PROMOTE NAMESPACE     ORDER CLUSTER SOURCE
                                    REF  PR
dev        Development Development Never   jx            0

staging    Staging    Permanent   Auto    jx-staging    100           https://github.com/brentlaste
r/environment-daggersun-staging.git
production Production Permanent   Manual  jx-production 200           https://github.com/brentlaste
r/environment-daggersun-production.git
```

14. Next let's look at the Jenkins project that was setup for us. Connect to the static Jenkins instance by typing

jx console

15. This should take you to the sign-in screen for the Jenkins instance that is running in the jx namespace in the cluster. Login with userid of "admin" and password of "admin".

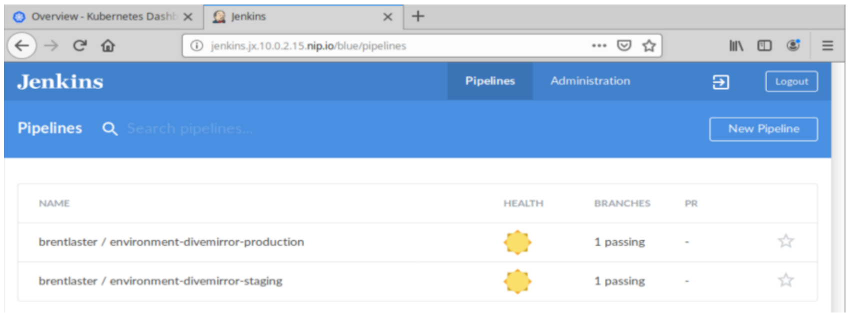

jenkins.jx.10.0.2.15.nip.io/login

Welcome to Jenkins!

admin

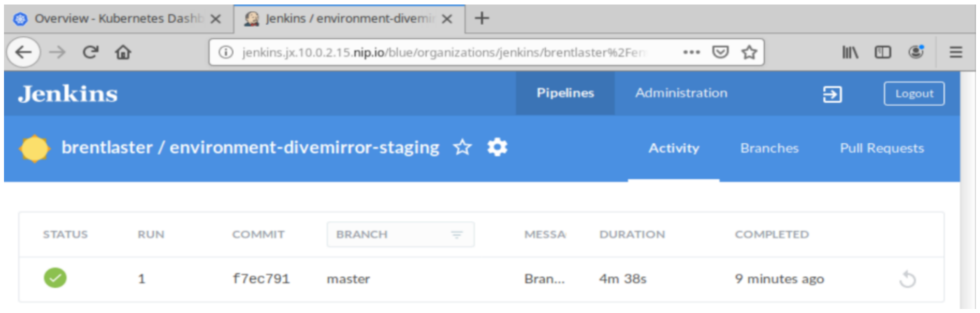
This connection is not secure. Logins entered here could be compromised. Learn More

16. This will put you on the Blue Ocean interface screen for the pipelines associated with the production and staging environments.



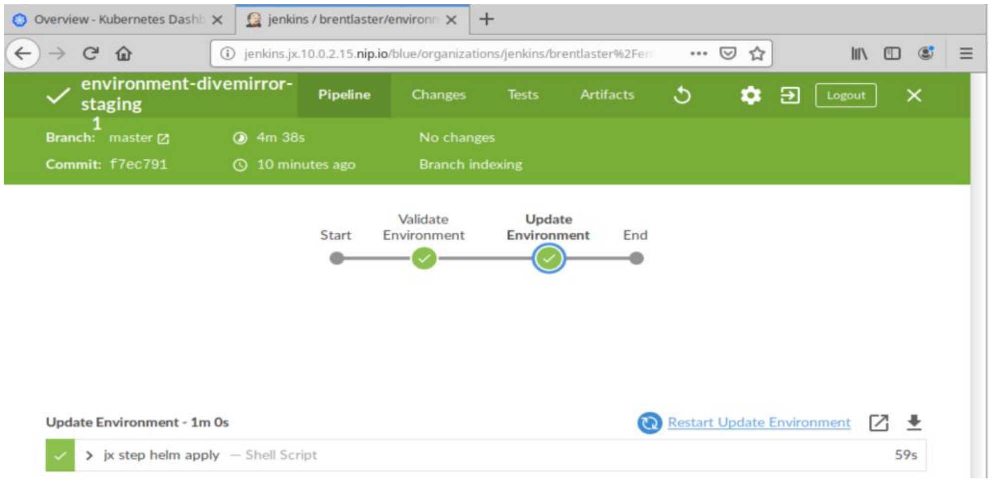
NAME	HEALTH	BRANCHES	PR
brentlaster / environment-divemirror-production	☀	1 passing	-
brentlaster / environment-divemirror-staging	☀	1 passing	-

17. Select one of the jobs and click on the name. These are multi-branch pipelines which means it creates jobs in Jenkins based on the branches in the GitHub repository that have Jenkinsfiles. For right now, this is only for master.


Overview - Kubernetes Dash
Jenkins / environment-divemi
jenkins.jx.10.0.2.15.nip.io/blue/organizations/jenkins/brentlaster%2Fen
Jenkins
Pipelines
Administration
Logout
brentlaster / environment-divemirror-staging
Activity
Branches
Pull Requests

STATUS	RUN	COMMIT	BRANCH	MESSA	DURATION	COMPLETED
✓	1	f7ec791	master	Bran...	4m 38s	9 minutes ago

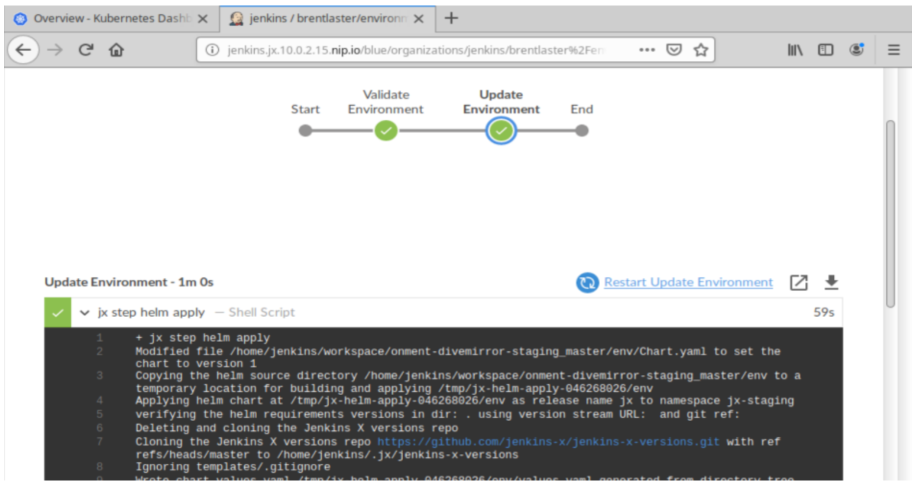
18. Click on the "master" branch and you can see the Blue Ocean output. Notice that the stages here correspond to the stages that were defined in the Jenkinsfile.


Overview - Kubernetes Dash
jenkins / brentlaster/environ
jenkins.jx.10.0.2.15.nip.io/blue/organizations/jenkins/brentlaster%2Fen
environment-divemirror-staging
1
Pipeline
Changes
Tests
Artifacts
Logout
Branch: master
4m 38s
No changes
Commit: f7ec791
10 minutes ago
Branch indexing
Start
Validate Environment
Update Environment
End
Restart Update Environment

Update Environment - 1m 0s

```
> jx step helm apply — Shell Script    59s
```

19. If you want to see more details on the build, you can click on the steps at the bottom to expand them. Note that the steps will vary depending on which circle you have selected.


Overview - Kubernetes Dashb
jenkins / brentlaster/environ
jenkins.jx.10.0.2.15.nip.io/blue/organizations/jenkins/brentlaster%2Fen
Start
Validate Environment
Update Environment
End
Update Environment - 1m 0s
Restart Update Environment
jx step helm apply — Shell Script
59s

```
+ jx step helm apply
Modified file /home/jenkins/workspace/onment-divemirror-staging_master/env/Chart.yaml to set the
chart to version 1
Copying the helm source directory /home/jenkins/workspace/onment-divemirror-staging_master/env to a
temporary location for building and applying /tmp/jx-helm-apply-046268026/env
Applying helm chart at /tmp/jx-helm-apply-046268026/env as release name jx to namespace jx-staging
verifying the helm requirements versions in dir: . using version stream URL:  and git ref:
Deleting and cloning the Jenkins X versions repo
Cloning the Jenkins X versions repo https://github.com/jenkins-x/jenkins-x-versions.git with ref
refs/heads/master to /home/jenkins/.jx/jenkins-x-versions
Ignoring templates/.gitignore
```

20. We need to fix up one more thing to avoid an error later in our build.   We need to change Docker to use the insecure-registry that is running in our jx namespace. In the original terminal session, run the command below.

    ./fixregistry.sh

    It will take a little bit to complete, but you can leave it running while we continue. It will also interrupt your Jenkins session and restart it but that's ok. (You may see a small box in your browser pop up that says "Connection lost: waiting" - that's ok. It will reload eventually.)

Lab 4: Creating a Quickstart project

Purpose: In this lab, we'll see how easy it is to create a simple Node project via a quickstart and see it go through our pipeline.

1. We're going to use an application called Ultrahook to allow the jx webhook functionality to be used with our minikube instance. Start that by opening a separate terminal session and running the following command, passing your GitHub userid as the one argument.

   ./runhook.sh <your GitHub userid>

   (For example, mine would be ./runhook.sh brentlaster)

2. In your original terminal session, issue the command to start creating the quickstart.

   jx create quickstart

3. At the prompt about selecting the quickstart you wish to create, use the arrow keys to select the "**node-http**" project type and hit Enter.

4. At the prompt about the Git user name, just hit Enter to use the GitHub username we've been using.

5. Follow the instructions to create a token and paste it at the "API Token:" prompt. (**Save the token for further use in step 11.**)

6. If prompted about the GitHub organization, just select the one that matches the one you selected before – probably your GitHub username.

7. At the prompt for the new repository name, enter a name such as "node-proj1". (If you want to use a different name, you can – just use that name wherever we use "node-proj1" in the rest of the labs.)

8. At the prompt about "initialize git now", just hit Enter to take the default yes option.

9. At the commit message prompt, you can type a new commit message or just hit Enter to accept the default one.

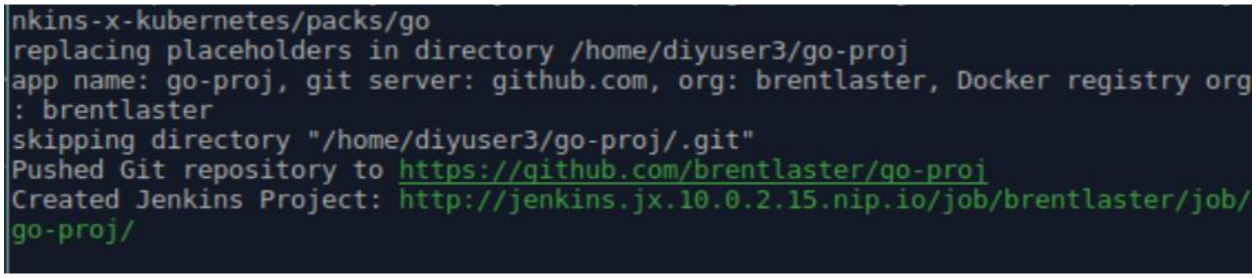
10. After a few moments, you should see the messages indicating that the project has been created. Notice that it has pushed it up to a GitHub repository and created a new Jenkins project for it.

11. Now let's update the webhook on this project so it can send notifications to our local system. For this one, Jenkins X provides a command line way to do this. Run the command below (substituting in your github userid where noted). And note that there are two hyphens before "endpoint".

    jx update webhooks --endpoint http://<your github userid>.jx1.ultrahook.com/

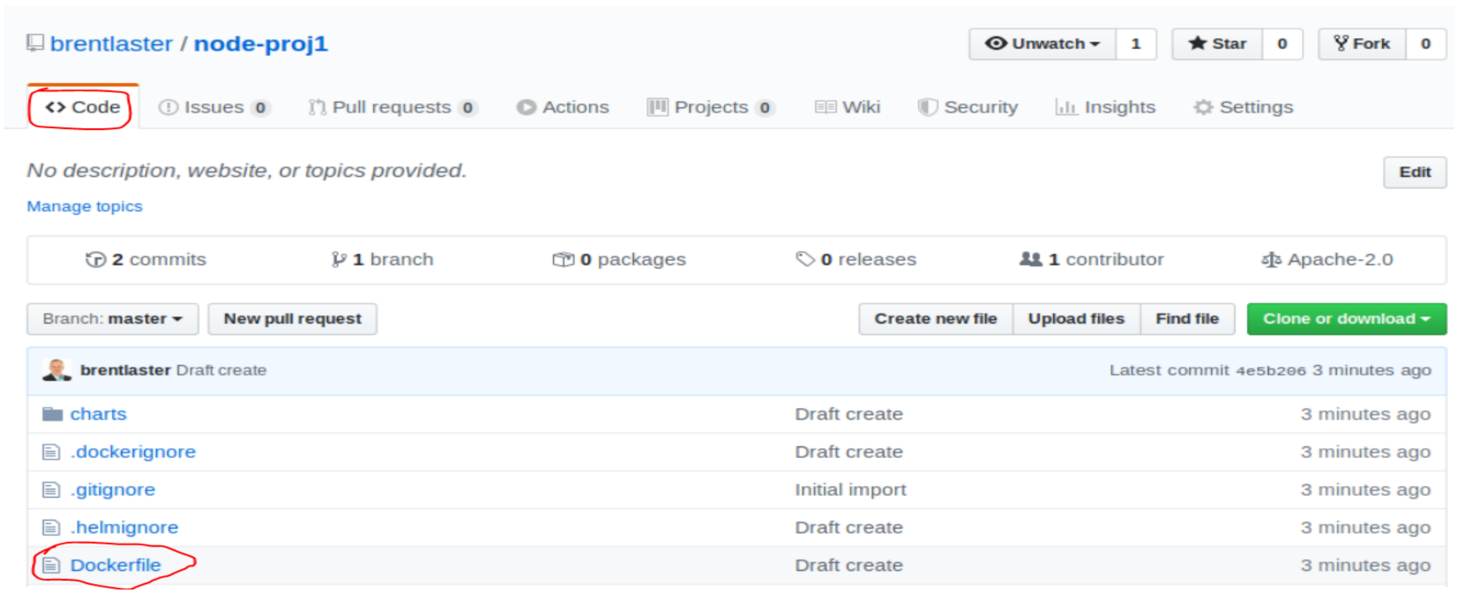
    If prompted, enter your GitHub userid. For password, you should be able to paste the token you used in step 5 above. Paste it – it will be blank – and hit enter. (Note it won't show up as pasted or "****" in the prompt.)

12. Find the link in the output for the GitHub project and then open it up and take a look.



```
nkins-x-kubernetes/packs/go
replacing placeholders in directory /home/diyuser3/go-proj
app name: go-proj, git server: github.com, org: brentlaster, Docker registry org
: brentlaster
skipping directory "/home/diyuser3/go-proj/.git"
Pushed Git repository to https://github.com/brentlaster/go-proj
Created Jenkins Project: http://jenkins.jx.10.0.2.15.nip.io/job/brentlaster/job/
go-proj/
```

13. Let's take a look at how the project is put together. Go back to the Code section by clicking on the **Code** tab on the far left. Then open up the Dockerfile to see how the container is constructed.

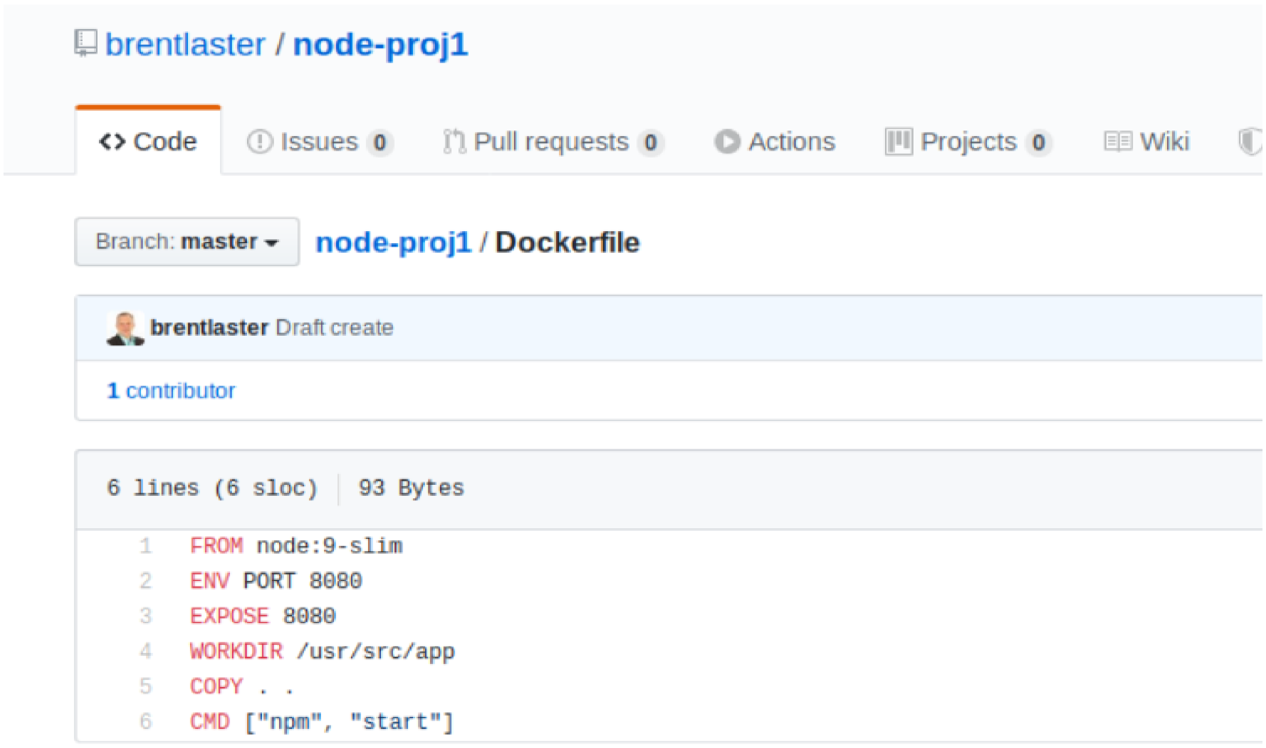


No description, website, or topics provided.

2 commits | 1 branch | 0 packages | 0 releases | 1 contributor | Apache-2.0

brentlaster Draft create — Latest commit 4e5b206 3 minutes ago

charts	Draft create	3 minutes ago
.dockerignore	Draft create	3 minutes ago
.gitignore	Initial import	3 minutes ago
.helmignore	Draft create	3 minutes ago
Dockerfile	Draft create	3 minutes ago



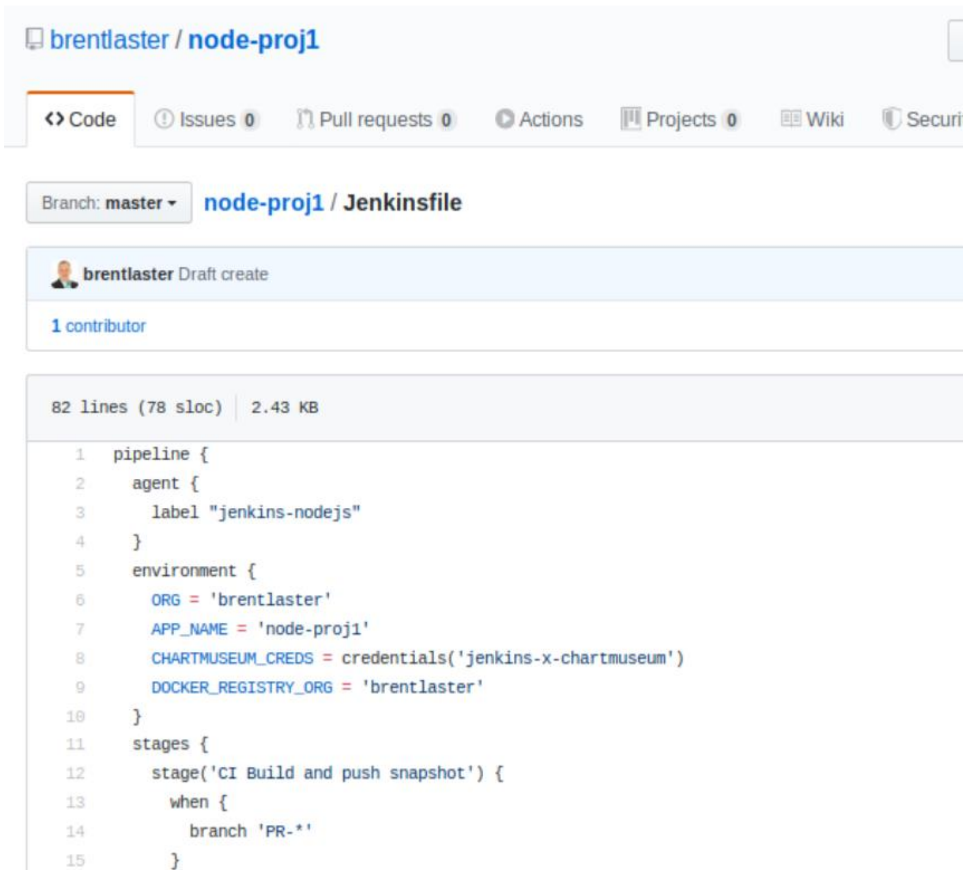
brentlaster Draft create

1 contributor

6 lines (6 sloc) | 93 Bytes

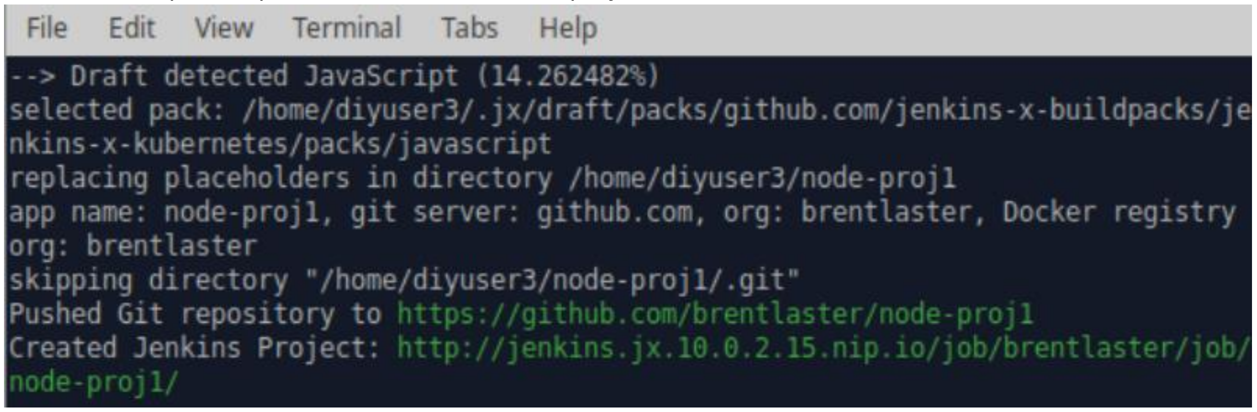
```
FROM node:9-slim
ENV PORT 8080
EXPOSE 8080
WORKDIR /usr/src/app
COPY . .
CMD ["npm", "start"]
```

14. Next, open up the “Jenkinsfile” (below the Dockerfile in the Code list) and look at the kind of processing that it’s doing.



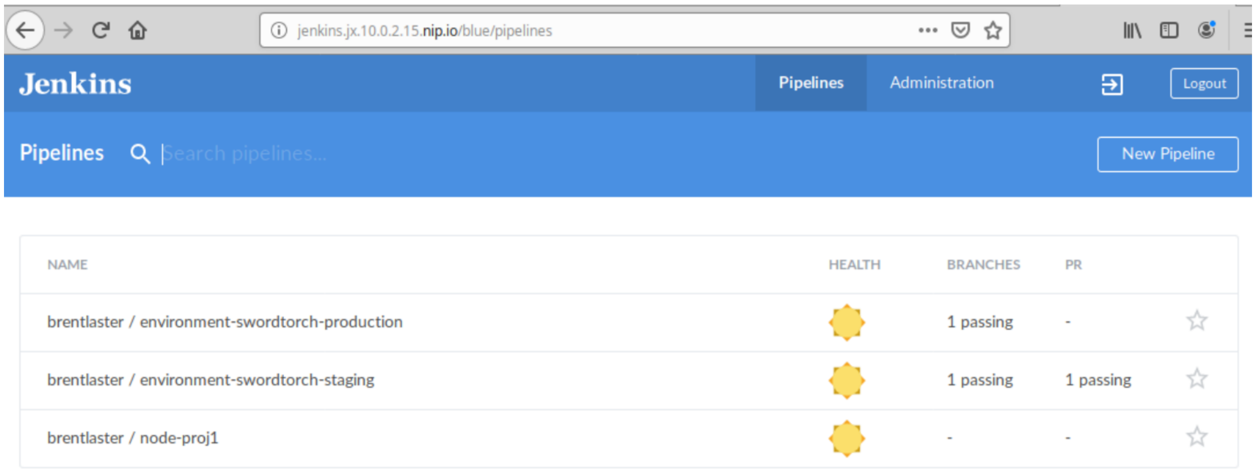
```
pipeline {
  agent {
    label "jenkins-nodejs"
  }
  environment {
    ORG = 'brentlaster'
    APP_NAME = 'node-proj1'
    CHARTMUSEUM_CREDS = credentials('jenkins-x-chartmuseum')
    DOCKER_REGISTRY_ORG = 'brentlaster'
  }
  stages {
    stage('CI Build and push snapshot') {
      when {
        branch 'PR-*'
      }
```

15. Now switch back to the terminal, find the link in the output for the Jenkins project that was created and open it up in Jenkins to look at the project.



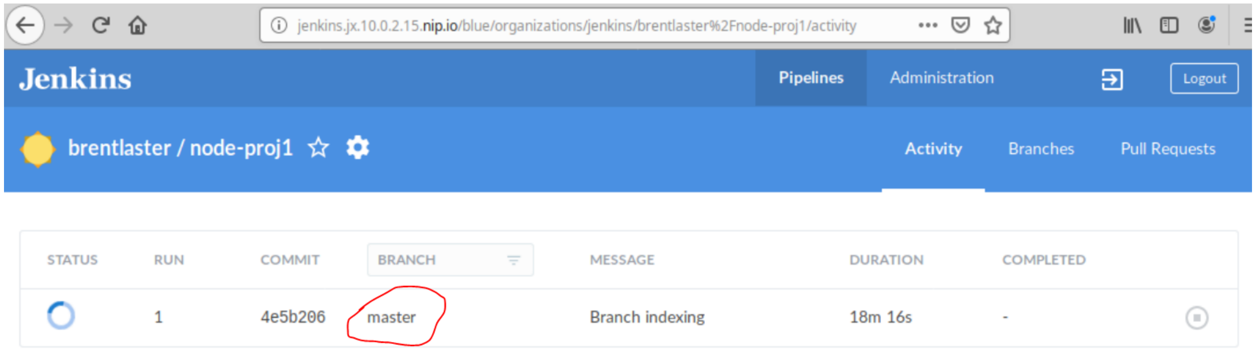
```
--> Draft detected JavaScript (14.262482%)
selected pack: /home/diyuser3/.jx/draft/packs/github.com/jenkins-x-buildpacks/je
nkins-x-kubernetes/packs/javascript
replacing placeholders in directory /home/diyuser3/node-proj1
app name: node-proj1, git server: github.com, org: brentlaster, Docker registry
org: brentlaster
skipping directory "/home/diyuser3/node-proj1/.git"
Pushed Git repository to https://github.com/brentlaster/node-proj1
Created Jenkins Project: http://jenkins.jx.10.0.2.15.nip.io/job/brentlaster/job/
node-proj1/
```

16. Login again with the same userid and password. If you are not seeing all the pipelines on the screen, click on the word "Pipelines" in the blue bar at the top. (If you are not in the Blue Ocean interface, you may need to click the “Open Blue Ocean” link in the left menu, and then click on “Pipelines”.) You should then see something like this.



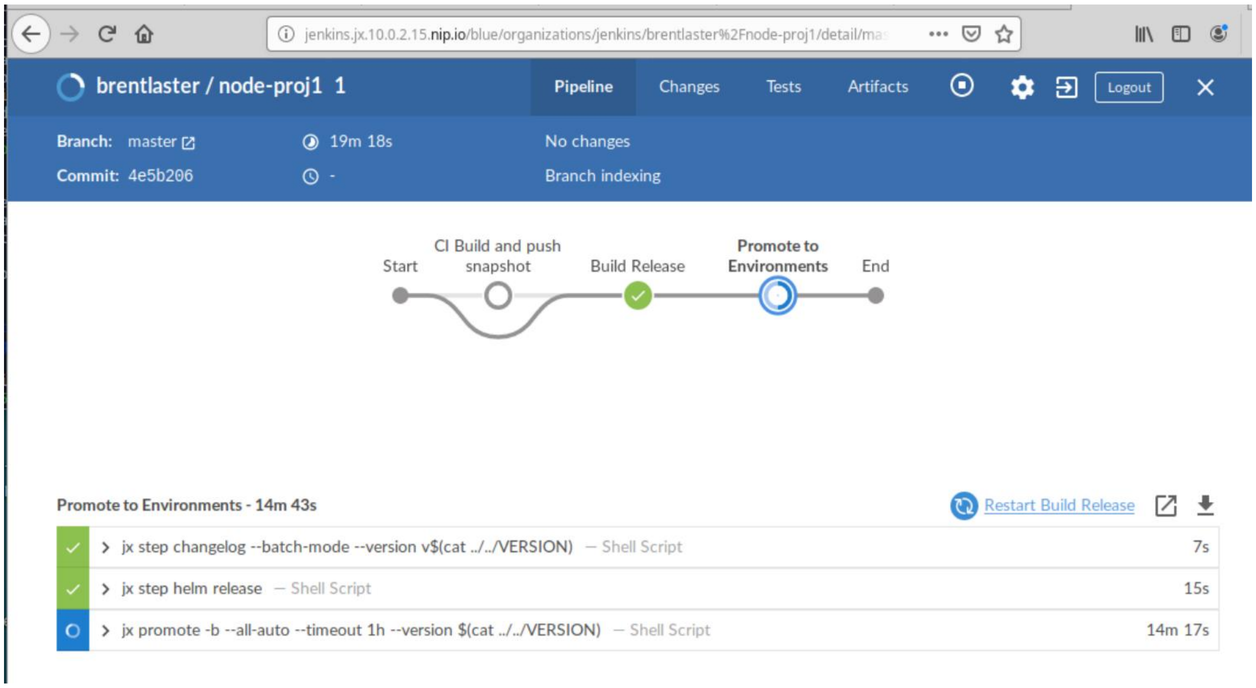
NAME	HEALTH	BRANCHES	PR
brentlaster / environment-swordtorch-production		1 passing	-
brentlaster / environment-swordtorch-staging		1 passing	1 passing
brentlaster / node-proj1		-	-

17. From here, click on the row for the name of the quickstart project you created.



STATUS	RUN	COMMIT	BRANCH	MESSAGE	DURATION	COMPLETED
	1	4e5b206	master	Branch indexing	18m 16s	-

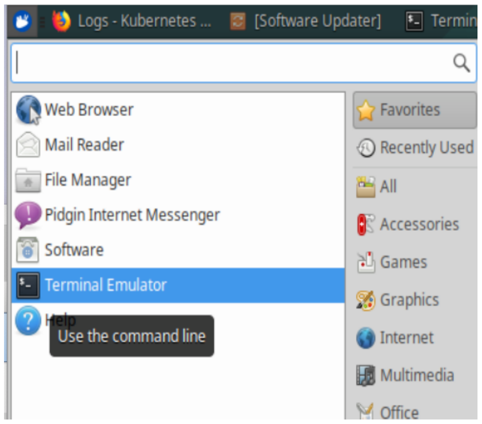
18. And then click on the entry for the "master" branch to see the running pipeline. You can expand any of the steps you want to see the output and what was actually done.



Promote to Environments - 14m 43s

```
✓ > jx step changelog --batch-mode --version v$(cat ../../VERSION) — Shell Script    7s
✓ > jx step helm release — Shell Script                                              15s
○ > jx promote -b --all-auto --timeout 1h --version $(cat ../../VERSION) — Shell Script    14m 17s
```

19. At the end of the output from the create quickstart command, you should see a list of jx commands you can run to see things about the pipeline. We'll use the watch option to watch the pipeline activity. Open up a third terminal window to use for watching the activity. To do this, click on the mouse head icon in the upper left of the VM window, then select "Terminal Emulator" from the list of apps.



20. In this third terminal window, enter the following command:

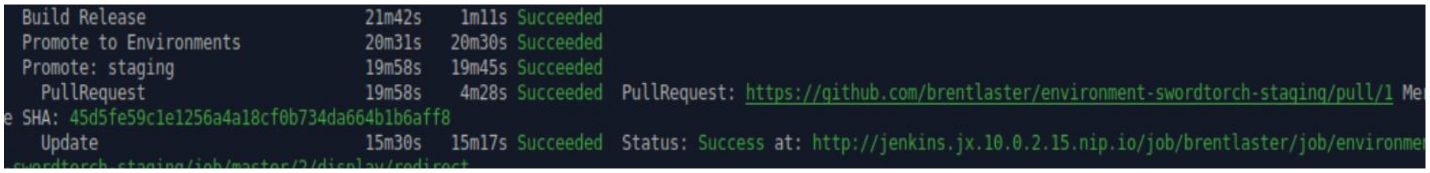
jx get activity -f <project name> -w

You can leave this running while class continues.

Lab 5: Creating a Preview environment

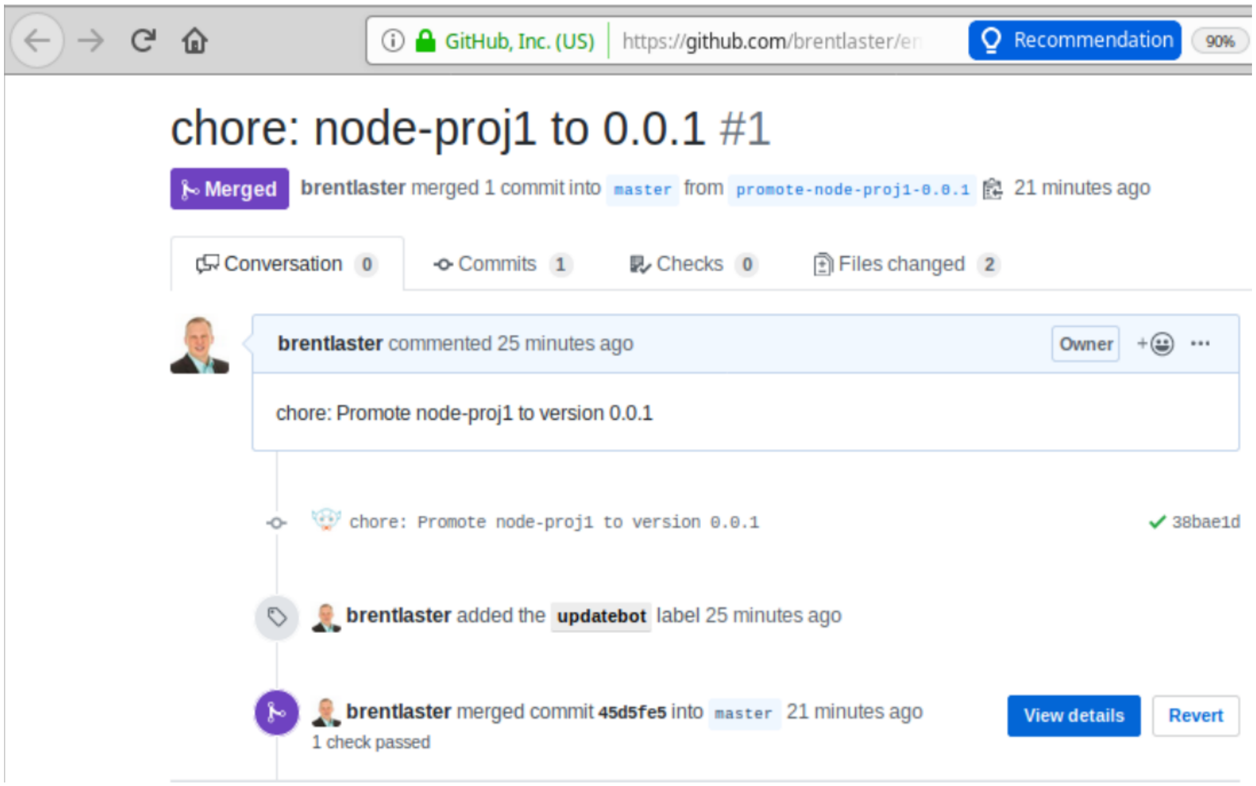
Purpose: In this lab, we'll see how to make a change to our app, push the changes, create a Pull Request, merge it, and have Jenkins X spin up a Preview Environment for us.

1. At this point, your new go quickstart should have gone through the CI/CD pipeline that Jenkins X created and hopefully completed that process successfully. One of the items that was created for this was a Pull Request in your GitHub project. Let's take a look at that. In the terminal window that has the output from the watch (the last command we did in the previous lab), find the line (a few from the bottom) that starts with "PullRequest". Hover over the link in that line that starts with "https", right-click and "Open Link".



```
  Build Release                    21m42s    1m11s Succeeded
  Promote to Environments          20m31s   20m30s Succeeded
  Promote: staging                 19m58s   19m45s Succeeded
    PullRequest                    19m58s    4m28s Succeeded  PullRequest: https://github.com/brentlaster/environment-swordtorch-staging/pull/1 Me
e SHA: 45d5fe59c1e1256a4a18cf0b734da664b1b6aff8
    Update                         15m30s   15m17s Succeeded  Status: Success at: http://jenkins.jx.10.0.2.15.nip.io/job/brentlaster/job/environme
```

2. Find the "View details" around the middle of the page (below the green checkmark). Click on the button to see what occurred automatically as part of the pipeline processing.



GitHub, Inc. (US) https://github.com/brentlaster/er Recommendation 90%

chore: node-proj1 to 0.0.1 #1

Merged brentlaster merged 1 commit into `master` from `promote-node-proj1-0.0.1` 21 minutes ago

Conversation 0 · Commits 1 · Checks 0 · Files changed 2

brentlaster commented 25 minutes ago — Owner

chore: Promote node-proj1 to version 0.0.1

`chore: Promote node-proj1 to version 0.0.1` ✓ 38bae1d

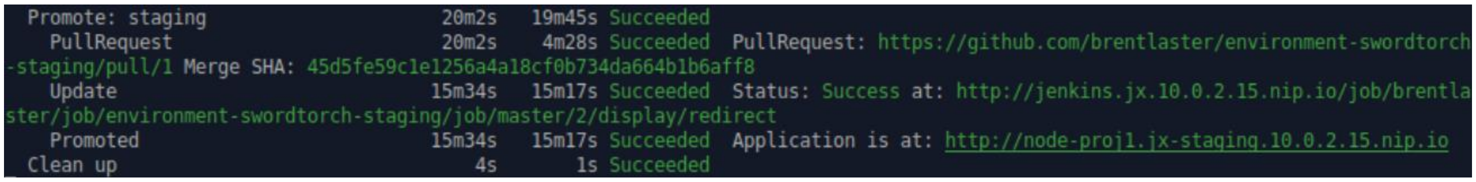
brentlaster added the **updatebot** label 25 minutes ago

brentlaster merged commit **45d5fe5** into `master` 21 minutes ago
1 check passed

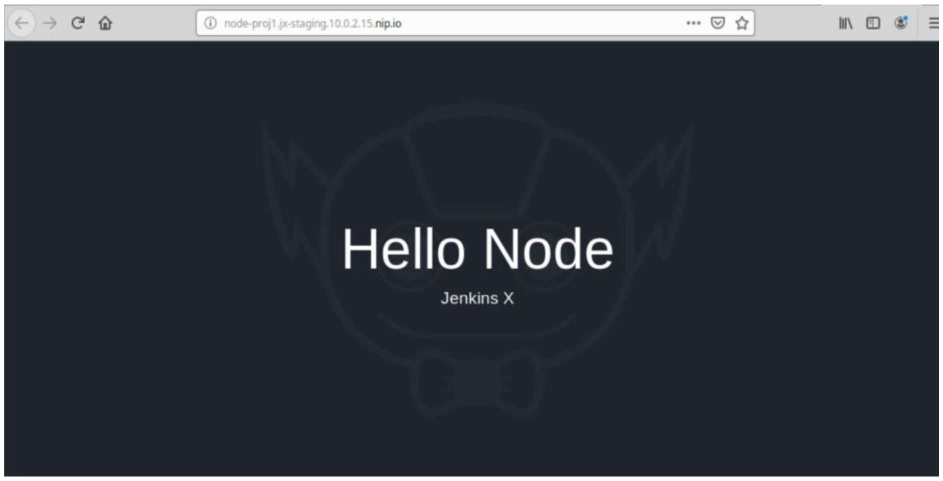
View details · Revert

3. This tells us that there was a merge check run for the pull request automatically that passed. The job may no longer be in Jenkins since the Pull Request has been merged successfully (as indicated by the "Merged" status in the upper left).

4. Now let's look at the actual app that got created as it is running. In the terminal session with the watch running, find the line (usually the last one) that starts with "Promoted" and has the phrase "Application is at:" followed by a link to the URL for the staging version of our running application. Open that link and you should see the running instance of your app in the staging environment.



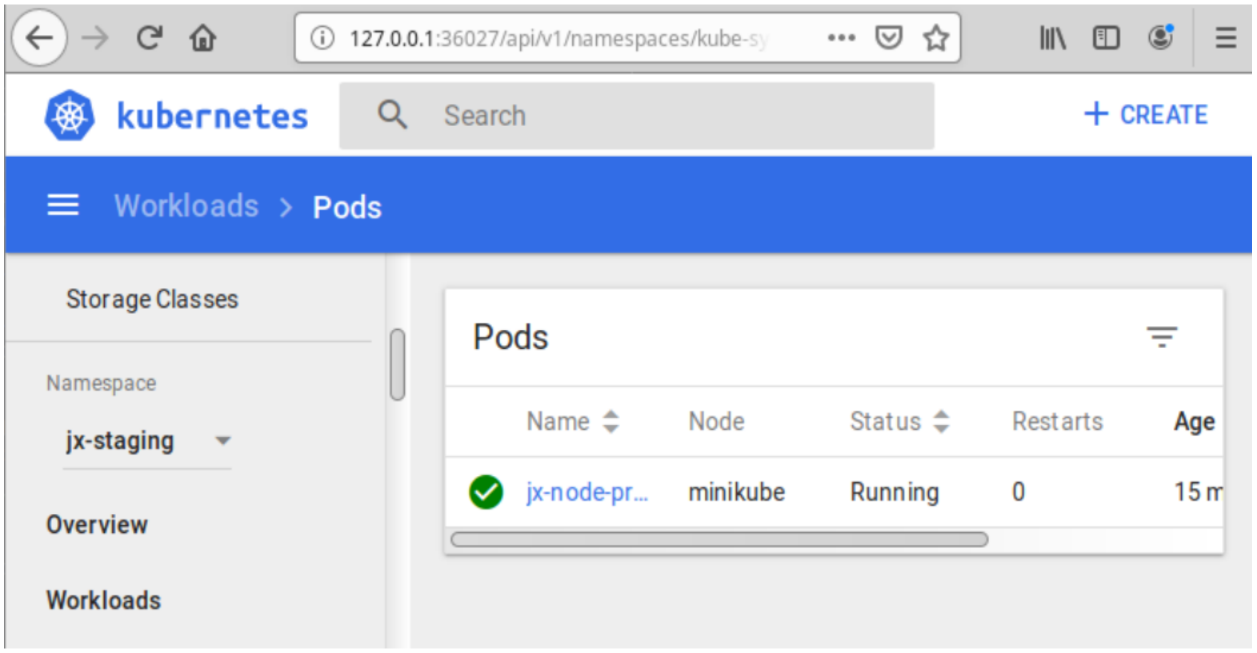
```
  Promote: staging                      20m2s    19m45s Succeeded
    PullRequest                         20m2s     4m28s Succeeded  PullRequest: https://github.com/brentlaster/environment-swordtorch
-staging/pull/1 Merge SHA: 45d5fe59c1e1256a4a18cf0b734da664b1b6aff8
    Update                             15m34s    15m17s Succeeded  Status: Success at: http://jenkins.jx.10.0.2.15.nip.io/job/brentla
ster/job/environment-swordtorch-staging/job/master/2/display/redirect
    Promoted                           15m34s    15m17s Succeeded  Application is at: http://node-proj1.jx-staging.10.0.2.15.nip.io
  Clean up                                 4s        1s Succeeded
```



5. This app is running in a container that is being run in a Kubernetes pod in our staging namespace within the cluster. Let's drill down to find that pod and look at the logs from it. If you no longer have the Kubernetes dashboard up, or if you experience problems, then start it again from a terminal session with:

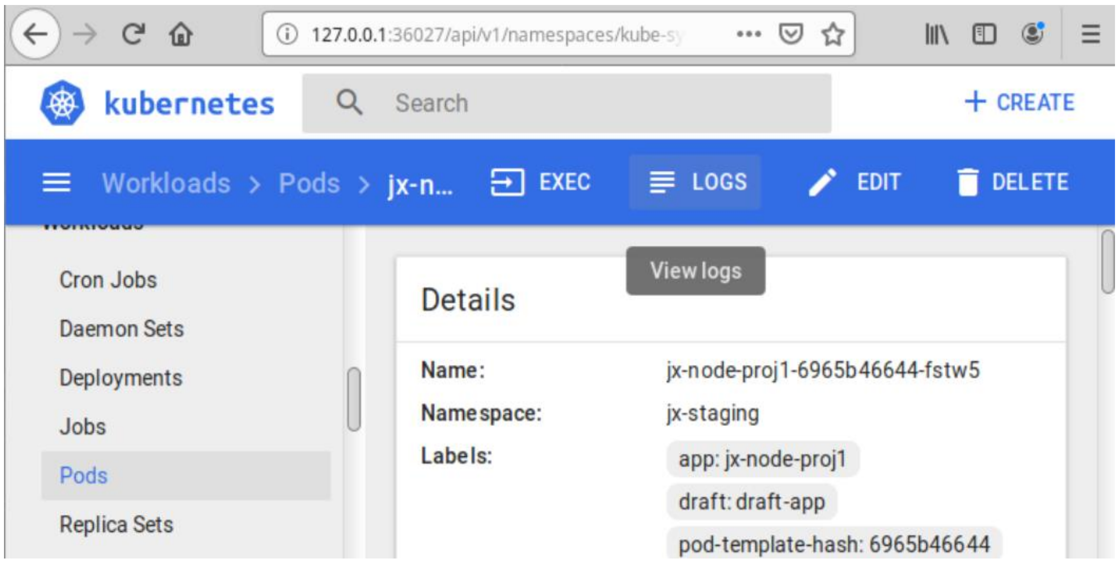
   minikube dashboard &

6. In the browser tab for the dashboard, under the line in the middle of the left column, under "Namespace" select **jx-staging**. Then under Workloads, scroll down and select **Pods**.

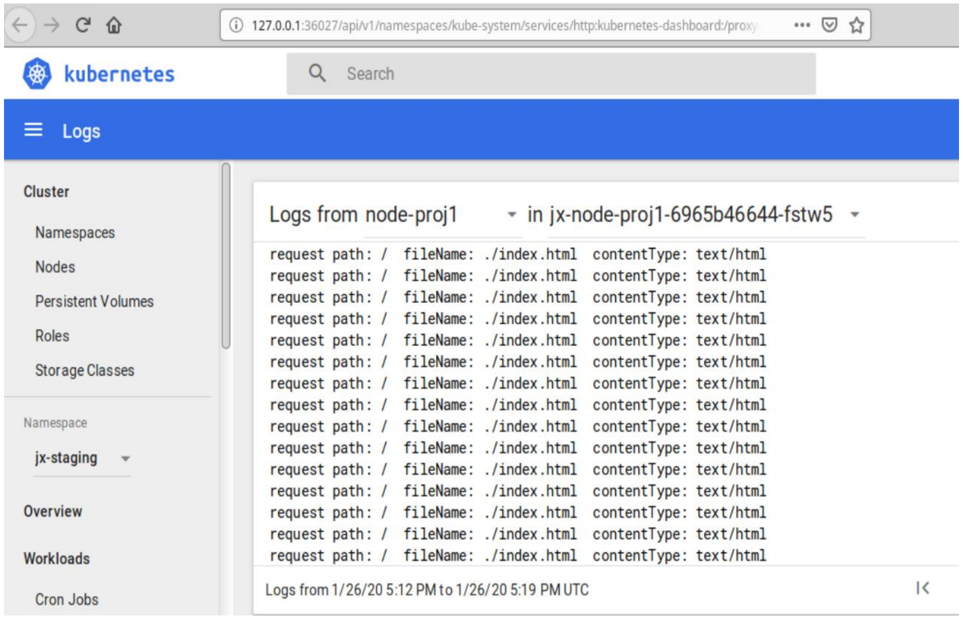


Name	Node	Status	Restarts	Age
jx-node-pr...	minikube	Running	0	15 m

7. In the Pods section on the right, click on the pod name for your project (will be different than what's shown). Then, in the blue bar across the top, click on **LOGS**. After that, you should see similar log output from the app running inside the container as you saw in the web page for the app. (Again, names will be different.)



Name: jx-node-proj1-6965b46644-fstw5
Namespace: jx-staging
Labels: app: jx-node-proj1
draft: draft-app
pod-template-hash: 6965b46644



Logs from node-proj1 in jx-node-proj1-6965b46644-fstw5

```
request path: /  fileName: ./index.html  contentType: text/html
request path: /  fileName: ./index.html  contentType: text/html
request path: /  fileName: ./index.html  contentType: text/html
request path: /  fileName: ./index.html  contentType: text/html
request path: /  fileName: ./index.html  contentType: text/html
request path: /  fileName: ./index.html  contentType: text/html
request path: /  fileName: ./index.html  contentType: text/html
request path: /  fileName: ./index.html  contentType: text/html
request path: /  fileName: ./index.html  contentType: text/html
request path: /  fileName: ./index.html  contentType: text/html
request path: /  fileName: ./index.html  contentType: text/html
request path: /  fileName: ./index.html  contentType: text/html
request path: /  fileName: ./index.html  contentType: text/html
request path: /  fileName: ./index.html  contentType: text/html
request path: /  fileName: ./index.html  contentType: text/html
```

Logs from 1/26/20 5:12 PM to 1/26/20 5:19 PM UTC

8. Now let’s see how Jenkins X can automatically create a Preview Environment for a Pull Request that we create.

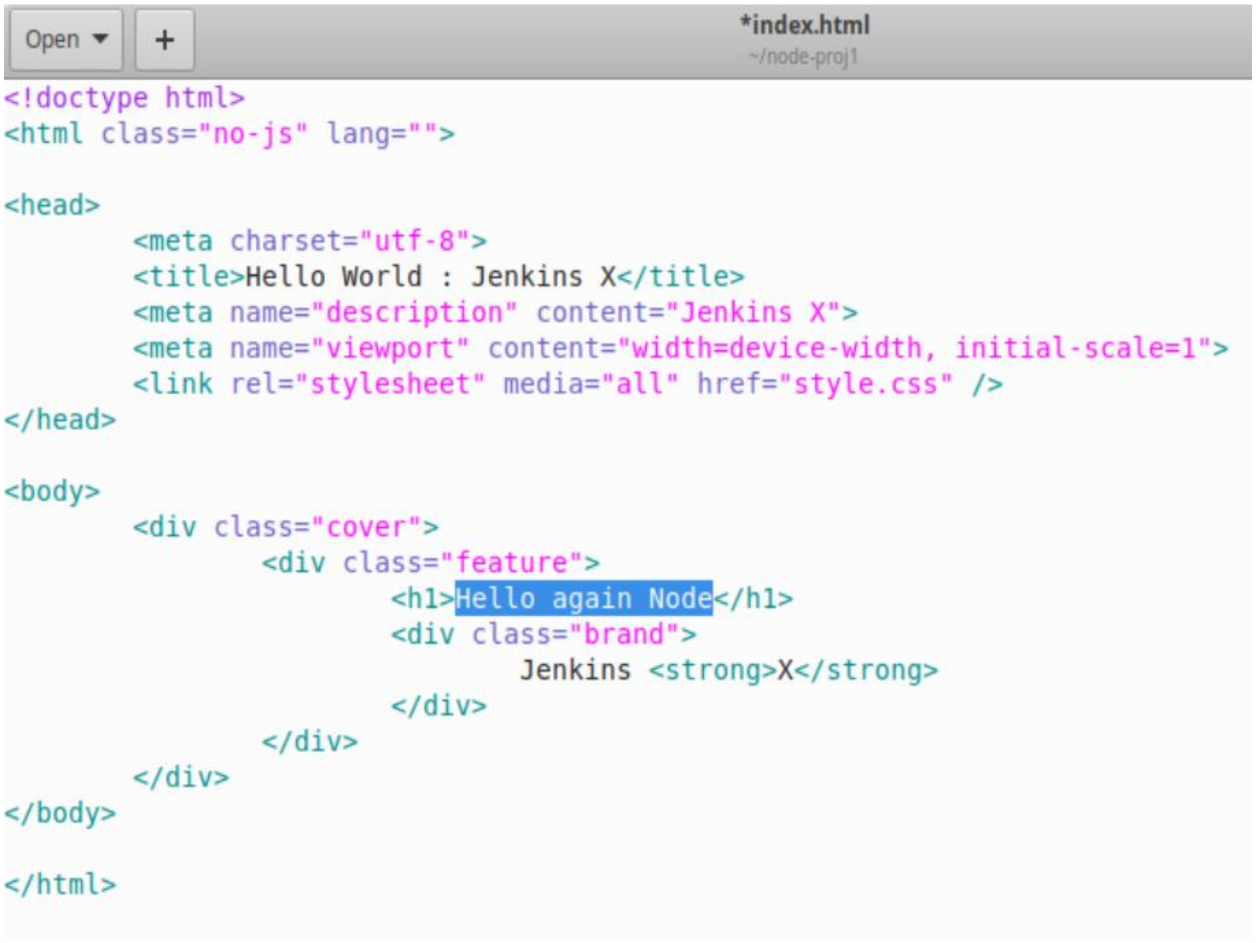
9. In an available terminal window, change into the directory for your project. (Hit Enter if you don't see a prompt.). Then create a new branch and modify the main.go file to add some other words in the message that is displayed in the browser.

   cd ~/<directory for project>

   git checkout -b update

   gedit index.html

   Then, modify the “<h1>Hello Node</h1>” statement in the editor (about 9th line from the bottom) as shown next. Change "Hello Node" to be "Hello again Node” - or any text you want). Ignore any errors in the terminal from gedit.

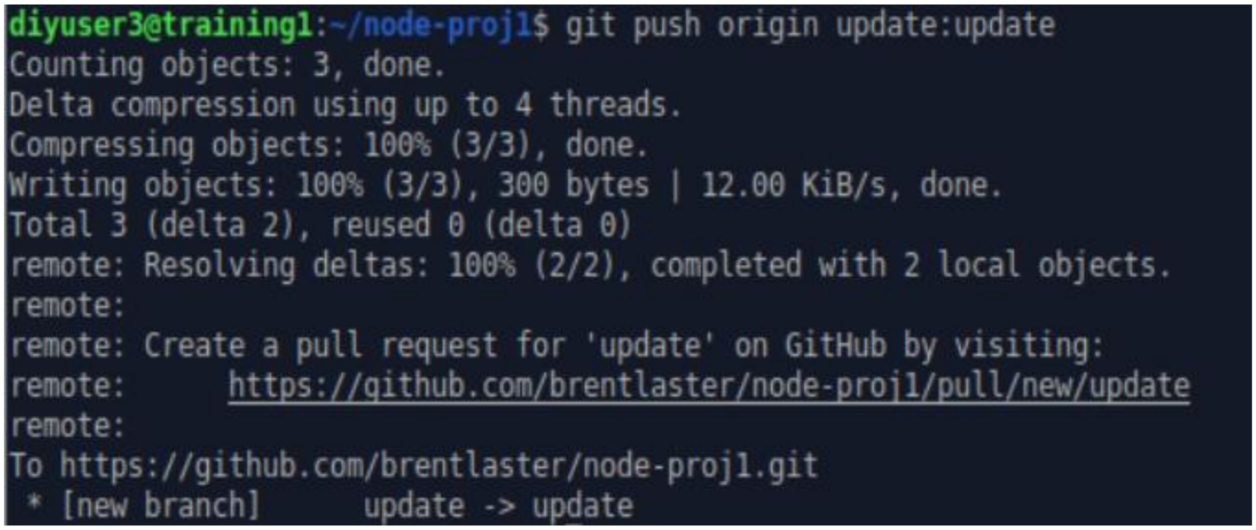


```
<!doctype html>
<html class="no-js" lang="">

<head>
        <meta charset="utf-8">
        <title>Hello World : Jenkins X</title>
        <meta name="description" content="Jenkins X">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet" media="all" href="style.css" />
</head>

<body>
        <div class="cover">
                <div class="feature">
                        <h1>Hello again Node</h1>
                        <div class="brand">
                                Jenkins <strong>X</strong>
                        </div>
                </div>
        </div>
</body>

</html>
```

10. Save your changes and exit the editor. Then commit and push your change into GitHub.

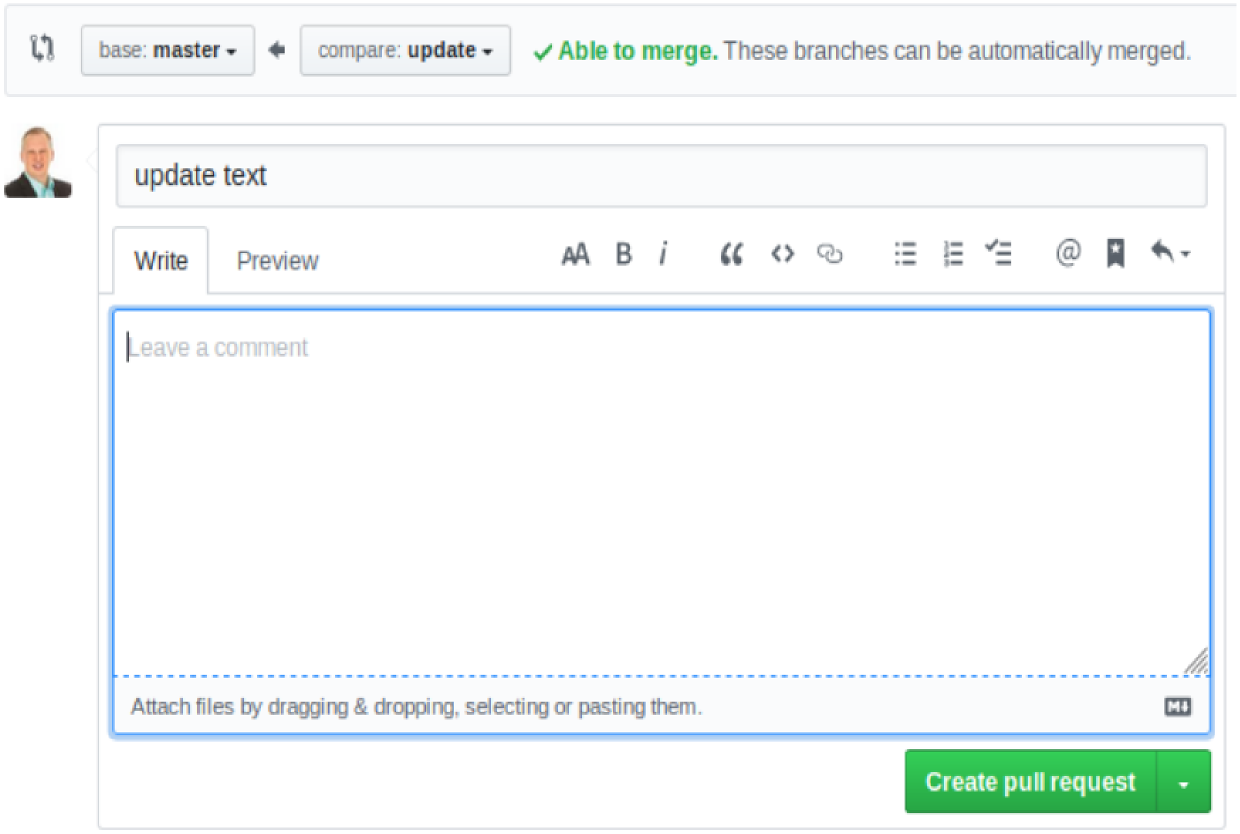
```
git commit -am "update text"

git  push  origin  update:update
```

11. When this completes, you should see a message in your output that tells you how to create a pull request. Go ahead and open the link that's provided for this.



```
diyuser3@training1:~/node-proj1$ git push origin update:update
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 300 bytes | 12.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'update' on GitHub by visiting:
remote:      https://github.com/brentlaster/node-proj1/pull/new/update
remote:
To https://github.com/brentlaster/node-proj1.git
 * [new branch]      update -> update
```

12. If you need to, login to your GitHub account. Then you'll be at the screen to create the pull request. Enter some text if you want in the "Update" box if you want and then click the green button to "Create pull request".



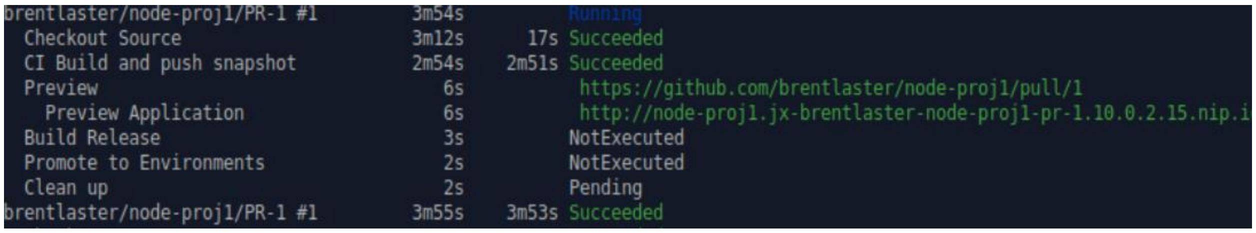
✓ Able to merge. These branches can be automatically merged.

update text

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

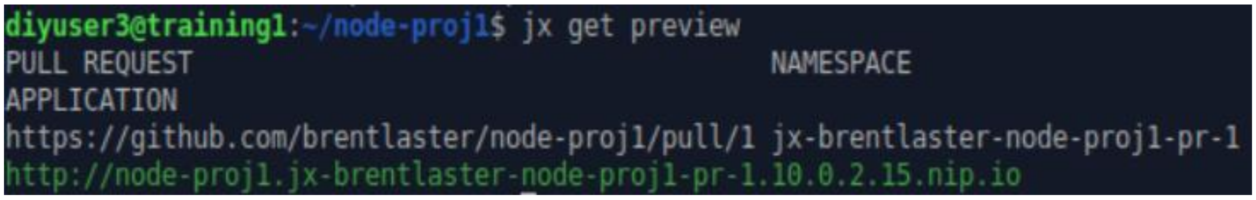
13. After you do this, Jenkins X should automatically create a preview environment for you. In the window where you have the watch activity running, you should see output shortly indicating that the preview environment was created and other activities around it are starting to take place.



```
brentlaster/node-proj1/PR-1 #1            3m54s          Running
  Checkout Source                         3m12s      17s Succeeded
  CI Build and push snapshot              2m54s    2m51s Succeeded
  Preview                                    6s           https://github.com/brentlaster/node-proj1/pull/1
    Preview Application                      6s           http://node-proj1.jx-brentlaster-node-proj1-pr-1.10.0.2.15.nip.i
  Build Release                              3s          NotExecuted
  Promote to Environments                    2s          NotExecuted
  Clean up                                   2s          Pending
brentlaster/node-proj1/PR-1 #1            3m55s    3m53s Succeeded
```

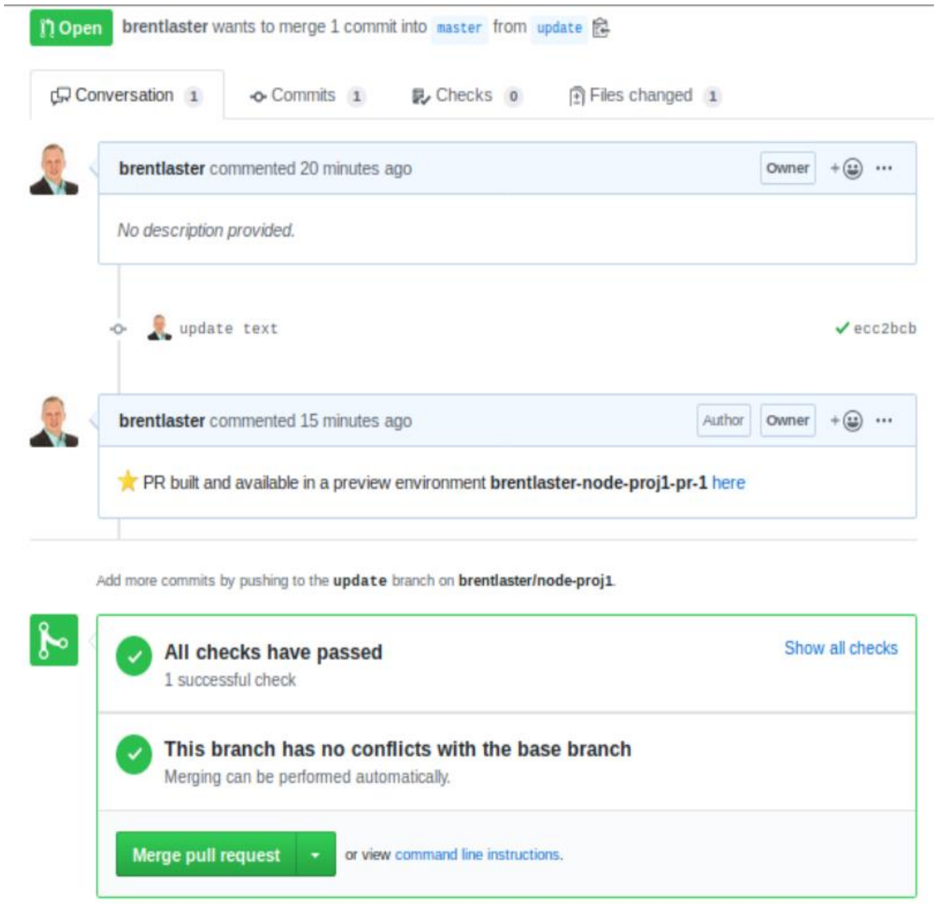
14. In the activity watch window, <u>wait</u> until you see the line about "Preview Application" (there may be others under it) . Then, in the terminal window, run the *jx get preview* command to see the preview environment that was setup for this and where the application running in the preview environment can be accessed. (Note: If the webhook piece doesn't work quickly, then you may need to wait about 5 minutes for the process timer to expire and move it through.)

jx get preview



```
diyuser3@training1:~/node-proj1$ jx get preview
PULL REQUEST                                      NAMESPACE
APPLICATION
https://github.com/brentlaster/node-proj1/pull/1 jx-brentlaster-node-proj1-pr-1
http://node-proj1.jx-brentlaster-node-proj1-pr-1.10.0.2.15.nip.io
```

15. Open the link under PULL REQUEST (the first one in the line) to open up the GitHub page for the Pull Request. You should see a screen that indicates the PR was built and is available in a preview environment.



Open brentlaster wants to merge 1 commit into `master` from `update`

Conversation 1 · Commits 1 · Checks 0 · Files changed 1

brentlaster commented 20 minutes ago — Owner

No description provided.

`update text` ✓ `ecc2bcb`

brentlaster commented 15 minutes ago — Author · Owner

⭐ PR built and available in a preview environment **brentlaster-node-proj1-pr-1** here

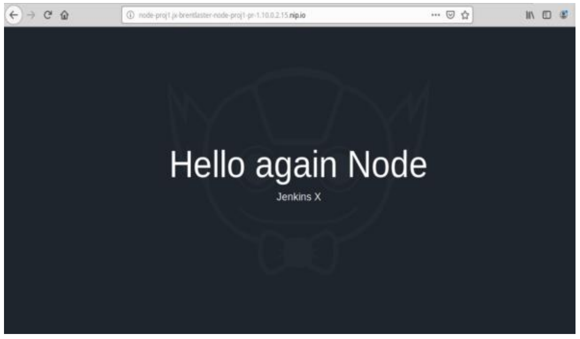
Add more commits by pushing to the **`update`** branch on **brentlaster/node-proj1**.

All checks have passed
1 successful check
Show all checks

This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request ▾ or view command line instructions.

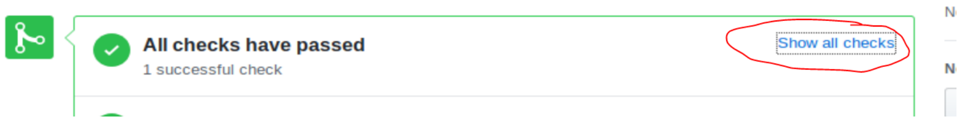
16. Right-click on the link that says “here” at the end of the line that has the star at the left and select “Open Link in New Tab”. This will take you to the running instance of your application in the preview environment. (You could have also gotten to it by clicking the second link in the output from the "jx get preview" command.)



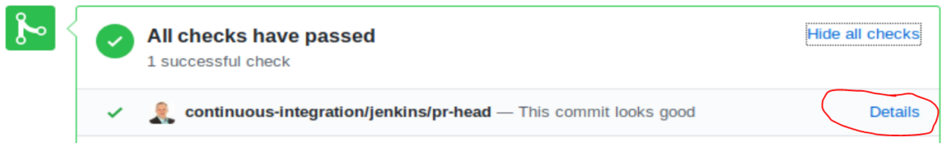
node-proj1.jx-brentlaster-node-proj1-pr-1.10.0.2.15.nip.io

Hello again Node
Jenkins X

17. Now go back to the previous page for the pull request and click on the “Show all checks” link in the window that says “All checks have passed”. When that expands, click on the “Details” link that comes up and then the “Open Link in New Tab” entry.



All checks have passed
1 successful check
Show all checks

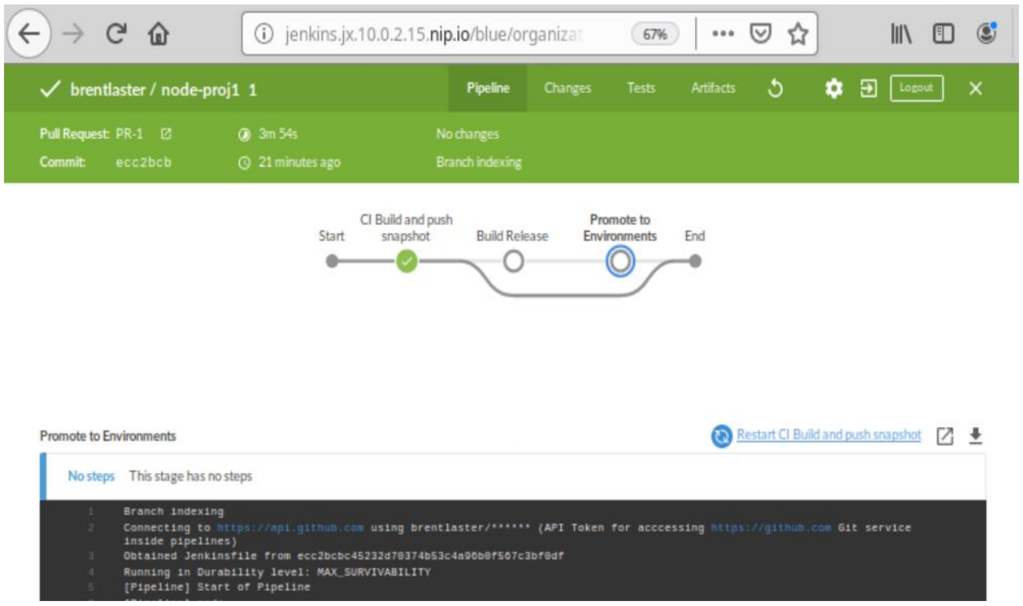


All checks have passed

1 successful check

continuous-integration/jenkins/pr-head — This commit looks good

18. Log into Jenkins again if needed with user "admin" and password "admin". You should now see the build for the preview environment.

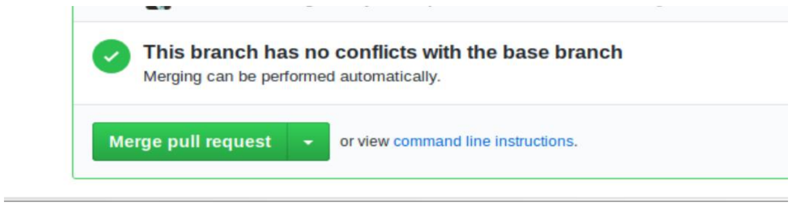


Promote to Environments

No steps This stage has no steps

```
Branch indexing
Connecting to https://api.github.com using brentlaster/****** (API Token for acccessing https://github.com Git service
inside pipelines)
Obtained Jenkinsfile from ecc2bcbc45232d70374b53c4a96b0f567c3bf0df
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
```

19. In the GitHub page for your PR that we were looking at in the last lab, go ahead and click on the big green button to "Merge pull request" and then confirm the merge.



This branch has no conflicts with the base branch

Merging can be performed automatically.

NOTE: If you do not see the expected versions in some of the commands in lab 5 or 6, check to see if your ultrahook.sh command that you started in the terminal at the start of chapter 4 is still running. Does it have any recent time responses? If not, you may need to stop it via CTRL-C and restart it via

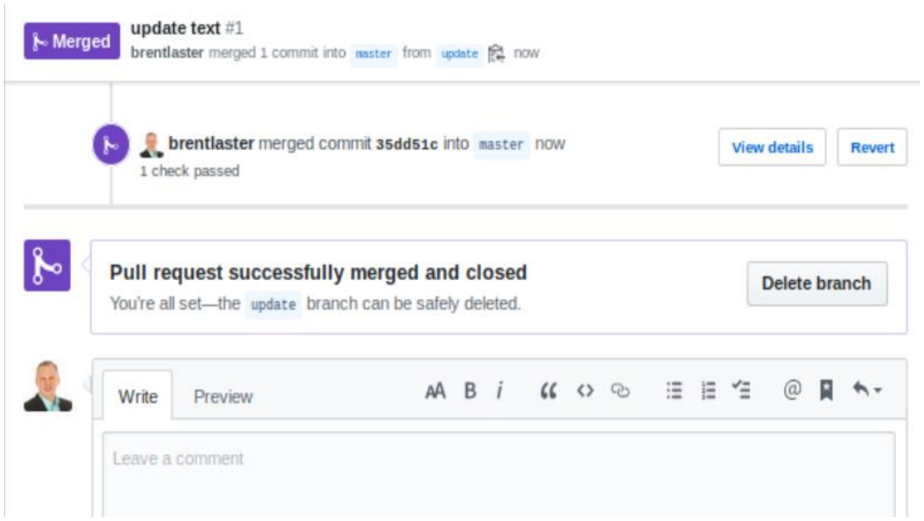
./runhook.sh <your GitHub userid>

The reason for the delay would be that the pipeline can't get updates via the webhook so it is stuck waiting for a timeout.

Lab 6: Promoting to Prod

In this lab, we'll take the version of our application that is in the Preview Environment and promote it to production.

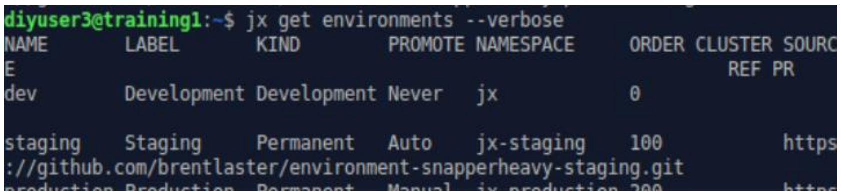
1. Looking at your pull request in GitHub, you should now see that it has been successfully merged and closed.


Merged
update text #1
brentlaster merged 1 commit into master from update now
brentlaster merged commit 35dd51c into master now
1 check passed
View details
Revert
Pull request successfully merged and closed
You're all set—the update branch can be safely deleted.
Delete branch
Write
Preview
Leave a comment

2. At this point, the app should have been promoted and built in staging. In the activity watch terminal, look for the line that starts with "Promoted" and shows "Succeeded" to know it has gotten this far. (If not there yet, wait for it.) In the original window, take a look at the environments you have again with the command below.

    jx get environments --verbose

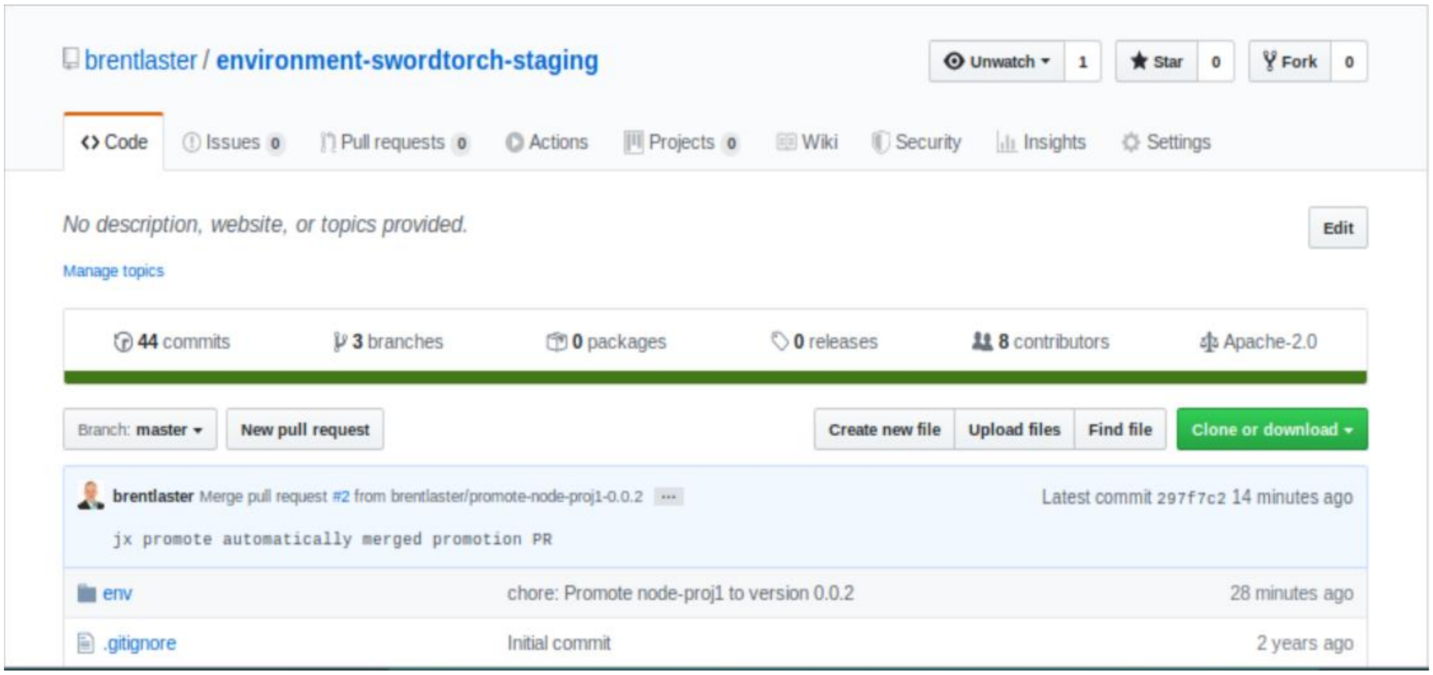
3. Look at the PROMOTE column. Notice that the Staging environment is set to "Auto" and the Production environments is set to "Manual". Open the output link for the GitHub staging area (the link in the line that starts with "staging") and look at the changes that happened in there.



```
diyuser3@training1:~$ jx get environments --verbose
NAME       LABEL       KIND        PROMOTE NAMESPACE     ORDER CLUSTER SOURC
E                                                                 REF PR
dev        Development Development Never   jx            0

staging    Staging     Permanent   Auto    jx-staging    100           https
://github.com/brentlaster/environment-snapperheavy-staging.git
```

4. In the line in the blue box that starts with "<github userid> Merge pull request #1..." notice the version number at the end. "-0.0.2". Where did that version number come from?

    Click on the three dots. "..." after the text on that line. Notice the message about "jx promote automatically merged promotion PR". What PR is it talking about?



No description, website, or topics provided.

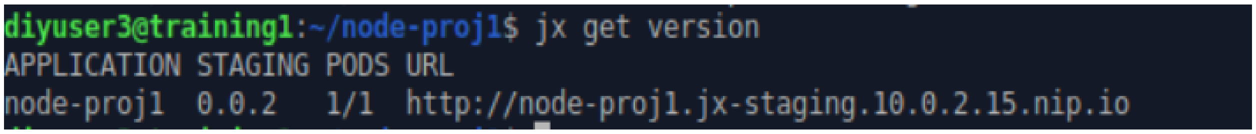
brentlaster Merge pull request #2 from brentlaster/promote-node-proj1-0.0.2 ... Latest commit 297f7c2 14 minutes ago

```
jx promote automatically merged promotion PR
```

env	chore: Promote node-proj1 to version 0.0.2	28 minutes ago
.gitignore	Initial commit	2 years ago

5. Take a look at the current version. In an available terminal window, type:

   jx get version



```
diyuser3@training1:~/node-proj1$ jx get version
APPLICATION STAGING PODS URL
node-proj1  0.0.2   1/1  http://node-proj1.jx-staging.10.0.2.15.nip.io
```

   <u>Take note of the version showing here - you will need it further down.</u>

6. There are other ways to get info about, and see, your application. Try the commands below.

   jx get applications
   jx open <application-name> -e staging

7. As we can see, our app was automatically promoted to staging. But as we saw when we looked at the promotion policies on the environments, we have to manually promote it to production.

8. To do this, in the original window, make sure you are in the app's directory and on the master branch. Then run the promote command to promote your app to production.

   cd ~/<app-name> (if not already there)
   git checkout master

   jx promote --version <version number from step 5> -e production

   (Don't worry about the WARNING: Failed to query here - it will resolve after a bit.)
   You may be asked about using your github userid as the user name to comment on issues. Just accept the default Yes answer.

9. Take a look at the current version.

```
jx get version
```

10. You should be able to see your app in the production environment by doing

```
jx open <app-name> -e production
```

THE END - THANKS!