Retrieve Jenkins X Activities, Logs, Pipelines, and More

This lesson explores how we can retrieve information concerning Jenkins X through the command line and the initial GitHub release created by Jenkins X.

We'll cover the following

While UIs are nice to look at, I am a firm believer that nothing beats the command line concerning speed and repeatability. Fortunately, we can retrieve almost any information related to Jenkins X through `jx` executable.

Retrieving activities

We can, for example, get the last activities (builds) of those jobs.

```
jx get activities
```

The output is as follows:

```
STEP                                                 STARTED AGO DURATION STATUS
vfarcic/environment-jx-rocks-staging/master #1          5m17s      59s Succeeded
  meta pipeline                                         5m17s      14s Succeeded
    Credential Initializer Cd47r                        5m17s       0s Succeeded
    Working Dir Initializer N5vv7                       5m17s       1s Succeeded
    Place Tools                                         5m16s       1s Succeeded
    Git Source Meta Vfarcic Environment Jx Roc ...      5m15s       4s Succeeded https://github.com
    Git Merge                                           5m11s       1s Succeeded
    Merge Pull Refs                                     5m10s       0s Succeeded
```

```
    Create Effective Pipeline                          5m10s       2s Succeeded
    Create Tekton Crds                                  5m8s       5s Succeeded
  from build pack                                       5m2s      44s Succeeded
    Credential Initializer Q4qvj                        5m2s       0s Succeeded
    Working Dir Initializer Gtj86                       5m2s       1s Succeeded
    Place Tools                                         5m1s       1s Succeeded
    Git Source Vfarcic Environment Jx Rocks St ...      5m0s       5s Succeeded https://github.co
    Git Merge                                          4m55s       1s Succeeded
    Setup Jx Git Credentials                           4m54s       1s Succeeded
    Build Helm Apply                                   4m53s      35s Succeeded
vfarcic/environment-jx-rocks-staging/PR-1 #1           6m22s      47s Succeeded
  meta pipeline                                        6m22s      12s Succeeded
    Credential Initializer Jdbwq                       6m22s       0s Succeeded
    Working Dir Initializer T54b2                      6m22s       0s Succeeded
    Place Tools                                        6m22s       1s Succeeded
    Git Source Meta Vfarcic Environment Jx Roc ...     6m21s       5s Succeeded https://github.co
    Git Merge                                          6m16s       0s Succeeded
    Merge Pull Refs                                    6m16s       1s Succeeded
    Create Effective Pipeline                          6m15s       3s Succeeded
    Create Tekton Crds                                 6m12s       2s Succeeded
  from build pack                                       6m7s      32s Succeeded
    Credential Initializer Ntjdr                         6m7s       0s Succeeded
    Working Dir Initializer 86qgm                        6m7s       1s Succeeded
    Place Tools                                          6m6s       1s Succeeded
    Git Source Vfarcic Environment Jx Rocks St ...       6m5s       4s Succeeded https://github.co
    Git Merge                                            6m1s       1s Succeeded
    Build Helm Build                                     6m0s      25s Succeeded
vfarcic/jx-go/master #1                                7m24s    2m26s Succeeded Version: 0.0.1
  meta pipeline                                        7m24s      18s Succeeded
    Credential Initializer J9wnj                       7m24s       0s Succeeded
    Working Dir Initializer Fs82g                      7m24s       2s Succeeded
    Place Tools                                        7m22s       2s Succeeded
    Git Source Meta Vfarcic Jx Go Master ...           7m20s       3s Succeeded https://github.co
    Git Merge                                          7m17s       1s Succeeded
    Merge Pull Refs                                    7m16s       0s Succeeded
    Create Effective Pipeline                          7m16s       3s Succeeded
    Create Tekton Crds                                 7m13s       7s Succeeded
  from build pack                                       7m4s     2m6s Succeeded
    Credential Initializer Fmc45                         7m4s       0s Succeeded
    Working Dir Initializer Vpjff                        7m4s       3s Succeeded
    Place Tools                                          7m1s       2s Succeeded
    Git Source Vfarcic Jx Go Master ...                6m59s      11s Succeeded https://github.co
    Git Merge                                          6m48s       1s Succeeded
    Setup Jx Git Credentials                           6m47s       0s Succeeded
    Build Make Build                                   6m47s       6s Succeeded
    Build Container Build                              6m41s       2s Succeeded
    Build Post Build                                   6m39s       1s Succeeded
    Promote Changelog                                  6m38s       4s Succeeded
    Promote Helm Release                               6m34s       5s Succeeded
    Promote Jx Promote                                 6m29s    1m31s Succeeded
  Promote: staging                                     6m24s    1m26s Succeeded
    PullRequest                                        6m24s    1m26s Succeeded  PullRequest: htt
    Update                                             4m58s       0s Succeeded
```

We can see that there were activities with each of the three jobs. We had one deployment to the production environment (`environment-jx-rocks-production`), and two deployments to staging (`environment-jx-rocks-staging`). The first build (activity) is always performed when a job is created. Initially, environments only

contain a few applications necessary for their correct operation. The reason for the second build of the staging environment lies in the creation of the *jx-go* project.

One of the steps in its pipeline is in charge of promoting a successful build to the staging environment automatically. When we explore jenkins-x.yml in more detail, you'll get a better understanding of the process, including promotions.

The last activity is of the *jx-go* pipeline. So far, we did not push any change to the repository, so we have only one build that was run when the job itself was generated through the quickstart process.

Applying filters on activities

Listing the most recent activities is very useful since we have only a few pipelines, but when their number grows, we'll need to be more specific. For example, we might want to retrieve only the activities related to the *jx-go* pipeline.

```
jx get activities --filter jx-go --watch
```

This time, the output is limited to all the activities related to *jx-go* which, in our case, is a single build of the *master* branch.

```
STEP                                        STARTED AGO DURATION STATUS
vfarcic/jx-go/master #1                           9m53s    2m26s Succeeded Version: 0.0.1
  meta pipeline                                   9m53s      18s Succeeded
    Credential Initializer J9wnj                  9m53s       0s Succeeded
    Working Dir Initializer Fs82g                 9m53s       2s Succeeded
    Place Tools                                   9m51s       2s Succeeded
    Git Source Meta Vfarcic Jx Go Master ...      9m49s       3s Succeeded https://github.com/vfar
    Git Merge                                     9m46s       1s Succeeded
    Merge Pull Refs                               9m45s       0s Succeeded
    Create Effective Pipeline                     9m45s       3s Succeeded
    Create Tekton Crds                            9m42s       7s Succeeded
  from build pack                                 9m33s     2m6s Succeeded
    Credential Initializer Fmc45                  9m33s       0s Succeeded
    Working Dir Initializer Vpjff                 9m33s       3s Succeeded
    Place Tools                                   9m30s       2s Succeeded
    Git Source Vfarcic Jx Go Master ...           9m28s      11s Succeeded https://github.com/vfar
    Git Merge                                     9m17s       1s Succeeded
    Setup Jx Git Credentials                      9m16s       0s Succeeded
    Build Make Build                              9m16s       6s Succeeded
    Build Container Build                         9m10s       2s Succeeded
    Build Post Build                               9m8s       1s Succeeded
    Promote Changelog                              9m7s       4s Succeeded
    Promote Helm Release                           9m3s       5s Succeeded
    Promote Jx Promote                            8m58s    1m31s Succeeded
  Promote: staging                                8m53s    1m26s Succeeded
    PullRequest                                   8m53s    1m26s Succeeded  PullRequest: https://g
    Update                                        7m27s       0s Succeeded
```

This time, we used the `--watch` flag to tell Jenkins X that we'd like to *watch* the

This time, we used the `--watch` flag to tell Jenkins X that we'd like to *watch* the activities. Since there are no pending builds, the output will stay intact, so please press *ctrl+c* to stop the watch and return to the prompt.

> 🔍 Internally, Jenkins X activities are stored as Kubernetes Custom Resources (CRDs). If you're curious, you can see them by executing `kubectl --namespace jx get act`.

Retrieving logs

Activities provide only a high-level overview of what happened. When everything is successful, that is often all the information we need. However, when things go wrong and some of the tests fail, we might need to dig deeper into a build by retrieving the logs.

```
jx get build logs
```

Since we did not specify from which build we'd like to retrieve logs, we are faced with the prompt to select the pipeline from which we'd like to extract the output. We could choose one of the pipelines, but we won't do that since I want to show you that we can be more specific in our request for logs.

> **Please press *ctrl+c* to return to the prompt.**

Applying filters on logs

We can use the `--filter` argument to retrieve logs from the last build of a specific pipeline.

```
jx get build logs --filter jx-go
```

The output should show the logs of the last build of *jx-go*, no matter the branch.

We can be even more specific than that and request logs from the specific GitHub user, of the specific pipeline, from the last build of a specific branch.

```
jx get build logs \
    --filter $GH_USER/jx-go/master
```

The output should show the logs of the last build of the *jx-go* pipeline initiated by a commit to the *master* branch.

Being able to retrieve logs from a specific pipeline is not of much use if we don't know which pipelines we have. Fortunately, we can extract the list of all the pipelines as well.

Retrieving pipelines

```
jx get pipelines
```

We can see that there are three pipelines named *environment-jx-rocks-production*, *environment-jx-rocks-staging*, and *jx-go* (I'll ignore the existence of the `dummy` pipeline). The first two are in charge of deploying applications to staging and production environments. Since we are using Kubernetes, those environments are separate namespaces. We'll discuss those two later. The third job is related to the *jx-go* project we created as a quickstart.

Retrieving applications

We can also retrieve the list of applications currently managed by Jenkins X.

```
jx get applications
```

The output is as follows:

```
APPLICATION STAGING PODS URL
jx-go       0.0.1   1/1  http://jx-go.jx-staging.34.206.148.101.nip.io
```

For now, retrieving the applications is uneventful since we have only one deployed to the staging environment.

Retrieving environments

So far, we talked about the staging and production environments. Are those the only ones we have? Let's check it out.

```
jx get env
```

The output is as follows:

```
NAME       LABEL       KIND        PROMOTE NAMESPACE     ORDER CLUSTER SOURCE
dev        Development Development Never   jx            0
staging    Staging     Permanent   Auto    jx-staging    100           https://github.com/vfarcic/
production Production  Permanent   Manual  jx-production 200           https://github.com/vfarcic/
```

As you can see, there is a third environment named `dev`. We'll explore it later. For now, remember that its purpose is true to its name, it is meant to facilitate development.

Retrieving applications from a specific environment

Now that we know which environments we have, we can combine that information and list only the applications in one of them. Let's see which ones are running in the staging environment.

```
jx get applications --env staging
```

The output is as follows.

```
APPLICATION STAGING PODS URL
jx-go       0.0.1   1/1  http://jx-go.jx-staging.34.206.148.101.nip.io
```

We already knew from before that the *jx-go* application is running in staging and we already know that nothing is installed in production. Nevertheless, we can confirm that with the command that follows.

```
jx get applications --env production
```

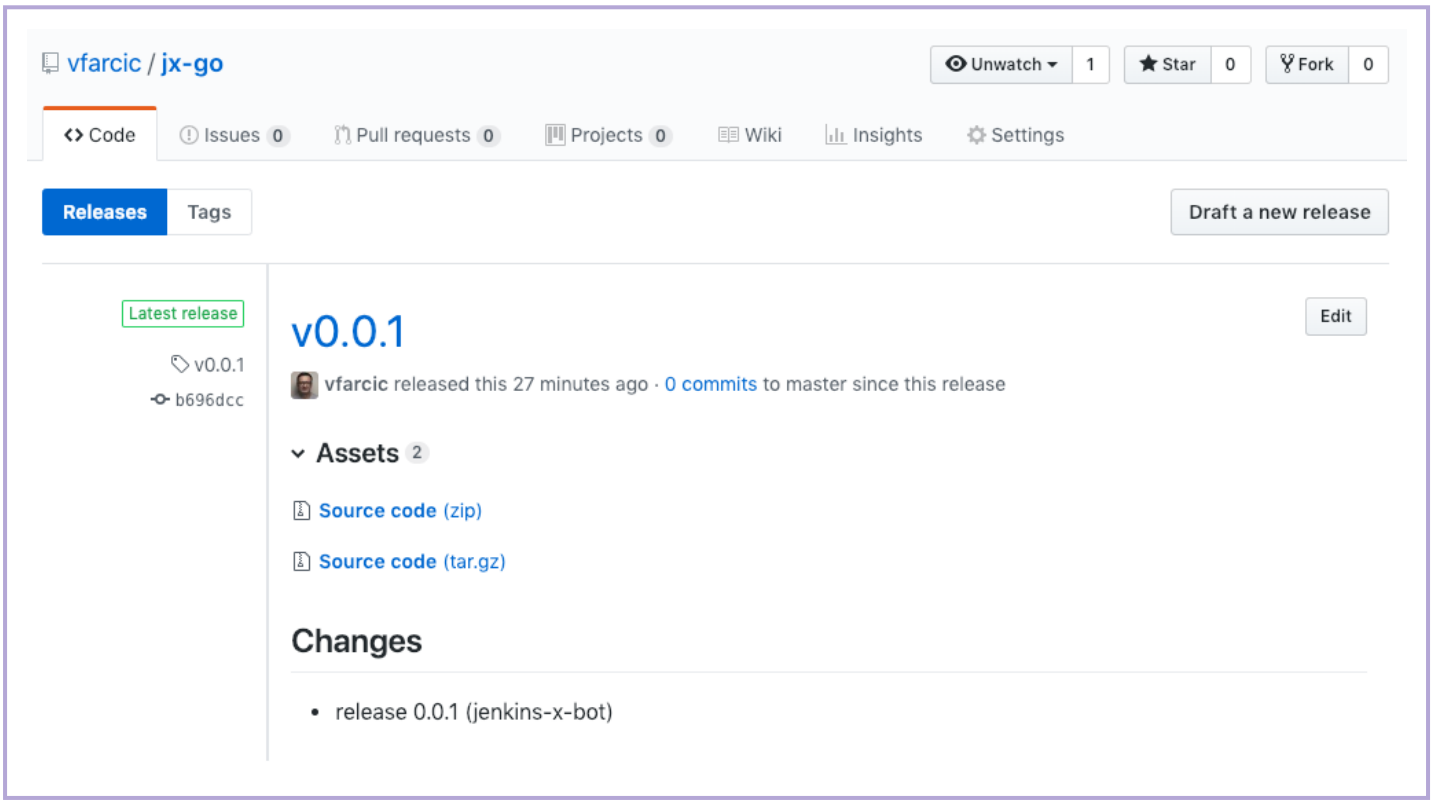
It should come as no surprise that the output states that `no applications` were `found in environments production`. We did not promote anything to production yet. We'll do that later.

Exploring project releases

Finally, Jenkins X also created a GitHub release for us. We can confirm that by going to project releases.

```
open "https://github.com/$GH_USER/jx-go/releases"
```

For now, we have only one release that is not very descriptive, since we did not create any issues that should be listed in release notes. The release you see in front of you is only the initial one created by pushing the quickstart files to Git.



Finally, we have not yet confirmed whether the new application is indeed deployed and can be accessed.

```
ADDR=$(kubectl --namespace jx-staging \
    get ingress jx-go \
    -o jsonpath="{.spec.rules[0].host}")

curl "http://$ADDR"
```

We retrieved the host from `jx-go` Ingress and used it to send a `curl` request. As a result, the output is:

```
Hello from: Jenkins X golang http example.
```

We confirmed that the application created as a quickstart was deployed to the staging environment. All we did was execute `jx create quickstart` and Jenkins X did most of the heavy lifting for us.

Let's wrap up this discussion in the next lesson.