Onboarding Best Practices: JFrog Pipelines

Overview

JFrog Pipelines empowers software teams to ship updates faster by automating DevOps processes in a continuously streamlined and secure way across all their teams and tools. Encompassing continuous integration (CI), continuous delivery (CD), infrastructure and more, it automates everything from code to production. Pipelines is natively integrated with the JFrog Platform and is available with both cloud (software-as-a-service) and on-prem subscriptions.

This video will help you to get started quickly with JFrog Pipelines by showing you how to connect Pipelines to GitHub as your source control system. This is one of the many out-of-box integrations that you can use to connect Pipelines to the DevOps tools you're already using. You'll see how the visual and interactive representation of your pipelines can be used to examine the execution status of each step, and how to examine the execution log of a step to diagnose and fix errors. Pre-built steps, called Native Steps, give you a streamlined way of specifying your most common DevOps tasks. The video shows how you can choose static or dynamic node pools for running your pipelines.

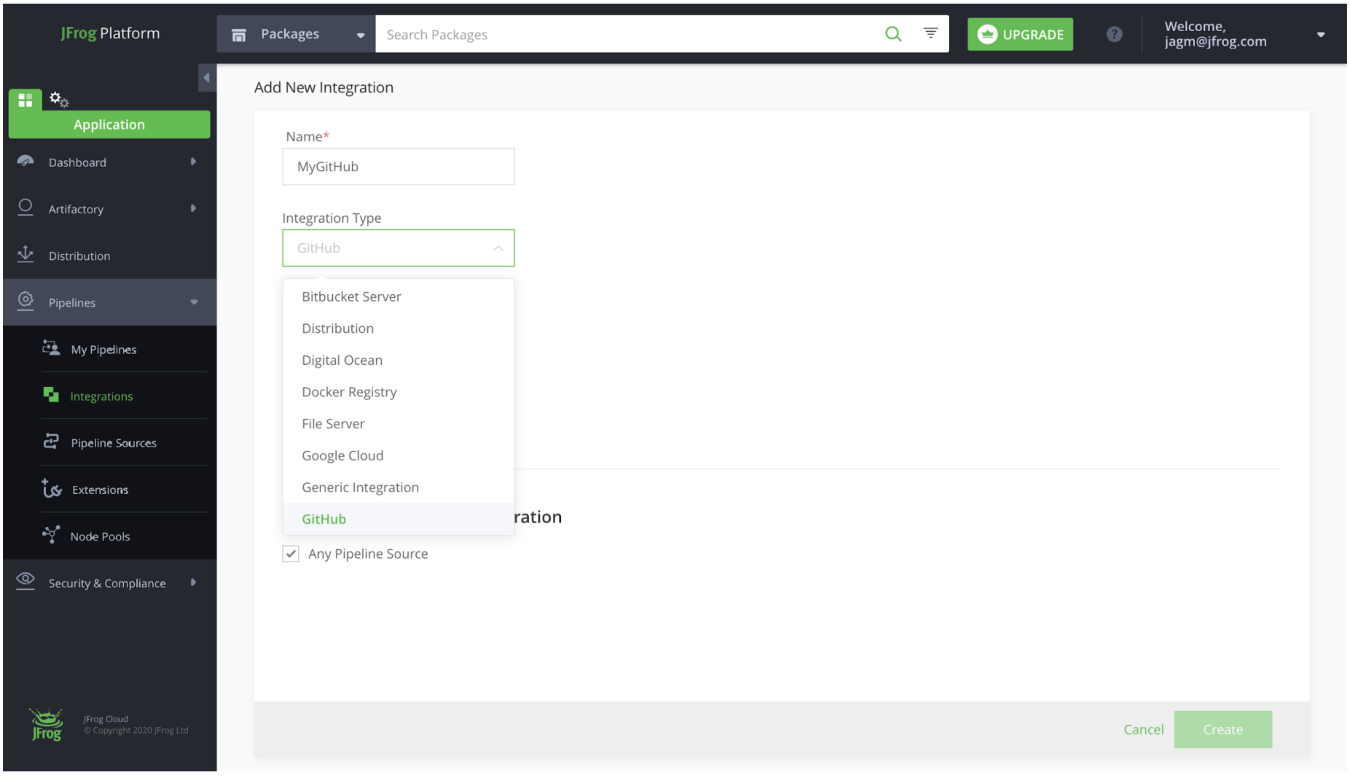
Page Contents

Onboarding Pipelines Best Practices

JFrog Pipelines is the next generation CI/CD solution for DevOps automation. It is part of the JFrog Platform, bringing together Artifactory, Xray and Distribution into a unified end-to-end system for one-stop DevOps. Through a single pane of glass, administrators can manage user and groups permissions to control who sees what.

Integrations

Pipelines comes with several out-of-the-box integrations for the DevOps tools you are likely to use most, enabling you to quickly add connections to services such as version control systems, cloud providers, Docker and Kubernetes.



Pipelines DSL

You code your pipeline in the YAML-based Pipelines DSL, and store it in your Git-compatible source code repository, such as GitHub. In this DSL, you declare the resources and steps of your pipeline.

Pipelines native steps provide you with a streamlined way to specify your most common DevOps tasks, such as building and pushing a Docker image, publishing buildinfo, and promoting between artifact repositories.

You'll tell Pipelines where your DSL file is stored by adding the repo as a Pipeline Source, connecting it through a Pipelines integration. Once you do, Pipelines will automatically sync with the source to load and process your YAML DSL file.

Every time you commit a change to your YAML DSL file, Pipelines will sync from that source to reload it.

49 lines (45 sloc) | 1.29 KB Raw Blame

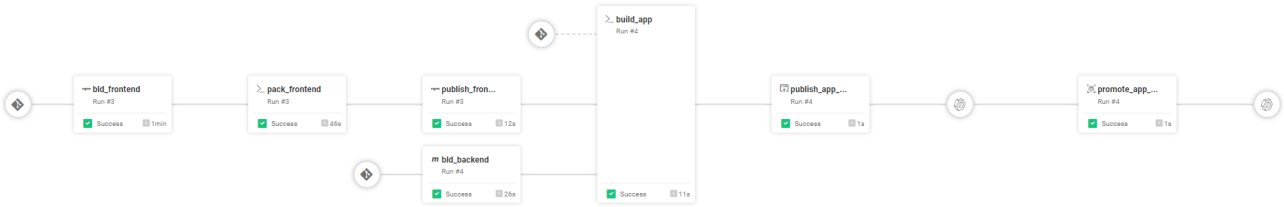
```yaml
resources:
  - name: myDockerRepo
    type: GitRepo
    configuration:
      gitProvider: my_github
      path: jagdishmirani/docker-build-example
    branches:
      include: master

  - name: build_info_1_docker_build_push
    type: BuildInfo
    configuration:
      sourceArtifactory: myArtifactory
      buildName: svc_build
      buildNumber: 1

pipelines:
  - name: pipeline_docker_build_push
    steps:
      - name: docker_build
        type: DockerBuild
        configuration:
          affinityGroup: dockerGroup
          dockerFileLocation: .
          dockerFileName: Dockerfile
          dockerImageName: 'pipe-master.jfrog.info:8081/docker-local/alpine37'
          dockerImageTag: ${run_number}
          inputResources:
            - name: myDockerRepo
          integrations:
            - name: myArtifactory

      - name: docker_push
        type: DockerPush
        configuration:
          affinityGroup: dockerGroup
          targetRepository: docker-local
          integrations:
```

Viewing and Running Pipelines

Viewing your pipeline presents you with an interactive diagram that visualizes your workflow sequence.

A CI build can trigger on a change to your source code, such as a new commit of any or selected files. Or you can manually trigger your pipeline from the diagram and watch it execute.

Your pipelines can be as complex as you need, and can trigger from multiple sources enabling it to flow through different paths.



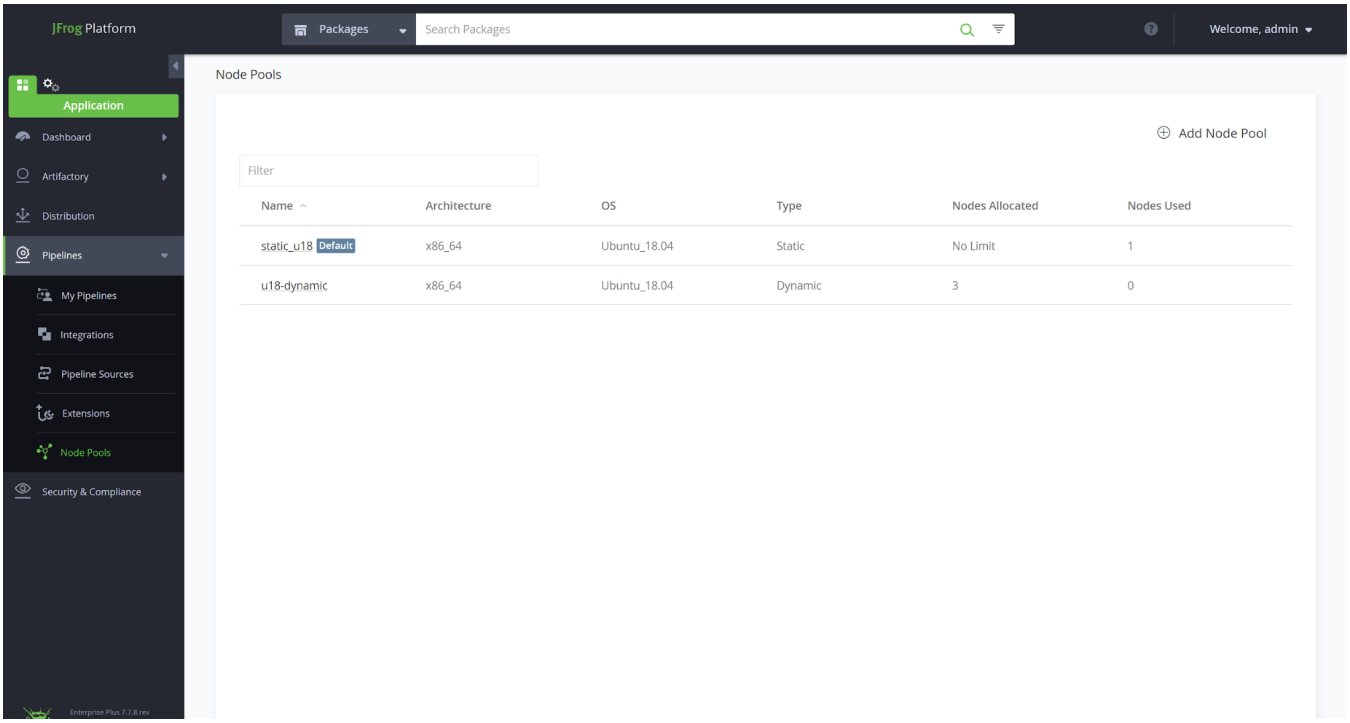
Every time your pipeline is run, Pipelines logs a record of its execution. You can see when every run occurred, and whether it was successful.

You can view the detailed log of every run, even for runs currently executing. In the run log, you can examine the actions and results of every step in your pipeline. This enables you to diagnose any errors in a failed step, or to watch the output of a currently executing step in real time.

Node Pools

Your pipelines run in build nodes that you provide grouped into node pools. The nodes in a pool run the same Host OS, so that multiple pipeline steps can execute in parallel.

You can create pools of static nodes, which are always-available machines on-prem or in the cloud, or dynamic pools, where nodes are spun up and down in a cloud service as they're needed, helping you to save costs.



What's Next

You can start exploring Pipelines through our quick starts and the JFrog Platform documentation.