Managing Runtimes

Overview

Every step in your pipeline executes on a build node that has been provisioned with a runtime environment. Through Pipelines DSL, you can control which runtimes your steps execute in.

By breaking up your pipelines into steps, Pipelines can distribute the work to wherever it can get done on the network. An administrator user must configure Pipelines with sets of build nodes (virtual machines) that are available for executing steps.

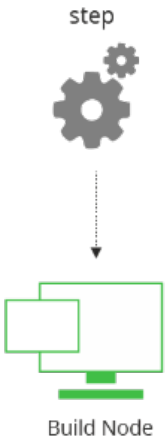
Pipelines sends each step to a build node to execute in a **runtime**, which is either the build node's virtual machine or a container provisioned with a Docker image on the build node. The runtime must have the necessary OS, software tools, packages, and configurations that the step needs to execute.

- **Running steps in series**: When steps in a pipeline must execute on the same build node, they can be assigned to the same **affinity group**. For more information, see Running multiple steps on the same build node.
- **Running steps in parallel**: Steps can be run in parallel to speed up pipeline execution. For more information, see Breaking Your Pipelines into Steps.

Host Runtimes

A traditionally designed CI server distributes tasks to other machines on the network, each of which has been configured with the OS, tools, and credentials that tasks need to perform their work. Each task executes in the runtime environment of a machine.

Pipelines can, when explicitly directed, execute any step this way, running it directly on the host build node in the runtime of its machine image. This may be appropriate when you need to perform a task from the baseline environment of the machine.



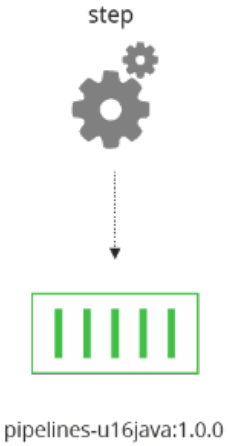
While efficient, executing directly on the host is very inflexible. If the task requires another tool on the VM, the task must either load it itself or a systems administrator must configure it for the developer in advance. When an organization needs to support many teams, staging environments, and DevOps activities, this can quickly become difficult and expensive to manage. Also, changes made to the host VM will persist for future steps running on the same host, leading to unpredictability.

Container Runtimes

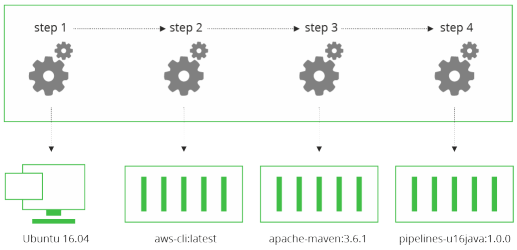
For maximum flexibility, Pipelines' preferred mode is to execute its steps in containers. For each step, Pipelines spins up a container on a build node using a Docker image that includes the runtime environment.

Page Contents

Page Contents



Through this method, each step executes with the runtime it needs to perform its particular set of work. For example, it can have the binaries and libraries needed for a language, and/or it can have tools and CLIs that the step needs to perform automated tests.



This system offers several key benefits:

- Each step executes with the tools it needs, and no others, configured with the specific settings for that environment.
- Steps in the same pipeline can execute in different runtimes, if they need a unique configuration or set of tools.
- Steps can execute simultaneously in different build nodes (when available), for faster build times.
- Pipelines are fully repeatable, always executing using the same set of immutable, versioned Docker images.
- Different teams can build with different tools and configurations without impacting each other.
- Pipelines can reliably execute with the runtime (including credentials) appropriate for the build environment, whether for Dev, Test, or Production.
- Execute pipelines on whichever infrastructure suits your needs best: on virtual machines hosted by a cloud provider or on servers in your own datacenter.

JFrog provides a set of standard Docker images of runtimes for supported architecture, OS, and language combinations. When the Pipelines DSL doesn't specify a runtime, Pipelines provisions the build node with one of these runtime images as best matches the build node environment.

Concepts

A step runtime is made up of the following components:

Nodes

To run any step in a pipeline, you need a build node (a virtual machine) where the step will execute.

An administrator user must provide nodes and attach them to Pipelines in the JFrog Platform Deployment. A node can be on any infrastructure that you choose to use, whether it is from a cloud provider (such as AWS, GCP, or Azure), or on your own infrastructure if your security policies require your operations to remain behind your own firewall.

Node Pools

Node Pools logically group nodes to make them available to execute the steps in a pipeline. This enables an administrator user to group nodes according to their processor architecture and baseline operating system. It enables pipelines to run on specific node pools, and to run steps simultaneously on different build nodes.

Node pools can contain nodes of two distinct types:

Static Nodes

Static nodes are configured and provided by administrator users to a node pool. They operate in perpetuity, and are available to execute steps at any time. Static nodes are especially useful if you need to run your operations on build nodes in your own data center. You may need to do this if you have security policies that prohibit your code from leaving your firewall, or if your jobs require access to internal resources that are not accessible from the internet. You can also attach build nodes from a cloud provider, although you will incur charges even when the build node is idle.

Dynamic Nodes

Dynamic nodes are on-demand compute environments that are spun up on a cloud provider when needed, and destroyed when seen to be idle. This is an efficient method that helps minimize compute costs. Dynamic nodes are connected through a dynamic node integration that connects to an IaaS provider such as Amazon or Google.

Runtime Images

A runtime image is a preconfigured Docker image that includes all the necessary components and settings to run your pipeline steps in a container.

In This Section

This section includes the following documentation: