

CODING Continuous Integration Introduction

Product Documentation





Copyright Notice

©2013-2024 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice

🔗 Tencent Cloud

All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

Contents

Introduction

Overview

Benefits

Scenarios

Introduction Overview

Last updated : 2023-12-29 11:44:51

Continuous Integration Overview

Continuous integration (CI) plays an important role in modern software development. In normal projects, some of the work is mechanical and repetitive, but prone to errors (such as during packaging and deployment). Users can instruct machines to complete this work with the click of a mouse and then take a break while the CI build plan automatically performs unit testing, code inspections, compilation and building, contract testing, and even deployment. This significantly reduces the workload of developers while improving code quality and development efficiency.

CODING Continuous Integration (CODING-CI)

CODING Continuous Integration (CODING-CI) is fully compatible with Jenkins' continuous integration services. It supports popular programming languages, such as Java, Python, and Node.js, and building Docker images. Its graphical orchestration, high-spec clusters, and multi-plan parallel builds help you comprehensively accelerate your build tasks. It supports mainstream Git code repositories, including CODING Code Repositories (CODING-CR), GitHub, and GitLab. In terms of build dependency pull, it has dedicated network optimizations for major image sources such as Maven and npm to ensure a high pull speed and further accelerate the build process.

Function Guide

The core components of continuous integration are build plans, which are displayed in cards. The buttons are described below:

Overview	Build Job	1 Build Now
Collaboration	My Stars System Source All Ungrouped More *	2 Setting
Repository	Trigger: All - Plan Source: Customize - Q	3 Latest build task information
Code Scanner beta >		
∞ CI ~	express-docker O ··· Test \diamond O ···	
Build Job		
	Suild succeeded.	
Artifact Management	Manually triggered by Testor Manually triggered by Testor	
Test Management >		
Document >		
🌣 Settings 🛛 «		

Hover your cursor over a chart to display button descriptions.

	Overview	Build Job
≻</th <th>Collaboration Repository</th> <th>My Stars System Source All Ungrouped More -</th>	Collaboration Repository	My Stars System Source All Ungrouped More -
(b) (c)	Code Scanner beta >	Build Now
	Build Job	express-docker 🔯 🖸 …
Ŷ	Build Node	Build succeeded.
	Artifact Management >	% 1
	Document >	
۵	Settings 《	



On the build records page, users can conveniently set build plans and filter build records.

uild Job	🗴 express 🏠 🛧 coding	😩 Shanghai 🚺 💿 Status Badge	Scheduled Trigger	🔉 Cache 🛛 🌣 Settings	• Build Now	 Code source and build nod Filter records built based or tag/branch/revision
igger: All v Plan Source: Customize v Search Q	Only Me 🔵 Filter: All 🗸 🤮	•				
	Status	Trigger Info	Time Information	Quick View	Oper ation	
express-docker D	 Build succeeded. 	Manually triggered by Tester #7 \$° master > 0045b7e	8 minutes ago 54 seconds	4 @ 11 &		
Build succeeded.	Automatically cancelled (duplic ate version number)	Manually triggered by Tester #6 \$2 master - 0045b7e	🖹 8 minutes ago 🕓 –	" o 1]		
Manually triggered by Testor	Build succeeded.	Manually triggered by Tester #5 \$2 master 0045b7e	8 minutes ago 3 53 seconds	*: @ 11 J		
	Build image and push to CODI NG Docker AR / Aborted by	Manually triggered by Tester #4 \$2 master - 0045b7e	9 minutes ago 44 seconds	n (1		
	Build succeeded.	Manually triggered by Tester #3 0045b7e	i0 minutes ago	" @ 11 ā		
	1–7. Total: 7.			Entries per pa	ge 15 👻 🚺 1	

In a single build record, you can follow the link to quickly go to the build branch and revision version. In the **Quick View** area, you can follow the link to view the build process, build snapshot, and change history.

My Stars System Source All Ungrouped More * Trigger: All ~ Plan Source: Customize ~ end ~ en	Build Job	express 🛱 🛧 CODING	🧐 Shanghai 🛛 🙆 Status Badge	Scheduled Trigger	🗘 Cache 🛛 🌣 Settings	Build Now
staus Trigger Info Time Information Quidk View Oper ation express-docker • • • • <t< th=""><th>My Stars System Source All Ungrouped More *</th><th>Only Me</th><th></th><th></th><th></th><th></th></t<>	My Stars System Source All Ungrouped More *	Only Me				
express-docker		Status	Trigger Info	Time Information	Quick View	Oper ation
• Automatically cancelled (duplica Manually triggered by GP199513 • It hours action number) • Dull Process • It is hours action • Outbore	express-docker • ···	Suild succeeded.	Manually triggered by GP199513 #7 \$* master 0045b7e	18 hours ago54 seconds	" @ 11 ā	
Manually triggered by Testor Image: Second secon	Build succeeded.	Automatically cancelled (duplica te version number)	Manually triggered by GP199513 #6	🗎 18 hours ag o () –	e کا Build Process	
Build image and push to CODI Manually triggered by Tester Image and push to CODI NG Docker AR / Aborted by #4 ½ master ~ 0045b7e Image and push to CODI Build succeeded. Manually triggered by Tester Image and push to CODI 1-7. Total: 7. Image and push to CODI Manually triggered by Tester	Manually triggered by Testor	Build succeeded.	Manually triggered by GP199513 #5 & master -> 0045b7e	🗎 18 hours ago 🕔 53 seconds	° 0 11 4	
Build succeeded. Manually triggered by Tester #3 \-> 0045b7e 10 minutes aqo 1 I minutes aqo 11 I minutes aqo 1 I minutes I minutes aqo 1 I minutes I minu		Build image and push to CODI NG Docker AR / Aborted by	Manually triggered by Tester #4 \$ master -> 0045b7e	9 minutes ago 44 seconds	4 o 11	
1–7. Total: 7. Entries per page 15 – 1		Suild succeeded.	Manually triggered by Tester #3 0045b7e	ago 10 minutes	r: 💿 ll Ā	
		1–7. Total: 7.			Entries per pa	ge 15 👻 🚺

Build plans

Build plans (jobs) are the basic units of continuous integration. You can open a build plan to set its code source, build process, trigger rules, environment variables, and notifications. In the future, this plan will be triggered based on the set rules to implement an automated build pipeline.

Build tasks

After configuring a build plan, each build execution generates a specific build task. You can go to a build task to see its build process, change history, test reports, build artifacts, build snapshot, and other execution information.

Jenkinsfiles

A Jenkinsfile defines a continuous integration pipeline, which implements stream encapsulation and management of steps. Pipelines are the basic units in continuous integration. Pipelines can be executed in series or parallel.

Benefits

Last updated : 2023-12-29 11:44:50

Benefits of CODING Continuous Integration (CODING-CI):

Comprehensive build types

In addition to Docker images, CODING-CI also supports the builds of JAR, APK, and other software packages. Plus, it has many preset build environment images such as Java, Python, and Node.js.

Parallel execution of multiple build plans

CODING-CI supports parallel builds of single projects to meet the high-level continuous integration needs of users. Its backend server cluster can schedule responsive computing resources according to user needs, ensuring that build tasks start quickly and spend less time queuing.

Cache acceleration

During the continuous integration and build process, the repetitive downloading of dependent files may result in a long build time. CODING-CI supports caching between different build tasks to accelerate repetitive builds by an average of 300%.

Graphical orchestration

In addition to manual build script editing, CODING-CI also offers high-quality graphical orchestration capabilities for greater ease of use. For each step of the build, it offers various build script templates that you can choose from, delivering an intuitive editing experience.

Full compatibility with Jenkins

The build scripts of CODING-CI have full syntactical compatibility with Jenkins, the most widely used continuous integration tool in the world, which means that you can seamlessly and easily migrate Jenkins builds to CODING.

Scenarios

Last updated : 2023-12-29 11:44:51

Auto building

CODING Continuous Integration (CODING-CI) can be used to automatically build code. Based on the user-defined build environment and build script instructions, it can quickly build the source code into runnable artifacts with the aid of the high computing and networking power of the cloud and store them in build version repositories.

Supplementary review

CODING-CI makes complicated yet important supplementary code review easier. It can perform prep work such as automated building, automated testing, style checks, quality scanning, and security assessments before the review starts, which greatly reduces the complexity and difficulty of the review work.

Automated testing

DevOps requires a high degree of automation for the software delivery process, in which automated testing is particularly important. CODING-CI can participate in automated tests at project release, code review, project acceptance, and other stages to improve the efficiency of the delivery process.