

GitLab (CI/CDパイプライン編)

矢野 哲朗







矢野 哲朗 ┎ <u>tetsurow.yano</u>







: システム運用 10年・ネットワーク 6年・SI 8年 近頃はNextcloud/Rancherを担当

: 全く上達しないRubyist

<u>一番最初のPCは、OKI if-800 でした…。</u>

<mark>ナ.ベース・</mark> レーション	これから始める企業のため ンテナ実践講座	のコ	00/05 07 00
2929年1409227444 2929年1400 2929年1400 2929年1400 2929年1400 2029 2029 2029年1400 2029 20	「これから始める企業のためのコンテナ実践講座」の連載記事一覧	Q&Aで早わかり 企業ユーザーのためのコンテナ活用法 真価は開発効率アップ インフラ主導では危う [第1回] なぜ、コンテナが必要なのか?	j9/25 07:00
		矢野 哲朗=スタイルズ、梶原 稔尚=スタイルズ	YSTEMS



GitLabの概要説明



GitLabとは?

GitLabは、Gitリポジトリをホスティングする オープンソースソフトウェアです

- Gitリポジトリの管理以外にも、Issue(課題)の管理、コードレビュー、Cl と CD (継続的デリバリ)を1つのツールとして統合的に提供しています
- GitHubと同様にソースコードを見たり、Issue(課題)管理を行ったり、 Wikiを書いたり、おおよそ開発に必要な機能が含まれています。これらの 機能をウェブブラウザーで利用することができます

Style_z3

GitLab.comとGitLab CE/EEの違い

□ GitLab.comについて

- GitLab.com はGithubのSaaSと同じWebサービス 無料版の制限がGitHubは、プライベートリポジトリが共同編集者が3人までなのに対し て、GitLab.comは無制限(機能制限あり)
- GitLab.comは、GitHubと同じように有料版があります。

GitLab CE/EE (= Community Edition/Enterprise Edition) について

GitLab CE/EEは、GitHub Enterpriseと同様に、自前のインフラにインストールして動かします。GitHubと違う所は、GitHub Enterpriseは有料のみですが、無料版のCommunity Editionがあります。



GitLabの機能について(無料版で使えるもののみ)

Gitリポジトリ管理
Issueトラッカー/ボード
GitLabパイプライン

□タイムトラッキング

ロサイクルアナリシス

□ 変更点をレビューするReviewApps

□GitLab ページ

□ Git LFS2.0対応

Wiki

- GitLab Container Registry
- GitLab ウェブエディター
- □組み込みMattermost(オンプレ用)
- □Slack連携(ChatOps)
- Webhook
- □Kubernetes連携
- AutoDevOps
- Serverless
- □Snippet機能



お勧めのGitLab書籍

□ GitLab実践ガイド



GitLab実践ガイド - インプレスブックス https://book.impress.co.jp/books/1116101163



インフラCI実践ガイド - 翔泳社の本 https://www.shoeisha.co.jp/book/detail/9784798155128



GitLabパイプライン



GitLabパイプラインについて

GitLabパイプラインはどういったものなのか?

Jenkinsのようなジョブ実行機能

Cl (Continuous Integration) とCD (Continuous Deployment or Delivery) を実現する

- GitLab、GitLab Container Registryとの統合
- □ GitLabへのgit pushイベントで実行される
- □.gitlab-ci.yamlファイルに手続き的に記述
- □ 実行はGitLabサーバー外にあるRunnerが担当





□ GitLabパイプラインが実現するCI/CD

Stylez 3





GitLab CIのRunnerがコンテナを動かす仕組み



.gitlab-ci.yamlの概要説明



.gitlab-ci.yaml 概要

- YAMLで記述する(サンプルを後述)
 パイプラインの構造と順序を定義する
 GitLab Runnerで実行される
 プロセスの終了状況により次のジョブが実行されるかどうかを決定する
 ファイルの設置場所 gitリポジトリの直下に置くのがデフォルト(他の場所にも設置可) /.gitlab-ci.yaml
- □ GitLab内の各種環境変数を利用できる



ジョブ実行環境

□パイプライン実行例



ジョブ間で共有したいファイルはartifactsで指定

artifacts.pathsで指定されると指定されたpaths配下は別のジョブでも利用できる artifacts:

paths:

- binaries/
- .config:
- <キャッシュしたいファイルパスを指定>





Stylez 3

【Demo】リポジトリを作成



GitLab.com にサインイン

GitLab.com (<u>https://gitlab.com/users/sign_in</u>) にアクセスしサインイン

□ プロジェクトの画面が表示される

₩	
.ab.com	Sign in Register
ers free unlimited (private) repositories and unlimited	Username or email
lore projects on GitLab.com (no login needed) re information about GitLab.com ab com Support Forum	Password
b Homepage	Remember me Forgot your password?
or and by signing in to this service you accept our: policy	Sign in
om Terms.	Sign in with
	G Google 🎐 Twitter
	🖸 GitHub
	Remember me

Gitlab.com にアクセスし、「Username or email」と 「Password」を入力しサインインします。

プロジェクトの画面が表示されます。



Stylez 3

.gitlab-ci.yamlファイルの場所

□ファイルの設置場所 gitリポジトリの直下に置くのがデフォルト(他の場所にも設置可)

名前	最新コミット
.mvn/wrapper	Initial application generated by JHipster-5.8.2
myapps	Initial commit
src src	Initial application generated by JHipster-5.8.2
🖿 webpack	Initial application generated by JHipster-5.8.2
editorconfig	Initial application generated by JHipster-5.8.2
gitattributes	Initial application generated by JHipster-5.8.2
■ .gitignore	Initial application generated by JHipster-5.8.2
🖹 .gitlab-ci.yml	Update .gitlab-ci.yml



ソースコードのクローン(コピー)

- □ 演習用リポジトリにアクセス
- □ Forkをクリック
- □ 作成したアカウントを選択

演習用ソースコードはこちら

https://gitlab.com/stylez-co-jp/jhipster-hands-on-example



ソースコードのクローンが終わるまで待ちます



ソースコードクローンの確認

□ クローンが完了したメッセージを確認

🦊 GitLab プロジェクト 🗸	グループ ~ アクティビティー マイルストーン スニペット 🕢 🕒 🖬 🖓 検索またはジャンプ先 Q 🗋 🖞 🕑 🖓 😽 😜
J jhipster-hands-on- example	You won't be able to pull or push project code via SSH until you SSH 鍵を追加 to your profile 次回から表示しない 後で通知
合 プロジェクト	Uchida Yuma > jhipster-hands-on-example > 詳細
詳細	The project was successfully forked.
アクティビティー Releases サイクル分析	J jhipster-hands-on-example Project ID: 11706365 ゆ うイセンスを追加 ⇔5 コミット ピ 1ブランチ ⊘ 0 タグ ◎ 0パイトファイル
🖪 リポジトリ	フォーク元 stylez-co-jp / jhipster-hands-on-example
 D) 課題 0 1) マージリクエスト 0 	master v jhipster-hands-on-example / + v 履歴 Q ファイルを検索 Web IDE Q v
🦃 CI/CD	gitlab.comのレジストリヘイメージをPushするように変更 c5541b0d G YANO Tetsuro が 1週間前 にコミットしました Control Control
☞ 運用	
📮 レジストリ	 README Add CHANGELOG Add CONTRIBUTING Kubernetes クラスターを追加 CI/CD 設定

The project was successfully forked. と表示されればクローン完了です



【Demo】パイプラインの起動



.gitlab-ci.yml を開く

クローンした「jhipster-hands-on-example」をクリック □「.gitlab-ci.yml」をクリック

7	•		(
	\bigvee	ļ,	٦

GitLab.com Gold プランの無償試用

Try all GitLab has to offer for 30 days. クレジット カードは必要ありません



Your projects 1	Starred projects	0	Explore projects
Your projects 1	Starred projects	0	Explore projects

すべて 個人

jhipster-hands-on-example Maintainer

j jhipster-hands-o

example

サイクル分析 0 リボシトリ

■ マージリクエスト ●

合 プロジェクト

\$¥38

0 1935

& CI/CD

• 2011			
D DEXEU	名前	最新コミット	最終更新
P1 1966	.mvn/wrapper	Initial application generated by IHipster-5.8.2	1ヶ月前
Collapse sidebar	myapps	Initial commit	1ヶ月前
🖊 GitLab プロジェクトッ	グループ - アクティビティー マイルストーン	· スニペット 〇 ●奈志たはジャンプ先_	E Ø× 😑
j jhipster-hands-on- example	myapps	Initial commit	1ヶ月前
A 705-05	src src	Initial application generated by JHipster-5.8.2	1ヶ月前
0 707171	 webpack 	Initial application generated by JHipster-S.8.2	1ヶ月前
27.00	ditorconfig	Initial application generated by JHipster-5.8.2	1ヶ月前
101-2	gitattributes	Initial application generated by JHipster-5.8.2	1ヶ月前
サイクル分析	.gitignore	Initial application generated by JHipster-5.8.2	1ヶ月前
C UKSEU	📓 .gitlab-ci.yml	gitlab.comのレジストリヘイメージをPushするように変更	1ヶ月前
	huskyrc	Initial application generated by JHipster-5.8.2	1ヶ月前
O TEM 0	.prettierignore	Initial application generated by JHipster-5.8.2	1ヶ月前
□ マージリクエスト (0)	D prettierrc	Initial application generated by JHipster-5.8.2	1ヶ月前
📽 CI/CD	Jyo-rc.json	Initial commit	1ヶ月前
G 運用	HELM-README.md	Initial commit	1ヶ月前
D 69219	README.md	Initial application generated by JHipster-5.8.2	1ヶ月前
m with	D mondarings	Initial application annarated by Ulinster 5.8.2	100 840

③ README
④ Add CHANGELOG
④ CONTRIBUTINGを追加
 ④ Kubernetes クラスターを追加
 ⑤ CI/CD 設定

on't be able to pull or puth project code via SSH until you SSH 留意追加 to your p

ワイセンスを追加 や5コミット ド1ブランチ クロタグ 〇のパイトファイル

ihipster-hands-on-example / + ~

· pipster hands on example > 248

roiert (D: 11982803

フォーク元 stylez-co-jp / jhipster-hands-on-example

jhipster-hands-on-example U

gitlab.comのレジストリヘイメージをPushするように変更

YANO Tetsuro が 1ヶ月前 にコミットしました

下にスクロールさせて「.gitlab-ci.yml」を クリックします。



0 v 0 Star 0 Y Fork 0

課歴 Q. ファイルを検索 Web IDE Q 、

c554160d 3





これは今回 JHipster をビルドするパイプラインの 記述です。このパイプラインを動かすためには Git レポジトリへのコミットが必要になります。 .gitlab-ci.ymlの右の「編集」をクリックします。

編集画面の一番下まで画面をスクロールさせます。 最後の行で「Enter」を押し、ひとつ改行を入れます。

Apply a GitLab CI Yaml template

⇒ Soft wrap text

.gitlab-ci.vml

コミットメッセージ Update .gitlab-ci.yml



コミットメッセージを追加する

・ 画面下のコミットメッセージにメッセージを入力 ・ 左下の「Commit changes」をクリック

<pre>42 # Uncomment the following line to use gitlabs container registry. You need to adapt the REGISTRY_URL in case) 63 * docker-push: 64 image: docker:latest 65 * services: 66 - docker:dind 67 * variables: 68 REGISTRY_URL: SCI_REGISTRY 69 ITMAGE:TAG: SCI_REGISTRY_IMAGE:\$CI_COMMIT_REF_SLUG 70 stage: release 71 * dependencies: 72 - maven-package 73 * before_script: 74 - docker info 75 - cp target/*.war src/main/docker 76 * script: 77 - docker login -u "gitlab-ci-token" -p "\$CI_BUILD_TOKEN" \$REGISTRY_URL 78 - docker build -f src/main/docker/Dockerfile -t \$IMAGE_TAG src/main/docker</pre>	<pre>71 dependencies: 72 - maven-package 73 + before_script: 74 - docker info 75 - cp target/*.war src/main/docker 76 * script: 77 - docker build - t src/main/docker/Dockerfile -t \$IMAGE_TAG src/main/docker 79 - docker push \$IMAGE_TAG 80 81 <</pre>
80 81	ターゲットブランチ master
٢	
コミットメッセージ Update .gitlab-ci.yml	Commit changes

入力メッセージは任意のもので構いませんが、 今回は「パイプラインの実行のためにコミット」 と入力します。

「Commit changes」をクリックします。



コミットの確認

「Your changes have been successfully committed.」を確認 改行が入っていることを確認

> jhipster-hands-on-example > Repository			67	variables:
			68	REGISTRY_URL: \$CI_REGISTRY
Your changes have been successfully committed.			69	<pre>IMAGE_TAG: \$CI_REGISTRY_IMAGE:\$CI_COMMIT_REF_SLUG</pre>
			70	stage: release
			71	dependencies:
master y jhipster-hands-on-example / .gitlab-ci.yml	Q ファイルを	4	72	- maven-package
			73	before_script:
(学) パイプラインの実行のためにつういた		74	74	- docker info
			75	- cp target/*.war src/main/docker
			76	script:
			77	- docker login -u "gitlab-ci-token" -p "\$CI_BUILD_TOKEN" \$REGISTRY_URL
 This GitLab CI configuration is valid. Learn more 			78	- docker build -f src/main/docker/Dockerfile -t \$IMAGE_TAG src/main/docker
			79	- docker push \$IMAGE TAG
i .gitlab-ci.yml 2.15KB	ф ф		80	
1 image: jhipster/jhipster:v5.8.2			4	

画面が戻り「Your changes have been successfully committed.」メッセージが 表示されます。 画面を下にスクロールさせ、改行が入っている ことを確認してください。



パイプラインのステータス確認

□「CI/CD」から「パイプライン」をクリック □ ステータスが「実行中」であることを確認



左のナビゲーションの「CI/CD」から 「パイプライン」をクリックします。

パイプライン画面になりましたら、ステータスが 「実行中」であることを確認してください。



パイプラインの繋がりとジョブの確認

□「実行中」または「# 数値」をクリック
 □ パイプラインを確認

				② 実行中 Pipeline #58300028 tr	riggered 6 minutes ago by 🌐		Cancel ru
ステータス	パイプライン	コミット	ステー	パイプラインの実行の	のためにコミット		
				⊘ 5 job for master			
	#58300028 by 🌐	₽master-⊶bf80fe3b		P (latest)			
③ 実行中	latest	🥮 パイプラインの実行の		- ↔ bf80fe3b ઉ			
				バイプライン ジョブ §			
				Build	Test	Package	Release
				The maven-compile O	() frontend-test	maven-package	docker-push

ステータスの「実行中」かパイプラインの 「# 数値」をクリックします。

今回実行しているパイプラインの繋がりと ジョブが表示されます。





- □ ② 青いタイマーは「実行中」です
- □ ② 緑色のチェックマークは「完了」になります
- □ グレーの丸は「未実行」です

パイプライン ジョブ 5







□「実行中」のものをクリック □ 内容を確認

			Request: Validation failed for argument at index 0 in method: public void com.mycompany.myapp.web.rest.AccountResource.registerAccount(com.mycompany.myapp.web.rest.vm.HanagedUser
			VM), with 1 error(s): [Field error in object "managedUserVM" on field 'email': rejected value [invalid]; codes [Email.managedUserVM.email.email.email.jmail.java.lang.String.Email]; arguments [org.springframework.context.support.oefaultMessageSourceResolvable: codes [managedUserVM.email,email]; arguments []; default message [email],
			<pre>[Ljavax.validation.constraints.Pattern\$Flag;@61ff42c4.org.springframework.validation.beanvalidation.Spri ngValidatorndaptersResolvableAttribute@205942b0]; default message [must be a well-formed email address]] 2019-04-25 03:37:23.373 WARN 271 [</pre>
2			<pre>com.mycompany.myapp.web.rest.AccountResource.registerAccount(com.mycompany.myapp.web.rest.vm.HanagedUser VM), with 1 error(s): [field error in object 'managedUserVM' on field 'login': rejected value [funky- login]; codes [Pattern.managedUserVM.login,Pattern.login,Pattern.java.lang.String,Pattern]; arguments</pre>
	Test	Package	<pre>(vg.springrimmed k.context.support.cormatressagesourcemestivater codes [managedurervv.login,ogin], arguments []; default message [login], [Ljavax.validation.constraints.Pattern\$Flag;@1309b36.org.springframework.validation.beauvalidation.Spri</pre>
0	✓ frontend-test	maven-packa	ngVallatarAdaptersesolvableAttributegoal#130; berault message [must match "~[@A-Za-20-9-]*5]] 2019-04-25 03:372:30:05 MARM 271 [main] o.z.problem.spring.common.AdviceTrait : Bad Request: Validation failed for argument at index 0 in method: public void
			com.mycompany.mynpp.web.rest.Accountectource.registerAccount(com.mycompany.mynpp.web.rest.vest.vm.Nanageduser W0) with 1 error(s): [Field error in object "manageduserW0 in Field 'password': rejected value [123]; codes [Size.managedUserW1,password,Size.password,Size.java.lang.String,Size]; arguments Lore scinteframmurk contart sumort Defaul MessamourceMorelubale codes
	Maven-test		<pre>[cong_spring() memory account support contractive suggestion contractive codes [managed/discretW.passeord.password() arguments []; default message [password],100,4]; default message [size must be between 4 and 100]]</pre>
			2019-04-25 03:37:25.878 WARN 271 ··· [main] o.z.problem.spring.common.AdviceTrait : Bad Request: Incorrect password

いずれか「実行中」のものをクリックします。

黒い画面にテキストメッセージが表示され、 実行中であることが確認できます。







実際のYAMLファイル <u>https://gitlab.com/stylez-co-jp/jhipster-hands-on-example/blob/master/.gitlab-ci.yml</u>



.gitlab-ci.yamlの制御構造 (only/except)

□ onlyの基本

ここで指定のブランチ名、タグ名がgit pushされた時に実行する

□ exceptの基本

- ▶ ここで指定のブランチ名、タグ名がgit pushされた時には実行しない
 □ 共通ルール
 - ▶ 正規表現での指定も可能(RE2なので注意)
 - branches、tags、api等のgit名、サービス名での実行指定も可能
- □その他の指定の仕方
 - only:refs/except:refs:指定のgit refの時
 - only:variables/except:variables:指定の変数があった時
 - only:changes/except:changes:指定のファイルの変更時

サンプル

job: # use regexp only: - /^issue-.*\$/ - dev # use special keyword except: - master



ステージ間でのファイルのやり取り(artifactsとcache)

artifacts

- ステージ間のジョブとジョブでのファイルのやり取り
- >コンパイルした成果物は、artifactsを使う
- artifactsを使うと指定部分が必ずGitLabにアップロードされる
- トパイプライン終了後もGitLabから参照したい場合も、artifactsを利用する

□ cacheの使い道

- プロジェクト (パイプライン) で一時的にインターネットからダウンロードされるファイルを ジョブ間でキャッシュとして共有し高速化するために使う
- ・キャッシュの存在は保証はされていない
- ・保証されない理由は、キャッシュはRunnerホスト内でのみ共有する仕組みで別の Rancherホストで動いた場合には利用できないから
- 保証されていないため、before_scriptと組み合わせて使うのを推奨



GitLabRunnerとGitLabPipeline、GitLabレジストリの解説



Gitlab.com上でのCI/CD設定概要

gitlab-ci.ymlファイルとCI/CD画面以外の重要項目を解説します 3つの注意するべき点を説明します

🦊 GitLab 🔤 プロジェク	ト > グループ > アクティビティー マイルストーン スニペット ② ② 検索またはジャンプ先 Q	📭 🗅 🕑 🗸 🎯 🗸						
J jhipster-hands-on- example	You won't be able to pull or push project code via SSH until you SSH 鍵を追加 to your profile 次回から表示しない							
☆ プロジェクト	YumaUchida 〉 Jhipster-hands-on-example 〉 詳細							
詳細	jhipster-hands-on-example U	< 0 クローン v						
アクティビティー リリース サイクル分析	Project ID: 12220727 ● ライセンスを追加 ~ 5 コミット 『 1 プランチ ♂ 0 タグ ◎ 0パイト ファイル フォーク元 tettsu / jhipster-hands-on-example							
Uポジトリ	master > jhipster-hands-on-example / + > 履歴 Q. ファイルを検索	Web IDE 🗣 🗸						
 D 課題 0 1) マージリクエスト 0 	② gittab.comのレジストリへイメージをPushするように変更 一般 YANO Tetsuro が1ヶ月前 にコミットしました	c5541b8d G						
	メンパー インテグレーション DME ① CI/CD 設定 ④ Add CHANGELOG ④ CONTRIBUTINGを追加 ④ Kubernetes クラスターを	and all a second s						
■ レジストリ	リポジトリ 最新コミット	最終更新						
🗅 Wiki	CI / CD	2ヶ月前						
み スニペット	apps Initial commit Pages	1ヶ月前						
春 設定	Initial application generated by JHipster-5.8.2	2ヶ月前						
	webpack Initial application generated by JHipster-5.8.2	2ヶ月前						
	.editorconfig Initial application generated by JHipster-5.8.2	2ヶ月前						

GitLab Next) プロジ	ェクト~ グループ	~ アクティビティー	マイルストーン	スニペット	۵		• ~	検索またはジャン	プ先 Q	()2)	n	Ŀ	@ ~	ľ
jhipster-hands-on- example		YumaUchida → jhipst	er-hands-on-exan	nple > CI / CD) 設定									
プロジェクト		ー般のパイプラ Runner トークンに	・イン アクセスし、パイ	イプラインの	設定をカスタマ	ィイズ、そしてパイプ	ラインの状	態とカバレッジレポ	ートを閲覧	します。		展開		
リボジトリ 果題 (Auto DevOps										屈期		
マージリクエスト		Auto DevOps はあり ド、テスト、デプロ	らかじめ定義され イします。 Auto	ぃた継続的イン o DevOps の	ンテグレーショ 詳細	ンとデリバリーの設	定に基づい	て、アプリケーショ	ンを自動的	にビル		展開		
CI / CD														
里用		Runner Register and see ye	our runners for t	his project.								展開		
レジストリ		Variables 😯										展開		
x=ペット		Environment variab branches or tags. A regexp requiremen	les are applied t Additionally, they ts to do so. You	to environme / will be mas can use envi	ents via the run ked by default vironment varial	ner. They can be prosoner. They can be prosoner in they are hidden i bles for passwords,	otected by in job logs, secret key	only exposing them , though they must r s, or whatever you v	i to protec natch cert vant. You r	ted ain nay also				
没定		add variables that a	are made availat	ble to the run	nning applicatio	on by prepending the	e variable l	key with K8S_SECRE	T詳しい	情報				
一般		バイプラインの)トリガー									展開		
メンバー		トリカーは、特定の へのアクセスやプロ)フランチまたは ジェクトのアク [.]	タクを API 明 セス権など、	ሦび出しで強制的 関連するユーサ	的に冉構築させること ガーを偽装します。	とかできま	す。これらのトークン	ノは、 プロ:	シェクト				
インテグレーション														
リポジトリ														

左サイドバー → 設定 → Cl/CDをクリック

CI/CDの動作を設定する画面





□一般のパイプライン

「一般のパイプライン」はパイプラインに関する一般的な設定を行います

Auto DevOPS

GitLab側で必要と考えているDevOpsの作業を自動的に実行してくれる機能

Runner

パイプラインの実行に必要なRunnerプロセスに関する設定を行う場所 このRunnerは、.gitlab-ci.yamlで書かれたジョブを実行するプロセスになります

Variables

Runnerでジョブを実行するときに利用する環境変数を設定する場所

パイプラインのトリガー

パイプライン実行の契機は、基本的にGitLab.comへの操作ですが、外部からAPIで呼び出 すことによりパイプラインを実行することもできる設定

🤞 GitLab 🛯 🕫	プロジェクト 〜	グループ v アクティビティー マイルストーン スニペット 🕢
J jhipster-hands- example	•on-	YumaUchida > jhipster-hands-on-example > CI / CD 設定
▶ プロジェクト		一般のパイプライン
) リポジトリ		Runner トークンにアクセスし、パイプラインの設定をカスタマイズ、
) 課題	0	Auto DevOps
] マージリクエスト	0	Auto DevOps はあらかじめ定義された継続的インテグレーションとデ ド、テスト、デプロイします。 Auto DevOps の詳細
CI/CD		Duran
運用		Register and see your runners for this project.
〕 レジストリ		Veriebles O
) Wiki		Environment variables are applied to environments via the runner. Th
。 スニペット		branches or tags. Additionally, they will be masked by default so the regexp requirements to do so. You can use environment variables fo add variables that are made available to the running application bur
1 設定		
一般		ハイノフィンのトリカー
メンバー		へのアクセスやプロジェクトのアクセス権など、関連するユーザーを9
インテグレーション	,	





Git strategy for pipelines

ソースコードを取得する際のコマンドを選択できる設定 ・git clone: 毎回全てのソースコードをダウンロード ・git fetch: 既存コードとの差分だけを取得

タイムアウト

ジョブ全体のタイムアウト時間。設定値以上の時間がかかる場合はfailとなる

□ カスタムCl configパス

リポジトリ内にある.gitlab-ci.yamlの場所を指定します デフォルト以外にgitlab-ci.ymlが存在する場合にパスを記入 ファイル名も変更可能

Runner トークンにアクセン	スし、パイプラインの設定をカスタマイズ、そしてパイプラインの状態とカバレッジレポートを閲覧します
	-
Git strategy for pipelines	
clone または fetch をx	&死して最新のアプリケーションコードを取得してください Q
git clone	
Slower but makes sure git fetch	the project workspace is pristine as it clones the repository from scratch for every job
Faster as it re-uses the	project workspace (falling back to clone if it doesn't exist)
ያተムアウト	
1h	E
f any job surpasses this t	meout threshold, it will be marked as failed. Human readable time input language is accepted like "1
nour". values without spe	cification represent seconds. 🔮
カスタム CI config パス	
カスタム CI config パス .gitlab-ci.yml	
カスタム Cl config パス gitlab-ci.yml コ 設定ファイルへのパス。	デフォルトは .gitlab-ci.yml 0
カスタム Cl config パス .gitlab-ci.yml J 設定ファイルへのパス。	デフォルトは .gitlab-ci.yml 0
カスタム Cl config パス .gitlab-ci.yml Cl 設定ファイルへのパス。	デフォルトは .gitlab-ci.yml 0
カスタム Cl config パス gitlab-ci.yml コ 設定ファイルへのパス。 2 公開バイブライン	デフォルトは .gitlab-ci.yml 0

一般のパイプライン - 2

□ 公開パイプライン

ジョブで生成したファイルなどをダウンロードできるようにするか 否かを設定できる機能(対象ファイルはコンパイル後のファイルなど)

Auto-cancel redundant, pending pipelines

HEADブランチ以外でパイプラインが重複した場合に古いジョブを 自動的に停止。複数ジョブが起動した場合に適用される

□ テストカバレッジ解析

カバレッジログを取得してテストのカバレッジの割合を取得する機能

公開バイプライン	プの詳細な情報に対して、パブリックレベルのアクセスを許可し	ます。
If enabled:		
For public	projects, anyone can view pipelines and access job details (output logs and artifacts)	
 For interna For private artifacts) 	al projects, any logged in user can view pipelines and access job details (output logs and artifac a projects, any member (guest or higher) can view pipelines and access job details (output logs	ts) and
If disabled, the ad	access level will depend on the user's permissions in the project.	
Auto-cancel redun	ndant, pending pipelines	
Auto-cancel redun	ndant, pending pipelines the same branch 2	
Auto-cancel redun	ndant, pending pipelines the same branch 🕢	
Auto-cancel redun New signification will o テストカバレッジ解析	ndant, pending pipelines the same branch 🕢	
Auto-cancel redun New sinciliace will o テストカバレッジ解析 / Regular expressio	indant, pending pipelines the same branch I index of the same branch I inde	
Auto-cancel redun Auto-cancel redun Fストカバレッジ解析 / Regular expressio ジョブトレースでテスト	ndant, pending pipelines the same branch on トカパレッジの結果を見つけるために使用される正規表現です。無効にする場合は空白のままにしま	ます。 የ
Auto-cancel redun Now significations will of テストカバレッジ解析 / Regular expressi ジョブトレースでテスト 以下のものは、既存	ndant, pending pipelines the same branch ion トカパレッジの結果を見つけるために使用される正規表現です。無効にする場合は空白のままにしま の正規表現ツールの例です:	ます。 😧
Auto-cancel redun New size line will of Fストカバレッジ解析 / Regular expressio ジョブトレースでテスト 以下のものは、既存 • Simplecov (R	ndant, pending pipelines the same branch ion トカパレッジの結果を見つけるために使用される正規表現です。無効にする場合は空白のままにしま の正規表現ツールの例です: Ruby) - \(\d+.\d+\%)) covered	ます。 😧
Auto-cancel redun New sizelises will of テストカバレッジ解析 / Regular expressi ジョブトレースでテスト 以下のものは、既存 • Simplecov (R • pytest-cov (P	ndant, pending pipelines the same branch @ ion トカバレッジの結果を見つけるために使用される正規表現です。無効にする場合は空白のままにしま の正規表現ツールの例です: Ruby) - \(\d+.\d+\%\) covered Python) - ^TOTAL\s+\d+\s+\(\d+\%)\$	ます。 🕤
Auto-cancel redun New significations will a テストカバレッジ解析 / Regular expressio ジョブトレースでテスト 以下のものは、既存 ・ Simplecov (Rt ・ pytest-cov (P ・ phpunitcov	ndant, pending pipelines the same branch ion トカパレッジの結果を見つけるために使用される正規表現です。無効にする場合は空白のままにしま の正規表現ツールの例です: kuby) - \(\d+.\d+\%)) covered Python) - ^T0TAL\s+\d+\s+\(\d+\%)\$ verage-textcolors=never (PHP) - ^\s+Lines:\s*\d+.\d+\%	ます。 <table-cell></table-cell>
Auto-cancel redun Nameierine will e テストカバレッジ解析 / Regular expressi ジョブトレースでテスト 以下のものは、既存 ・ Simplecov (Rt ・ pytest-cov (P ・ phpunitcov ・ gcovr (C/C++	ndant, pending pipelines the same branch ion トカバレッジの結果を見つけるために使用される正規表現です。無効にする場合は空白のままにしま の正規表現ツールの例です: Ruby) - \(\d+.\d+\%) covered Python) - ^TOTAL\s+\d+\s+\d+\s+(\d+\%) \$ verage-textcolors=never (PHP) - ^\s*Lines:\s*\d+.\d+\% +) - ^TOTAL.*\s+(\d+\%) \$	ます。 <table-cell></table-cell>
Auto-cancel redun Auto-cancel redun Auto-cancel redun Facharity () Regular expressiv ジョブトレースでテスト 以下のものは、既存 Simplecov (R ・ pytest-cov (P ・ phpunitcov ・ gcovr (C/C++ ・ tapcoverag	ndant, pending pipelines the same branch ion トカパレッジの結果を見つけるために使用される正規表現です。無効にする場合は空白のままにしま #の正規表現ツールの例です: huby) - \(\d+.\d+\%\) covered >ython) - ^TOTAL\s+\d+\s+\(d+\%)\$ verage-textcolors=never (PHP) - ^\s+Lines:\s*\d+.\d+\% +) - ^TOTAL.*\s+(\d+\%)\$ ge-report=text-summary (NodeJS) - ^Statements\s*:\s*([^%]+)	ます。 f
Auto-cancel redun Auto-cancel redun Auto-cancel redun Fストカバレッジ解析 / Regular expressio ジョブトレースでテスト 以下のものは、既存 Simplecov (R pytest-cov (P phpunitcov gcovr (C/C++ tapcoverag nyc npm test	ndant, pending pipelines the same branch ion トカパレッジの結果を見つけるために使用される正規表現です。無効にする場合は空白のままにしま #の正規表現ツールの例です: Ruby) - \(\d+.\d+\%\) covered Python) - ^TOTAL\s+\d+\s+\d+\s+(\d+\%)\$ werage-textcolors=never (PHP) - ^\s+Lines:\s+\d+.\d+\% +) - ^TOTAL.*\s+(\d+\%)\$ ge-report=text-summary (NodeJS) - ^Statements\s+:\s+([^%]+) : (NodeJS) - All files[^]]*\[[^]*\s+([\d.]+)	ます。 f
Auto-cancel redun Num visuling will of Fストカバレッジ解析 / Regular expressio ジョブトレースでテスト 以下のものは、既存 Simplecov (R pytest-cov (P phpunitcov gcovr (C/C++ tapcoveralls (E excoveralls (E	ndant, pending pipelines the same branch the same branch t	ます。 <table-cell></table-cell>
Auto-cancel redun New size line o will of Fストカバレッジ解析 / Regular expressio ジョブトレースでテスト 以下のものは、既存 ・ Simplecov (R ・ pytest-cov (P ・ phpunitcov ・ g covr (C/C++ ・ tapcoverag ・ nyc npm test ・ excoveralls (E ・ mix testcov	ndant, pending pipelines the same branch the same branch t	ます。 🕑





Pipline status

パイプラインの状況を表示する機能 成功すれば「passed」、どこかで失敗していたら「fail」で表示

Coverage report

カバレッジの状態を表示する機能 カバレッジ解析の結果を画像として表示する

Pipeline status			
Pipeline status · pi	peline unknown	master	~
Markdown	<pre>[![pipeline status](https://gitlab.com/r aster/pipeline.svg)](https://gitlab.com/ /master)</pre>	outeplus/jhipster-hands-on-example/badges routeplus/jhipster-hands-on-example/commi	/m .ts
HTML	<a href="https://gitlab.com/routeplus/jh
g alt=" pipeline="" src="https://git
/badges/master/pipeline.svg" status"="">	ipster-hands-on-example/commits/master"≻< lab.com/routeplus/jhipster-hands-on-examp	im le
AsciiDoc	<pre>image:https://gitlab.com/routeplus/jhips .svg[link="https://gitlab.com/routeplus/ itle="pipeline status"]</pre>	ter-hands-on-example/badges/master/pipeli jhipster-hands-on-example/commits/master"	.ne ,t
Coverage report			
Coverage report ·	overage unknown	master	~
Markdown	<pre>[![coverage report](https://gitlab.com/r aster/coverage.svg)](https://gitlab.com/ /master)</pre>	outeplus/jhipster-hands-on-example/badges routeplus/jhipster-hands-on-example/commi	/m .ts
HTML	<a coverage="" href="https://gitlab.com/routeplus/jh
g alt=" report"="" src="https://git
/badges/master/coverage.svg">	ipster-hands-on-example/commits/master">< lab.com/routeplus/jhipster-hands-on-examp	im le
AsciiDoc	<pre>image:https://gitlab.com/routeplus/jhips .svg[link="https://gitlab.com/routeplus/ itle="coverage report"]</pre>	ter—hands—on—example/badges/master/covera jhipster—hands—on—example/commits/master"	ge ,t



Auto DevOps

Auto DevOps

Auto DevOpsはGitLabが規定するDevOpsのやり方に沿り自動的に パイプラインを実行してくれる機能

Auto DevOpsにより、ビルドからコードのテスト、チェック、アプリのレビューまで自動的に進行する事が可能

加スマノノ・コーノ ノーコーノ

Runner トークンにアクセスし、パイプラインの設定をカスタマイズ、そしてパイプラインの状態とカバレッジレポートを閲覧します。

Auto DevOps

Auto DevOps はあらかじめ定義された継続的インテグレーションとデリバリーの設定に基づいて、アプリケーションを自動的にビル ド、テスト、デプロイします。 Auto DevOps の詳細

□ デフォルトの Auto DevOps パイプライン

その他に CI 設定ファイルが見つからない場合、Auto DevOps パイプラインが実行されます。 詳しい情報

変更を保存

Runner



Runner

□ Runnerの概要

オンプレで動かす場合は、別途自分でRunnerを準備する必要がありますが、 GitLab.comでは他のユーザーと共有する共有Runnerが自動的に割りあてらる ため、特別な設定が必要なくすぐに利用可能

□ 利用可能な共有 Runner

オレンジ枠に割り当てられた共有Runnerが表示される

オンプレミス上のGitlabの場合は管理者からの設定が必要なため、共有Runner の利用が必要な場合は事前に問い合わせる

* gitlab.com上であっても自前のRunnerを設定することが可能

Runner
Register and see your runners for this project.
「Runner」はジョブを実行するプロセスです。必要な数の Runner は別々のユーザー、サーバー、さらにはあなたの日
各 Runner は次のいずれかの状態をとります:
● <mark>active</mark> - Runnerがアクティブで新しいジョブを処理 ● paused - Runnerが停止中のため新しいジョブは処理

To start serving your jobs you can either add specific Runners to your project or use shared Runners

Specific Runners

共有 Runner

Set up a specific Runner automatically

Kubernetes クラスターに Runner を簡単にインストールで きます。Kubernetes の詳細

- 下のボタンをクリックすると、Kubernetesのページ に遷移し、インストールプロセスを開始します
- 既存の Kubernetes クラスターを選択するか、新しい ものを作成
 Kubernetes クラスターの詳細画面を介して、アプリ ケーションリストから Runner をインストールしま
- す。

Kubernetes に Runner をインストール

Set up a specific Runner manually

```
    GitLab Runner のインストール
    Runner セットアップの際に次の URL を指定してください: https://gitLab.com/ G
    セットアップの際に次の登録トークンを使用してください: QK9qPqz8xBKpaGqY7Xbb G
```

Runner 登録トークンをリセット

4. Runner を起動!

 だ
 Shared Runners on GitLab.com run in autoscale mode

 and are powered by Google Cloud Platform. Autoscaling

 means reduced wait times to spin up builds, and isolated

 VMs for each project, thus maximizing security.

 They're free to use for public open source projects and

 limited to 2000 Cl minutes per month per group for

 private projects. Read about all GitLab.com plans.

 U

 共有 Runner を無効化

 for this project

 利用可能な共有 Runner 8

 ● ed2dce3a

 shared-runners-manager-6.gitlab.com
 #380987

 だ

 ● 0277ea0f

shared-runners-manager-5.gitlab.com

docker gce

Style₂3

#380986

Variables

□ Variablesの内容

Runnerでジョブを実行するときに利用する環境変数を設定する ジョブ実行時に値を表示しない「Masked」がデフォルト(オレンジ枠の部分をクリックで切り替え可能) 保護をonにすると、保護ブランチ、タグ上でしか利用されない環境変数になる

Variables 😯

Environment variables are applied to environments via the runner. They can be protected by only exposing them to protected branches or tags. Additionally, they will be masked by default so they are hidden in job logs, though they must match certain regexp requirements to do so. You can use environment variables for passwords, secret keys, or whatever you want. You may also add variables that are made available to the running application by prepending the variable key with K85_SECRET_. 詳しい情報





GitLab CI/CD実行時の環境変数

□ 自動的にCl_xxxの環境変数がセットされる

\$ env

jile b to a

CI BUILD ID=12866 CI SERVER VERSION PATCH=7 CI SERVER REVISION=93e3215 CI_COMMIT_SHORT_SHA=66db292b GITLAB USER LOGIN=t-yano CI BUILD REF=66db292be13a6738dfa472aeaf25aac0bcb 25bc5 CI=true CI PROJECT NAME=test-gitlab-repo CI RUNNER REVISION=8bb608ff HOSTNAME=runner-1e3e3c11-project-410-concurrent-0 CI JOB STAGE=check CI COMMIT DESCRIPTION= CI SERVER VERSION=11.9.7 SHLVL=2 HOME=/root OLDPWD=/ CI JOB ID=12866 CI COMMIT REF NAME=master CI PIPELINE SOURCE=push CI RUNNER VERSION=11.7.0 CI SERVER VERSION MAJOR=11 CI BUILD TOKEN=xxxxxxxxxxxxxxxxxxxxxxx GITLAB FEATURES= CI PROJECT ID=410 CL REGISTRY-MAGE=registry.stylez.co.jp/t-yano/test-

yano/test-gitlab-repo.git GITLAB CI=true CI COMMIT SHA=66db292be13a6738dfa472aeaf25aac0 CI RUNNER ID=6 CI REGISTRY=registry.stylez.co.jp bcb25bc5 CI API V4 URL=https://gitlab.stylez.co.jp/api/v4 CI REGISTRY USER=gitlab-ci-token CI PROJECT DIR=/builds/t-yano/test-gitlab-repo GITLAB USER NAME=YANO Tetsurou CI PROJECT PATH=t-yano/test-gitlab-repo CI PIPELINE IID=12 CI PROJECT NAMESPACE=t-yano CI JOB URL=https://gitlab.stylez.co.jp/t-yano/test-gitlab-CI JOB TOKEN=xxxxxxxxxxxxxxxxxxxxxx repo/-/jobs/12866 CI SERVER NAME=GitLab CI BUILD NAME=check CI RUNNER EXECUTABLE ARCH=linux/amd64 CI NODE TOTAL=1 CI PIPELINE URL=https://gitlab.stylez.co.jp/t-yano/test-CI_COMMIT_REF_SLUG=master gitlab-repo/pipelines/4829 CI DISPOSABLE ENVIRONMENT=true PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/s PWD=/builds/t-yano/test-gitlab-repo CI RUNNER TAGS= bin:/bin CI SERVER VERSION MINOR=9 CI SERVER TLS CA FILE=/builds/t-yano/test-gitlab-CI RUNNER DESCRIPTION=gitlab-runner-03 repo.tmp/CI SERVER TLS CA FILE CI BUILD REF SLUG=master CI PIPELINE ID=4829 CI PROJECT VISIBILITY=private CI COMMIT BEFORE SHA=67295bd87b1ce237dd0f698b CI COMMIT TITLE=beforeとafter scriptを削除 434816f36a16f4ce GITLAB USER EMAIL=tetsuro.yano@stylez.co.jp CI BUILD REF NAME=master CI PROJECT PATH SLUG=t-yano-test-gitlab-repo CI SERVER=ves CI PROJECT URL=https://gitlab.stylez.co.jp/t-yano/test-CI BUILD BEFORE SHA=67295bd87b1ce237dd0f698b43 4816f36a16f4ce gitlab-repo CI CONFIG PATH=.gitlab-ci.yml CI PAGES URL=https://t-yano.pages.stylez.co.jp/testgitlab-repo FF_K8S_USE_ENTRYPOINT_OVER_COMMAND=true FF_K8S_USE_FF_K8S_USE_FF_K8S https://giitaconny/work/items/4c5085b4cd6221bb71c5 CI PAGES DOMAIN=pages.stylez.co.jp CI REPOSITORY URL=https://gitlab-ci-CI JOB NAME=check token:xxxxxxxxxxxx@gitlab.stylez.co.jp/t-

GitLab CI/CDの画面解説

□ ダッシュボード

メインカラムにジョブの一覧が表示される(ダッシュボード) 各ジョブの実行状態が表示されている

ジョブ	詳細
パイプライ	・中のジョブを1つずつ詳細に表示している

jhipster-hands-on-	スタイルズ太郎 > j	hipster-hands-on-example >	パイプライン			
example プロジェクト	すべて 10 🖗	保留中 0 稼働中 0 完	了 10 プランチ タグ			
リポジトリ	ステータス	パイプライン	コミット	ステージ		
課題 0 マージリクエスト 0	④ 成功	#60495409 by ## latest	P master 		ð 00:19:16 ៣ 12 hours ago	ç
CI/CD	⊘ 成功	#59765023 by 🌞	Pmaster - ◆ 1534c44c		⊘ 00:18:29 ∰ 4 days ago	
7/1/71/2	⊙ 成功	#59258318 by ∯	Pmaster - ◆ 4af03220		⊚ 00:17:11	
チャート	◎ キャンセル	#59257241 by 🌺	Pmaster -> db667c7f ∯ 削除	0000	⊚ 00:09:34 首 1 week ago	
レジストリ Wiki	◎ キャンセル	#59256450 by 🇱	ド revert-05ab → 02787ada ∯ Revert *アップデート src/ma	0000	⊘ 00:01:54 首 1 week ago	
スニペット	③キャンセル	#59256430 by 🏰	P master -> ef81f65f ● Revert "ロゴを変更"	0000	⊙ 00:02:14	

🦊 GitLab 🗪 プロジェクト	ヽッ グループ ∨ アクテ	ィビティー マイルストーン スニペット 🅢		0		/プ先 ♀ 012		0 ~	۵.
jhipster-hands-on- example	スタイルズ太郎 > jhipste	er-hands-on-example > Jobs							
	All 50 Pending	0 Running 0 Finished 50							
リポジトリ	Status	Job	Pipeline	Stage	Name		Coverage		
0 課題 0	②成功	#209692008 ¥ master -> d7ac2c11	#60495409 by ∰	release	docker-push	♂ 00:01:27 曲 12 時間前			T
îî マージリクエスト 0	◎ 成功	#209692007 ¥ master -> d7ac2c11	#60495409 by 🌐	package	maven-package	◎ 00:07:01 曲 12 時間前			q
🤗 CI/CD	◎ 成功	#209692006 ¥ master ∞ d7ac2c11	#60495409 by 🌞	test	frontend-test	⊘ 00:04:43 曲 12 時間前			q
ジョブ	◎成功	#209692005 ¥ master -∞ d7ac2c11	#60495409 by 🌞	test	maven-test	◎ 00:06:12 曲 12 時間前			q
チャート	◎成功	#209692004 ¥ master -∞ d7ac2c11	#60495409 by 🌞	build	maven-compile	◎ 00:04:34 曲 12 時間前			q
ロレジストリ	◎成功	#207195937 ¥ master ↔ f534c44c	#59765023 by 🇱	release	docker-push	◎ 00:01:20 曲 4日前			
○ Wiki み スニペット	◎成功	#207195936 ¥ master -> f534c44c	#59765023 by 🌐	package	maven-package	◎ 00:06:26 曲 4日前			
載 メンバー	◎ 成功	#207195935 ¥ master -	#59765023 by 🌐	test	frontend-test	⊘ 00:04:13 曲 4日前			
						a 00:06:24			



スケジュール画面

スケジュールとは

イベントやトリガー契機でパイプラインを実行するのではなく、時間で パイプラインを実行する。

実行の時間指定方法は、cron構文で行い、タイムゾーンとブランチの 指定、環境変数も設定が可能

□ スケジュール設定

- 新規スケジュールをクリック → ・パイプラインの説明
- ・タイムゾーン
- ・実行パターン(CRON形式での指定)
- ・ターゲットブランチ

・変数

などを指定し、スケジュール実行が可能



🦊 GitLab 🗪 プロジェク	ト v グループ v アクティビティー マイルストーン スニペット 🕢	• •	検索またはジャンプ先 Q	0)2	n	ß	0 ~
J jhipster-hands-on- example	YumaUchida > jhipster-hands-on-example > Pipelines > Schedules						
合 プロジェクト	新しいパイプラインのスケジュールを作成						
B リポジトリ	說明						
0 課題 0	このパイプラインについて簡単に記述してください。		E3				
1 マージリクエスト 0	間隔のパターン						
🦸 CI/CD	○ カスタム (Cron の構文) ● 毎日 (午前4:00) ○ 毎週 (日曜日の午前4:00) ○ 毎月 (1日の午前4:00)						
バイプライン	04***						
ジョブ	Cron のタイムゾーン						
スケジュール	итс		~				
チャート	ターゲットプランチ						
G 運用	master		~				
	変数						
皇 レジストリ	キーを入力 値を入力		hi.				
🖞 Wiki	アクティブ						
み スニペット	☑ 有効						
春 設定	パイプラインスケジュールを保存						キャンセ

レジストリ

Container Registry とは

CI/CDで使用するDocker イメージを格納できる機能 現在保存されているコンテナイメージついての情報を 確認できる

クリックで展開

リンクになっているためアイコンをクリックするとimage のURLをコピー可

コンテナイメージ取得方法の解説



GitLab Container Registry



Docker Container Registry

Docker Container Registryとは?

- Dockerイメージを保存する場所 (docker pushして送信する先)
- Dockerイメージをプライベートに管理する(有料ではDocker Hubでも可能)
- Dockerイメージを配布管理する





GitLab Container Registry

GitLab Container Registryとは?

- Docker Container Registryと同じもので、GitLabに統合されている
- dockerコマンドで利用できる(docker loginが必要)
- ▶ GitLabのアカウントで認証する
- ▶ GitLabパイプラインの中から使うと認証は楽に統合できる
- プロジェクト単位でPublic、Private設定が可能
- ・その他の機能はDocker Container Registryと同じ
- GitLab Container Registryにpushする場合は、docker imageにContainerレジストリのタグをつけること



レジストリ

Container Registry とは

CI/CDで使用するDocker イメージを格納できる機能 現在保存されているコンテナイメージついての情報を 確認できる

クリックで展開

リンクになっているためアイコンをクリックするとimage のURLをコピー可

コンテナイメージ取得方法の解説



GitLab CI/CDパイプライン設定リファレンスについて

より詳細なGitLab CI/CDパイプラインの.gitlab-ci.yamlの書き方については、 以下を参照してください。 https://gitlab.com/stylez-co-jp/gitlab-ce/tree/dev-v12.0.3-

ja.1/doc-ja/ci/yaml

イビティー マイルストーン スニペット 🙆	8 ~	検索またはジャンプ先	٩	0)4	1 11	
type: reference						
GitLab CI/CD <mark>パイプライン設定リファレンス</mark>						
GitLab CI/CD pipelinesは、プロジェクト毎に <mark>.gitlab-ci.yml</mark> というYAMLファイルを使って設定します。						
.gitlab-ci.ymlファイルにパイプラインの構造と実行順序を定義して以下の内容	を実行しる	ます:				
 GitLab Runnerで何を動かすか。 特定の条件になったときにどうするかを指定します。例えば、プロセスが成功 	または失則	女した場合です。				
こちらの章では、CI/CDパイプラインの設定について解説しています。その他のCI/C	Dの設定(については以下を参照して	ください	し ヽ :		
 GitLab CI/CD 変数、パイプラインが実行される環境の変数について。 GitLab Runner 詳細設定, GitLab Runnerの設定について。 						



- まとめ
- GitLab CI/CDパイプラインは、Jenkinsのような自動ビルドをするための機能 がある
- 設定方法は、.gitlab-ci.ymlファイルをGitリポジトリに設置することで自動的に 実行される
- □.gitlab-ci.ymlには構文があり、複数のステージとジョブを定義する
- ジョブは、GitLab Runnerというコンテナー内のシェルで実行される
- □ビルド成果物は、artifactsとして保存され、次のジョブに利用することもできる □環境変数を指定することによりジョブの挙動を変更することができる

