

تحقیق پایانترم برنامه سازی وب

## معرفی CI/CD و کاربردهای آن



دانشگاه صنعتی شریف

تهیه کنندگان:

سارا زاهدی موحد

سینا الهی منش

داریوش امیری

سانیا صفری

## فهرست مطالب

۳	معرفی CI/CD .....
۹	وظایف یک متخصص CI/CD .....
۱۰	مهارت های یک متخصص CI/CD .....
۱۱	معرفی ابزارها .....
۱۹	سرویس های آماده CI/CD در داخل و خارج ایران .....
۲۹	منابع .....

## (A) معرفی CI/CD

همانطور که می دانیم، در توسعه اپلیکیشن های مدرن، هدف این است که چندین توسعه دهنده به طور همزمان روی ویژگی های مختلف یک برنامه کار کنند. با این حال، اگر سازمانی برای ادغام همه Branch های Source Code با یکدیگر در یک روز راه اندازی شود، کار حاصل می تواند خسته کننده، دستی و زمان بر باشد چرا که ممکن است وقتی توسعه دهنده ای که به صورت مجزا کار می کند، تغییری در یک برنامه ایجاد کند، با تغییرات مختلفی که به طور همزمان توسط توسعه دهندگان دیگر ایجاد می شوند، تضاد داشته باشد. اگر هر توسعه دهنده محیط توسعه یکپارچه محلی (IDE) خود را سفارشی کرده باشد، این مشکل می تواند بیشتر شود، به جای اینکه تیم بر روی یک IDE که Cloud-Based است، توافق کنند.

به طور کلی مشکلات روش سنتی را میتوان به ۳ دسته تقسیم کرد:

۱. تحویل آهسته: کارهای دستی برای کسانی که آنها را کامل میکنند خسته کننده بوده و این وظایف روند تحویل را کند می کند و در نهایت نوآوری را متوقف می کند. و طبیعی است که اگر رقابت از فرایند خودکار سازی استفاده می کند و شما از آن استفاده نمی کنید، رقابت برنده است.

۲. عدم دید: سوالاتی مثل؛ خطا کجا رخ می دهد؟، چه چیزی باعث شد؟، چه چیزی در هر محیط مستقر شده است؟، آیا می توانیم به عقب برگردیم؟ و غیره که طبیعتاً پاسخ به آن ها به سادگی و سرعت نخواهد بود.

۳. خطاها و ناراحتی کاربران: مکرر بودن خطاها، احتمال اشتباه در روش های سنتی و دستی و غیره که طبیعتاً نارضایتی کاربران را به همراه خواهد داشت. از طرف دیگر خطاها باعث ایجاد تنش بین افراد و بخش های درگیر در فرآیند تحویل نرم افزار می شود. عملیات، توسعه دهندگان را برای کد بد مقصر می دانند. توسعه دهندگان از تمام وظایف دستی ناامید هستند و تیم های تضمین کیفیت یا همان QA را به خاطر عدم تشخیص خطاها سرزنش می کنند. خدمات مشتری نیز هنگامی که باید پاسخگوی یک

کاربر ناراضی باشند، همه کسانی را که در این فرآیند دخیل هستند سرزنش می کند. در نهایت، مجموعه فاقد همکاری و رفاقت خواهد بود.

علاوه بر این، از آنجایی که مقادیر زیادی از کد، گاهی اوقات ادغام میشوند، خطاها در پایان یک چرخه توسعه طولانی پیدا می شوند و ممکن است رفع آنها چالش برانگیز تر باشد یا پیامدهایی برای سایر بخش های پایگاه کد داشته باشد که عیب یابی آنها دشوار و سخت است.

اما میتوان فرایندهای مذکور را خودکار نمود و تا حد زیادی مشکلات را کاهش داد یا به عبارتی از روش CI/CD استفاده نمود. در ادامه به برخی از مزایای آن اشاره خواهیم کرد:

۱. **تحويل سریعتر:** با اتوماسیون، می‌توانید به تحويل سریع‌تر و بازخورد کاربران نهایی دست یابید زیرا کارهای دستی کمتری برای تکمیل وجود دارد و تغییرات کوچک‌تری را به دفعات بیشتری در تولید ایجاد می‌کنید .

۲. **افزایش دید:** با CI/CD ، فرآیند آزمایش و استقرار شفاف است و به همین دلیل، هر مشکلی تقریباً بلافاصله قابل مشاهده است و منبع را می توان به سرعت پیدا کرد، بنابراین حدس و گمان معمولاً در شناسایی علت کاهش می یابد و از آنجایی که اشکالات به راحتی قابل ردیابی هستند، افراد پاسخگو هستند. که البته به معنی سرزنش افراد و پیدا کردن مقصر نیست بلکه، به این معنی است که شخصی که روی آن بخش خاص کار کرده است، بهترین کسی است که آن را میتواند حل کند و سرعت و کیفیت عیب یابی و انجام کار بالاتر خواهد رفت.

۳. **خطاهای حذف شده:** ویژگی های نرم افزار مدرن، پروژه ها و برنامه های کاربردی پیچیدگی است. اما یکپارچه سازی مداوم (CI) برخی از این پیچیدگی ها را از بین می برد، زمینه هایی را که در آن مشکلات وجود دارد را کاهش داده و احتمال موفقیت را افزایش می دهد.

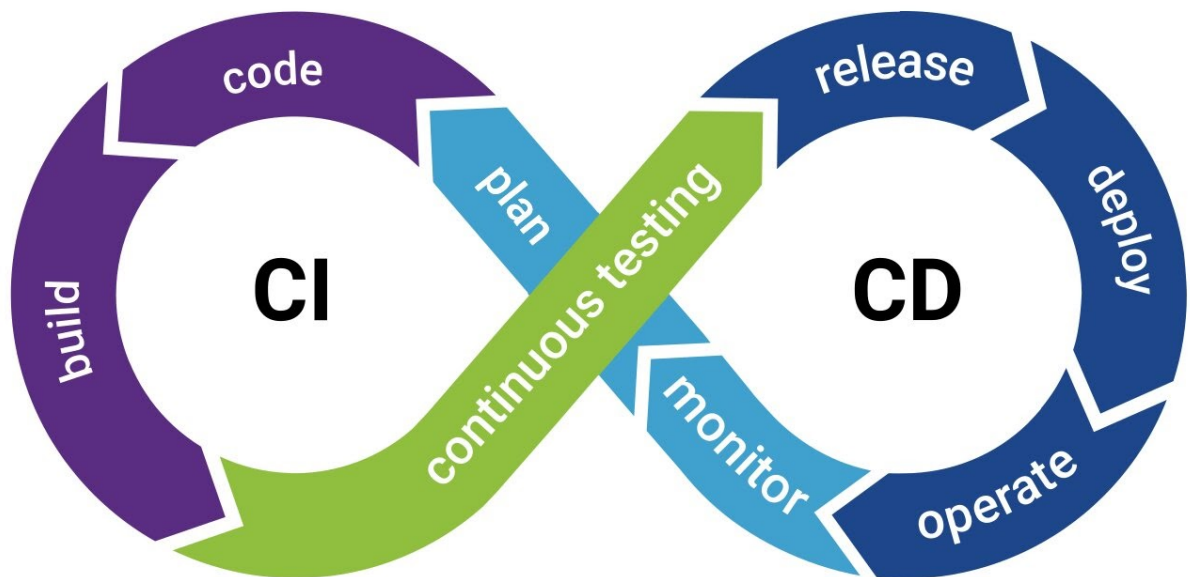
دو اصل مهم یکپارچه سازی مداوم، اجرای مکرر کد و خودکارسازی بخش های ساخت و آزمایش چرخه عمر تحويل نرم افزار است. اجرای مکرر یکپارچه سازی کد به این معنی است که خطاها زودتر پیدا شده و میتوان آنها را زودتر اصلاح کرد زیرا حجم کار کمتری برای انجام دادن وجود دارد.

از طرف دیگر، توسعه دهندگان وظایف دستی کمتری دارند، به این معنی که فرصت کمتری برای خطای انسانی وجود دارد. عملیات کد با کیفیت بالا دریافت می کند QA. مسائل کمتری برای حل کردن دارد. خدمات مشتری بازخورد منفی کمتری از مشتریان دارد و به تبع موفقیت و همکاری داخل مجموعه نیز بیشتر و بهتر خواهد بود.

۴. منابع آزاد شده: اگر کارهای تکراری به اتوماسیون واگذار شود، زمان برای توسعه دهندگان آزاد شده تا بتوانند کاری را که دوست دارند در سطح بالاتری انجام دهند.

۵. کاربران نهایی رضایتمندتر: انتشار سریع، مکرر و خطاهای کمتر منجر به اعتماد بین توسعه دهندگان و بقیه کسب و کار، رعایت ددلاین های زمانی، نتایج قابل اعتماد و رضایت بیشتر کاربران نهایی می شود.

در ادامه شمایی کلی از CI/CD را مشاهده خواهیم کرد:



CI/CD مخفف Continuous Integration و Continuous Deployment/ Continuous Delivery می باشد، که روشی است برای ارائه مکرر برنامه ها به مشتریان با استفاده از خودکار سازی در مراحل توسعه برنامه که به یک تیم کمک میکند تا مراحل ادغام، آزمایش، تحویل و استقرار برنامه ها را خودکار کنند و نظارت مستمری وجود داشته باشد. در ادامه به توضیح هر یک از ۳ مفهوم بالا می پردازیم؛



1. Continuous Integration یا همان یکپارچه سازی مداوم که یک روش توسعه نرم افزار مدرن است که در آن تغییرات تدریجی کد به طور مکرر و قابل اطمینان، انجام میشود.

در واقع CI به توسعه دهندگان کمک می کند تغییرات کد خود را در یک Branch مشترک به دفعات مختلف ادغام کنند، گاهی اوقات روزانه یا حتی ساعتی. هنگامی که تغییرات یک برنامه نویس در یک برنامه ادغام می شوند، این تغییرات با ساخت خودکار برنامه و اجرای سطوح مختلف آزمایش خودکار، معمولاً تست های واحد و یکپارچه سازی، تأیید می شوند تا اطمینان حاصل شود که تغییرات برنامه را خراب نکرده است.

این به معنای آزمایش همه چیز از کلاس ها و عملکردها گرفته تا ماژول های مختلف است که کل برنامه را تشکیل می دهند. اگر آزمایش خودکار یک تضاد بین کد جدید و موجود را کشف کند، CI رفع سریع و مکرر آن اشکالات را آسان تر می کند. به عبارت دیگر می توان گفت، هدف CI ایجاد یک روش ثابت و خودکار برای ساخت، بسته بندی و آزمایش برنامه ها است که منجر به کیفیت بهتر نرم افزار می شود.

CI Workflows بسته به ابزار، زبان برنامه نویسی، پروژه و بسیاری از عوامل دیگر بسیار متفاوت است، اما به طور کلی یک flow مراحل زیر را دارد:

۱. Push کردن به repository.

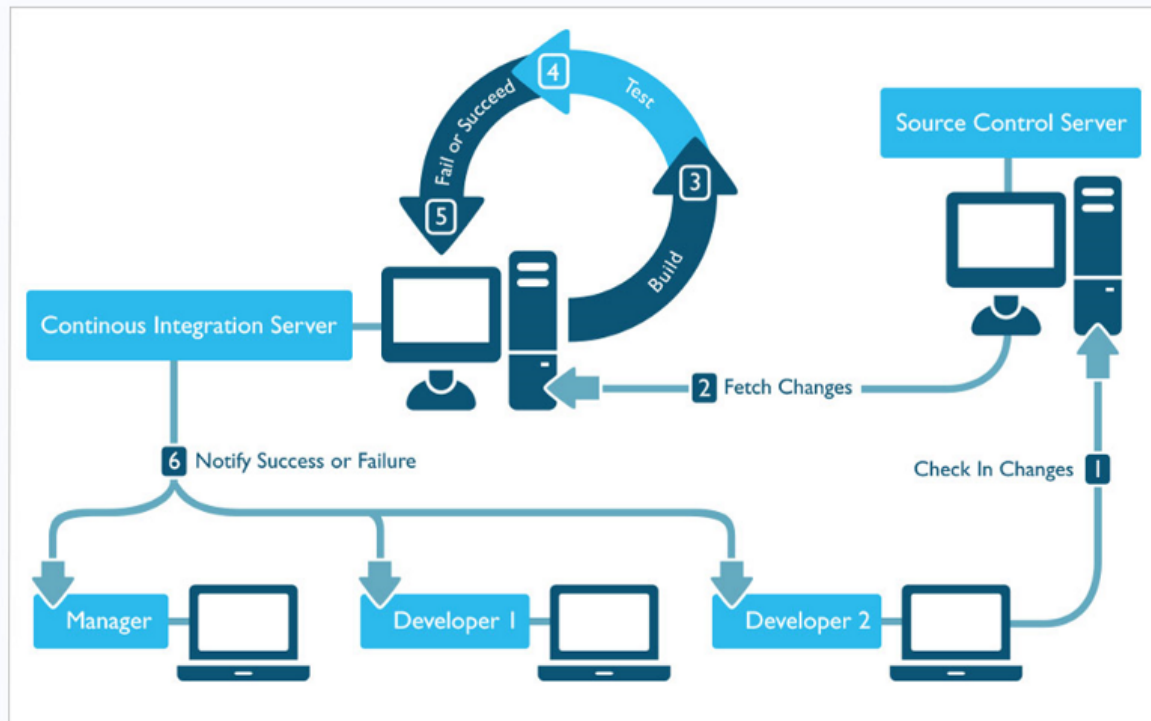
۲. تجزیه و تحلیل استاتیک.

۳. آزمایش قبل از استقرار.

۴. بسته بندی و استقرار در محیط آزمایش و تست.

۵. تست پس از استقرار.

در ادامه تصویری از فرایند CI را مشاهده خواهیم نمود:



2. Continuous Delivery یا همان تحویل مداوم، رویکردی به مهندسی نرم افزار است که بر اساس تولید نرم افزار در چرخه های کوتاه است. با توسعه در چرخه های کوتاه، تیم ها می توانند نرم افزار خود را در هر زمانی به طور قابل اعتماد منتشر کنند. با استفاده از CD، تیم های توسعه می توانند نرم افزار را سریع تر و بیشتر بسازند، آزمایش کنند و منتشر کنند. در نتیجه، آنها می توانند هزینه، زمان و ریسک ارائه هر تغییر را کاهش دهند. یک فرآیند استقرار تکرارپذیر برای تحویل مداوم مهم است. در تحویل مداوم، که می تواند به عنوان توسعه یکپارچه سازی مداوم به آن نگاه شود، توسعه دهندگان اغلب کدهای جدید را برای آزمایش به تیم های تضمین کیفیت (Quality Assurance یا QA) و عملیات تحویل می دهند. تحویل مداوم دومین مرحله از CI/CD است، روشی که تیم های توسعه برنامه را قادر می سازد تا تغییرات کد افزایشی را به سرعت و به طور منظم منتشر کنند.

در واقع، Continuous Delivery یک روش توسعه نرم افزار است که در آن تغییرات کد به طور خودکار ساخته، آزمایش می شوند و برای عرضه به تولید آماده می شوند. تحویل

مداوم پس از CI و با استفاده از استقرار همه تغییرات کد در یک محیط آزمایشی یا یک محیط تولید پس از مرحله ساخت، گسترش می‌یابد.

3. Continuous Development یا همان استقرار مداوم، مرحله بعدی تحویل مداوم است که انتشار یک محصول آماده برای تولید را به یک Code Repository، خودکار می‌کند و انتشار یک برنامه برای تولید را خودکار می‌کند.

در عمل، استقرار مداوم به این معنی است که تغییر یک برنامه‌نویس به یک برنامه ابری می‌تواند در عرض چند دقیقه پس از نوشتن آن فعال شود (البته با فرض اینکه تست خودکار را پشت سر بگذارد). این امر دریافت مداوم و ترکیب کردن بازخورد کاربران را بسیار آسان تر می‌کند. در واقع این امکان را به ما می‌دهد که ویژگی‌ها و برنامه‌های جدید را سریعتر و مکرر منتشر کنید، در حالی که نیاز به دخالت انسان در استقرار برنامه‌ها را از بین می‌برد.

لازم به ذکر است، هنگامی که تیم توسعه، برنامه‌ها را به یک روش پیاده‌سازی یا محیط‌ها را پیکربندی کند و تیم‌های عملیاتی به روشی دیگر مستقر و پیکربندی کنند، آنگاه اتوماسیون استقرار کار نخواهد کرد، در واقع هنگامی که این تیم‌ها در یک راستا نیستند، شما با خطر مدیریت دستی استقرار توسط تیم عملیاتی مواجه می‌شوید که منجر به خطاها، ناسازگاری‌ها و چرخه انتشار طولانی تر می‌شود.

برای جلوگیری از این موضوع، محیط باید سازگار باشد یا به عبارتی، فرآیند استقرار یکسان باید برای هر محیطی از جمله محیط تولید شما استفاده شود.

در مجموع، همه این روش‌های مرتبط CI/CD، استقرار یک برنامه کاربردی را کم‌ریسک تر کرده و می‌توانند به یک تیم کمک کنند تا توسعه، استقرار و آزمایش خود را خودکار نمایند.



## (B) وظایف یک متخصص CI/CD

حال سوالی که پیش می‌آید این است که وظایف متخصص ci/cd چیست؟ به طور کلی، یک متخصص ci/cd باید بتواند اصول ci/cd و pipeline را توسعه داده، مرور و اصلاح کند و همچنین باید توانایی خودکار کردن هرچه بیشتر فرایندها را داشته باشد. علاوه بر این‌ها، متخصص ci/cd، به عنوان لیدر بخش deploy، باید مهارت بالایی در برقراری ارتباط با اعضای تیم، سرپرستی و تحلیل و بررسی شرایط را دارا باشد.

- توسعه، مرور و اصلاح اصول ci/cd
- مهندس ci/cd در تمامی مسائل مربوط به ci/cd، نقش تصمیم‌گیرنده نهایی را داشته و در نتیجه به طور فعالانه در بخش توسعه و اجرای فرایندهای مرتبط مشارکت دارد.
- مرور و اصلاح اصول ci/cd
- پس از تصمیم‌گیری و مشخص کردن شیوه‌ها، متخصص ci/cd با بازدید مستمر از کارکرد و پشتیبانی آن‌های از عملیاتی که در جریان است، اطمینان حاصل می‌کند. در اینجا نیز، متخصص باید نقش رابط بین بخش‌های متفاوت تیم را اجرا کرده و همچنین از تعامل درست بین اعضای تیم و سایر واحد‌های مرتبط اطمینان حاصل کند. این بررسی‌های مستمر، به مرور باعث به ثبات رسیدن تیم خواهند شد.
- توسعه pipeline
- تمامی ابزارهای ci/cd، به مکانیزمی نیاز دارند که مراحل و بخش‌های مختلف را در اجرای فرایندهای ci/cd مشخص کند. این مکانیزم (یا pipeline)، با توجه به ابزارهای استفاده شده می‌تواند به صورت GUI و یا در کد، مشخص شود. پیکربندی این مکانیزم، اجرای مراحل مشخص شده را هماهنگ می‌کند. مسئولیت تشخیص و نگهداری از مکانیزم، تماماً بر عهده متخصص ci/cd است. همچنین متخصص، برای هماهنگی و بهینه‌سازی مراحل که در pipeline اجرا می‌شوند، با تیم‌های DevOps ارتباط برقرار می‌کنند.
- خودکار کردن فرایندها
- اتوماسیون یا خودکارسازی فرایندها، در هسته اصلی ci/cd قرار داشته و انجام

درست این بخش، تیم را از خدمات شخص سوم برای برقراری ارتباط بین پلتفرم ها و پردازش دستورات خطوط pipeline، بی نیاز می کند.

### (C) مهارت های یک متخصص CI/CD

حال که با وظایف مختلف متخصص ci/cd آشنا شدیم، به معرفی بعضی مهارت های مورد نیاز این متخصص می پردازیم

- نوشتن دستور و مراحل کار
- توانایی تفسیر و نوشتن سورس کد
- مدیریت دارایی های زیرساختی
- آشنایی با ابزار های software packing
- آشنایی با ابزارهای version control
- مدیریت ارائه دهندگان cloud
- آشنایی با ابزار های امنیت
- آشنایی با ابزار های تحلیل
- آشنایی با ابزار های نظارت

البته بدیهی است که آشنایی با تمامی ابزار های ذکر شده ضروری نبوده و تا زمانی که برنامه متخصص، به خوبی پیش برود، مشکلی به وجود نخواهد داشت.

به طور کلی می توان گفت که نقش اصلی متخصص ci/cd در تیم، مدیریت و بهینه سازی فرایندهای develop و deploy است.

## (D) معرفی ابزارها

در ادامه با دو ابزار مشهور CI/CD را معرفی میکنیم و سپس نحوه نصب و کار کردن با آن ها را توضیح میدهیم.

در ادامه برخی از ابزارهای CI/CD را معرفی میکنیم و به معرفی آنها می پردازیم.

GitLab



یکی از ابزارهای رایج، GitLab CI/CD است. این ابزار به دلیل سرعت بالا و open source بودن، محبوبیت زیادی دارد و کاربران زیادی از آن استفاده میکنند. اگر روی سیستم خود اجرا کننده در دسترس ندارید، میتوانید install GitLab Runner را نصب و برای پروژه خود یک runner ثبت نام کنید. حال به یک فایل نیاز است که عملیات CI/CD را در آن تعریف شده باشد. یک فایل با پسوند gitlab-ci.yml ساخته و در root ریپازیتوری خود قرار دهید. ساخت فایل gitlab-ci.yml ساخت فایل gitlab-ci.yml. یک فایل YAML است که میشود به صورت خاص تنظیماتش را برای gitlab ci/cd تغییر داد. در این فایل ساختار و ترتیب کارهایی که باید اجرا شوند را مشخص میکنیم و شرایط خاصی را که ممکن است به وجود بیاید و تصمیمات مربوط به آن را تعیین میکنیم.

برای ساخت این فایل در قسمت سمت چپ اسلایدبار Project information و سپس Details را انتخاب میکنیم. پس از آن شاخه ای که میخواهیم در آن کامیت کنیم را انتخاب میکنیم و روی + کلیک میکنیم و سپس new file را انتخاب میکنیم. نام فایل را gitlab-ci.yml گذاشته و داخل فایل سمپل کد زیر را قرار میدهیم.

```
build-job:

  stage: build

  script:

    - echo "Hello, $GITLAB_USER_LOGIN!"


test-job1:

  stage: test

  script:

    - echo "This job tests something"


test-job2:

  stage: test

  script:

    - echo "This job tests something, but takes more time than test-job1."
```

```
- echo "After the echo commands complete, it runs the sleep command for 20 seconds"

- echo "which simulates a test that runs 20 seconds longer than test-job1"

- sleep 20
```

deploy-prod:

stage: deploy

script:

```
- echo "This job deploys something from the $CI_COMMIT_BRANCH branch."
```




GITLAB\_USER\_LOGIN و CI\_COMMIT\_BRANCH متغیرهایی هستند که از پیش تعریف شده اند و در هنگام اجرا از آنها استفاده میکنیم و پرکار میشوند. در مرحله بعد روی commit changes کلیک میکنیم و هنگامیکه کامیت ها کامیت شدند، pipeline شروع به کار میکند. پس از ساختن اولین فایل gitlab-ci.yml. برای آنکه در آینده نیز بتوانید ادیت کنید، میتوانید از ادیتور pipeline استفاده کنید. با استفاده از ادیتور pipeline میتوانید تنظیمات آن را تغییر دهید تا به صورت اتوماتیک برخی سینتکس ها را تصحیح کند. میتوانید پیکربندی فایل gitlab-ci.yml را به صورت گرافیکی داشته باشید. اگر بخواهید اجرا کننده از یک docker container برای اجرای کارها استفاده کند، باید فایل gitlab-ci.yml را ویرایش کنید به طوری که شامل یک image با نام زیر باشد:

default:

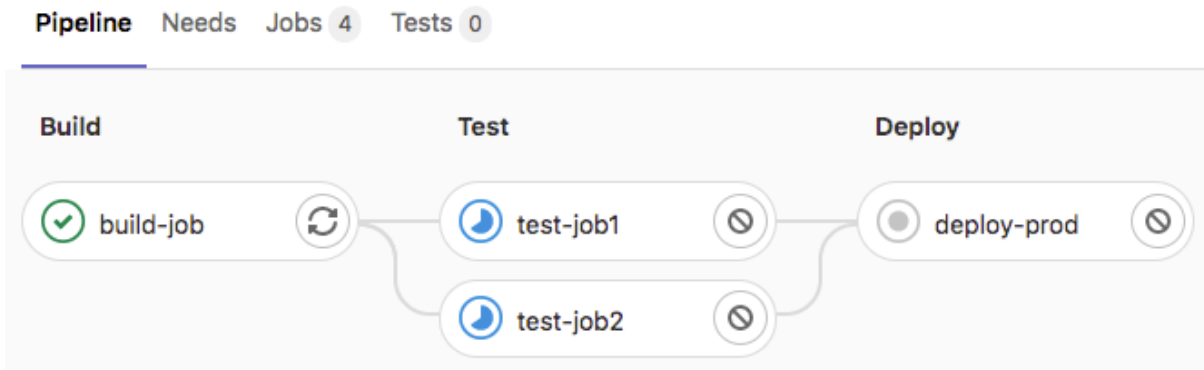
image: ruby:2.7.4

این فرایند با فرایند ساختن container داکر تفاوت دارد. در اینجا اپلیکیشن نیاز ندارد که یک container داکر بسازد تا کارهای CI/CD را در container اجرا کند. هرکدام از کارها شامل اسکریپت ها و مراحل می شود که کلیدواژه default برای پیشفرض های سفارش شده استفاده میشود Stage. مراحل انجام کار را بصورت متوالی تعیین میکند و در صورت در دسترس بودن اجراکننده ها هر کار به صورت موازی اجرا میشود .

میتوانید تنظیمات اضافه دلخواهتان را اضافه کنید و ساختار آن را شخصی سازی کنید که کارها به چه صورت اجرا شوند. از کلیدواژه rules میتوانید استفاده کنید تا به صورت خاص کاری یا اجرا کند یا آن را رد کند. همچنین میتوانید اطلاعات مربوط به هر کاری را در cache و artifacts در یک pipeline داشته باشید. میتوانید استاتوس های pipeline و کارها را ببینید. با کامیت کردن تغییراتتان، pipeline روشن میشود. برای آنکه pipeline را بتوانید ببینید به CI/CD و سپس به Pipeline بروید. یک pipeline سه مرحله دارد که مشابه تصویر زیر است:

Status	Pipeline	Triggerer	Commit	Stages
 running	#217408060 latest		master -> 271ad0cd Update .gitlab-ci.yml	

برای مشاهده تصویر pipeline، روی ID pipeline کلیک کنید و تصویری مشابه تصویر زیر رو مشاهده خواهید کرد:



برای مشاهده جزئیات هر کار، میتوانید روی نام کار کلیک کنید؛ برای مثال اگر روی deploy-prod کلیک کنیم تصویر زیر را خواهیم داشت:

✓ passed Job #855275091 triggered 23 minutes ago by Suzanne Selhorn

```
1 Running with gitlab-runner 13.6.0-rc1 (d83ac56c)
2   on docker-auto-scale ed2dce3a
3   ✓ Preparing the "docker+machine" executor
4   Using Docker executor with image ruby:2.5 ...
5   Pulling docker image ruby:2.5 ...
6   Using docker image sha256:b7280b81558d31d64ac82aa66a9540e04baf9d15abb8fff
   ed62cd60e4fb5bf4132943d6fa2688 ...
7   ✓ Preparing environment
8   Running on runner-ed2dce3a-project-16381496-concurrent-0 via runner-ed2dc
9   ✓ Getting source from Git repository
10  $ eval "$CI_PRE_CLONE_SCRIPT"
11  Fetching changes with git depth set to 50...
12  Initialized empty Git repository in /builds/sselhorn/test-project/.git/
13  Created fresh repository.
14  Checking out 7353da73 as master...
15  Skipping Git submodules setup
16  ✓ Executing "step_script" stage of the job script
17  $ echo "This job deploys something from the $CI_COMMIT_BRANCH branch."
18  This job deploys something from the master branch.
19  ✓ Cleaning up file based variables
20  Job succeeded
```

توجه: اگر استاتوس یک کار، stuck بود، بررسی کنید و مطمئن شوید که یک اجراکننده به پروژه متصل شده است.

TeamCity



## TeamCity

یکی دیگر از ابزارهای رایج TeamCity است که توسط JetBrains معرفی شده است . نصب برنامه

ابتدا برنامه را از دانلود میکنید. برای دانلود برنامه میتوانید از این [لینک](#) استفاده کنید. پس از دانلود برنامه آن را نصب کنید. برای نصب برنامه میتوانید از این [لینک](#) کمک بگیرید.

پس از آنکه برنامه را نصب و اجرا کردید، می توانید پروژه خود را بسازید. برای ساخت پروژه میتوانید روی Administration که در گوشه سمت راست بالا رابط کاربری TeamCity قرار دارد کلیک کنید و سپس با کلیک بر روی create project پروژه خود را بسازید.

در ادامه به شرح نحوه ساخت پروژه با استفاده از URL ریپازیتوری میپردازیم.

- استفاده از URL ریپازیتوری

در صفحه create project روی from a repository کلیک کنید و URL پروژه خود را وارد کنید TeamCity. فرمت های URL مختلفی را ساپورت میکند که در این [لینک](#) میتوانید مشاهده کنید.

در صورتیکه دسترسیتان به ریپازیتوری محدود شده است باید credentials را نیز وارد کنید.



Administration / <Root project>

## Create Project

From a repository URL

From GitHub  
Set up connection

From Bitbucket Cloud  
Set up connection

From Visual Studio Team Services  
Set up connection

Manually

Parent project: \* <Root project>

Repository URL: \*  A VCS repository URL. Supported formats: http(s)://, svn://, git://, etc. as well as URLs in Maven format.

Username:  Provide username if access to repository requires authentication.

Password:  Provide password if access to repository requires authentication.

[Proceed](#)

روی Proceed کلیک کنید و سپس TeamCity نوع ریپازیتوری شما را مشخص میکند و اتصالات را بررسی میکند و به صورت اتومات تنظیمات ریپازیتوریتان را بررسی میکند.

Administration / <Root project>

## Create Project From URL

✓ The connection to the VCS repository has been verified

Project name: \*

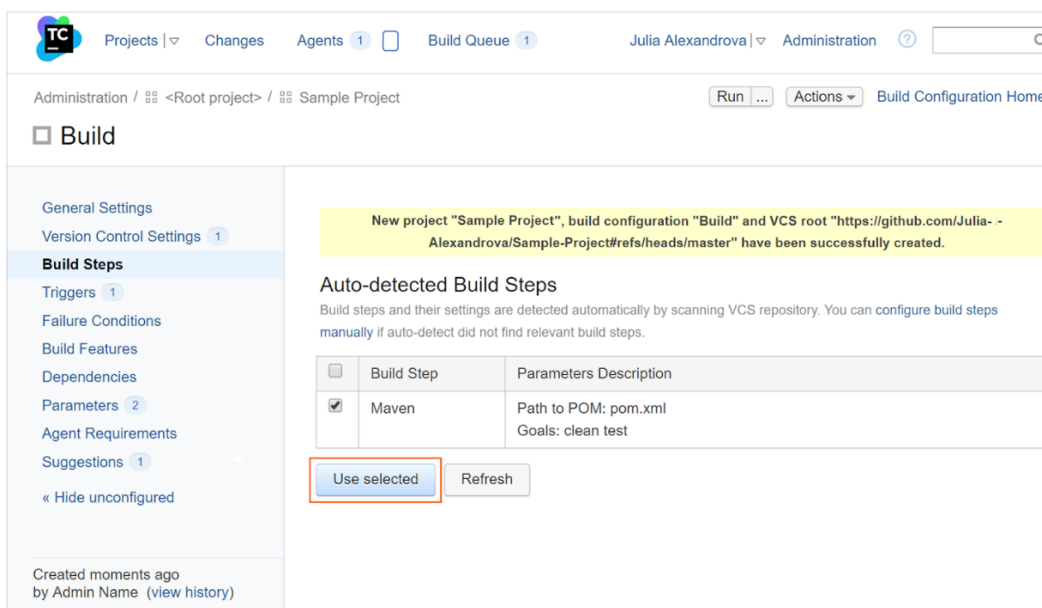
Build configuration name: \*

VCS root: (Git) https://github.com/Julia-Alexandrova/Sample-Project

[Proceed](#) [Cancel](#)

روی Proceed کلیک کنید تا TeamCity، در ریپازیتوریتان VCS را اسکن کند و به صورت خودکار، مرحله های پروژه را تشخیص دهد.

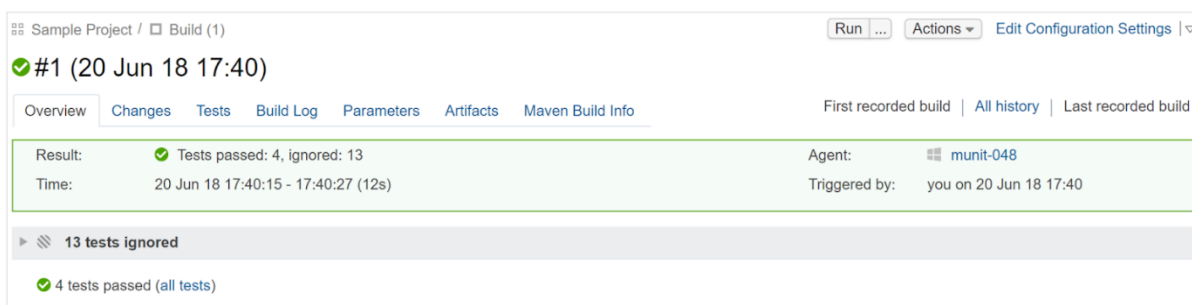
حال میتوانید مرحله‌ای که مناسب‌تان است را علامت بزنید تا به پیکربندی پروژه اضافه شود.



حال مراحل ساخت پروژه تمام شده است و میتوانید پروژه تان را اجرا کنید.

- اجرای پروژه

برای اجرای پروژه در صفحه Build Configuration Settings بر روی run کلیک میکنیم و پروژه اجرا میشود و صفحه نتایج باز میشود و از طریق این صفحه میتوانید استاتوس هر کاری را ببینید و به تنظیمات پیکربندی دسترسی داشته باشید و اگر لازم بود، آن را ویرایش کنید.



- تغییر تنظیمات پیکربندی

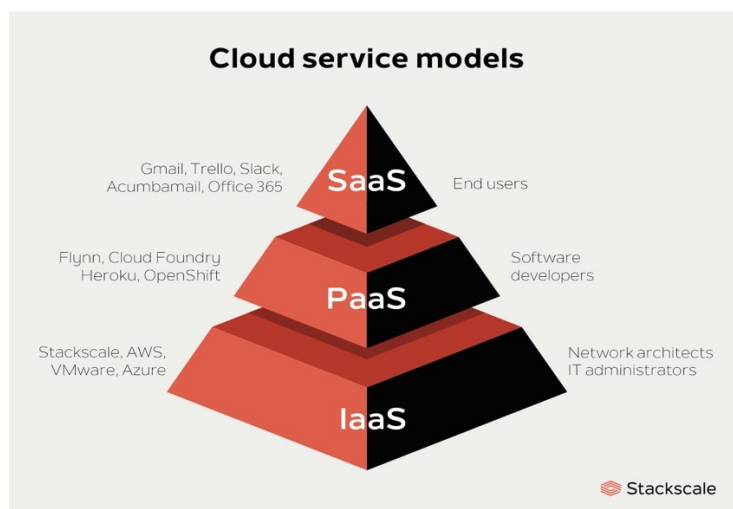
ممکن است در ابتدا بخواهید تنظیماتی مانند مسیرهای ساخت مصنوعات، trigger های خودکار یا یک الگوی خاص برای شماره build را پیکربندی کنید. TeamCity این امکان را فراهم کرده است تا در صورت نیاز این تنظیمات را تغییر دهید.

## (E) سرویس های آماده CI/CD در داخل و خارج از ایران

همانطور که در بخش های قبلی هم اشاره شد، ابزار های متفاوتی برای دیپلوی و تست پیوسته نرم افزارها وجود دارد اما اینکه هر بار بخواهیم این ابزارها را تنظیم کنیم و از پایه و به عبارت معروف from scratch ابزارهای CI/CD را بالا بیاوریم، کار سختی است و امروزه کمتر سراغ این روش می روند و علی رغم اینکه شرکت های با زیرساخت های بزرگ و قوی همچنان با ابزارهای CI/CD تمام تنظیمات را بالا می آورند و از ابزار ها استفاده می کنند، ولی شرکت هایی در داخل ایران و همچنین خارج از کشور هستند که خدمات CI/CD را به صورت آماده در اختیار مشتریان قرار می دهند. به عبارتی با رابط گرافیکی و کاربری تمام امکانات مورد نیاز را به صورت یکجا در اختیار کاربران قرار می دهند.

به عنوان نمونه به جای اینکه از سرویس هایی مثل IaaS که مخفف Infrastructure as a Service است، استفاده کنیم می توان از سرویس های PaaS یا Platform as a Service استفاده کنیم که کار را بسیار ساده تر می کنند و می توان بدین صورت نرم افزار را به سادگی با کمترین دردسر بالا آورد. نکته ی مهم اینگونه سرویس این است که برای شرکت های کوچک و استارتاپ ها بسیار مناسب هستند و شرکت های نوپا در صورتی که از آنها استفاده کنند می توانند با هزینه کمتر و نیروی انسانی کمتر در زمان کمتر نرم افزار های خود را به بهره برداری برسانند.

سرویس های متفاوتی از این نوع در داخل و خارج از ایران به منظور ابزار CI/CD وجود دارد که در ادامه به یکی از نمونه های داخلی و یکی از نمونه های خارجی این سرویس ها می پردازیم:



## - سرویس AWS



یکی از معروف ترین سایت هایی که در خارج از ایران همواره برای سرویس های Paas و laas و بقیه ی سرویس های ابری و Cloud آن مورد استفاده قرار می گیرد ولی معمولاً به دلیل اینکه برای ما پرداخت هزینه ی آن در ایران با مشکل همراه است نمی توانیم از آن استفاده کنیم، Amazon Web Services است که به اختصار AWS نامیده می شود.

در ابتدا کافیسیت وارد [سایت AWS](#) شویم و یک حساب کاربری بسازیم. البته به دلیل اینکه ما در ایران حساب بانکی نداریم که بتوانیم از آن به صورت بین المللی استفاده کنیم، نمی توانیم از سرویس های AWS به سادگی استفاده کنیم.



Explore Free Tier products with a new AWS account.

To learn more, visit [aws.amazon.com/free](https://aws.amazon.com/free).



## Sign up for AWS

### Email address

You will use this email address to sign in to your new AWS account.

### Password

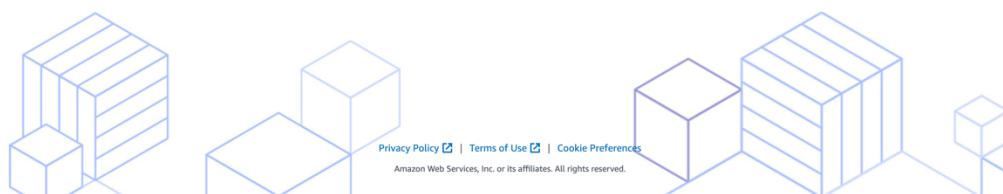
### Confirm password

### AWS account name

Choose a name for your account. You can change this name in your account settings after you sign up.

[Continue \(step 1 of 5\)](#)

[Sign in to an existing AWS account](#)



[Privacy Policy](#) | [Terms of Use](#) | [Cookie Preferences](#)

Amazon Web Services, Inc. or its affiliates. All rights reserved.


برای بردن یک ریپازیتوری گیت هاب روی سرویس Pipeline سایت AWS کافیست در ابتدا وارد گیت هاب و این ریپازیتوری شویم و سپس یک GitHub Access Token ایجاد کنیم و در نهایت یک access token برای دسترسی AWS به این گیت هاب ایجاد کنیم.

Settings / Developer settings

OAuth Apps  
GitHub Apps  
Personal access tokens

### New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Token description  
eks-workshop  Token name

What's this token for?

Select scopes  
Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo:deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input type="checkbox"/> admin:org	Full control of orgs and teams
<input type="checkbox"/> write:org	Read and write org and team membership
<input type="checkbox"/> read:org	Read org and team membership
<input type="checkbox"/> admin:public_key	Full control of user public keys
<input type="checkbox"/> write:public_key	Write user public keys
<input type="checkbox"/> read:public_key	Read user public keys
<input type="checkbox"/> admin:repo_hook	Full control of repository hooks
<input type="checkbox"/> write:repo_hook	Write repository hooks
<input type="checkbox"/> read:repo_hook	Read repository hooks
<input type="checkbox"/> admin:org_hook	Full control of organization hooks
<input type="checkbox"/> gist	Create gists
<input type="checkbox"/> notifications	Access notifications
<input type="checkbox"/> user	Update all user data
<input type="checkbox"/> read:user	Read all user profile data
<input type="checkbox"/> user:email	Access user email addresses (read-only)
<input type="checkbox"/> user:follow	Follow and unfollow users
<input type="checkbox"/> delete_repo	Delete repositories
<input type="checkbox"/> write:discussion	Read and write team discussions
<input type="checkbox"/> read:discussion	Read team discussions
<input type="checkbox"/> admin:gpg_key	Full control of user gpg keys ( <a href="#">Developer Preview</a> )
<input type="checkbox"/> write:gpg_key	Write user gpg keys
<input type="checkbox"/> read:gpg_key	Read user gpg keys

[Generate token](#) [Cancel](#)

در نهایت این access token را ذخیره می کنیم تا در سرویس AWS از آن استفاده نماییم.

Settings / Developer settings


OAuth Apps  
GitHub Apps  
Personal access tokens

### Personal access tokens

[Generate new token](#) [Revoke all](#)

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your new personal access token now. You won't be able to see it again!

✓ 1a76...9e21  [Delete](#)

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

در نهایت کافیت وارد پنل AWS خود شویم و از قسمت CloudFormation می توان Code Pipeline را انتخاب نمود و پایپ لاین کد را ایجاد کرد که به صورت زیر خواهد بود:

**Stack name**

Stack name

eksws-codepipeline

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

**Parameters**

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

**GitHub**

GitHubUser  
Username

testgitrz

Enter username

GitHubToken  
Access token

[redacted]

Enter access token

GitSourceRepo  
Repository

eks-workshop-sample-api-service-go

GitBranch  
Branch

master

**CodeBuild**

CodeBuildDockerImage  
Docker image

در صفحه ای که باز می شود کافیت GitHub و GitHub Access Token را بنویسیم که در نتیجه AWS با اتصال به گیت هاب ما و راه اندازی پایپ لاین فرآیند بیلد، دیپلوی و تست کردن پروژه را انجام می دهد. در نتیجه چند دقیقه باید صبر کنیم که همه ی فرآیندها Complete شود.

CloudFormation > Stacks

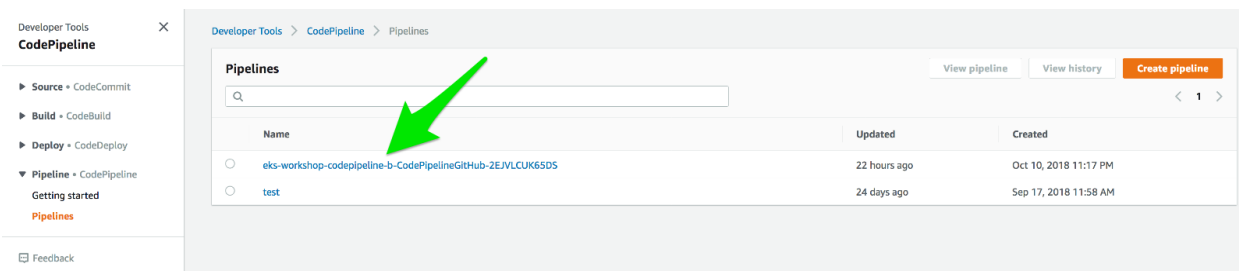
**Stacks**

Active Filter stacks

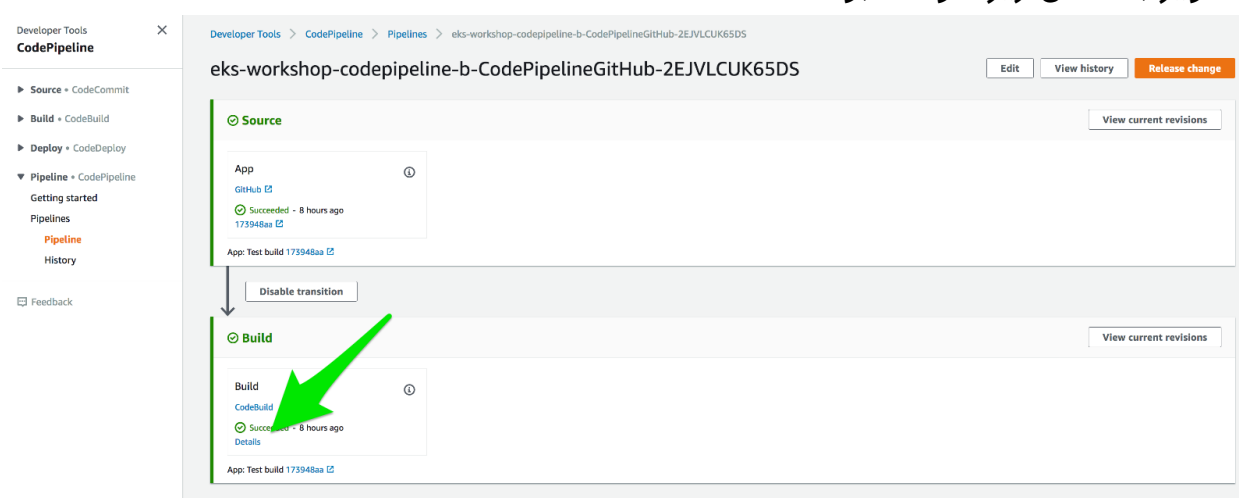
Create stack

Stack name	Status	Created time	Description
eksws-codepipeline	CREATE_IN_PROGRESS	Wed, 21 Nov 2018 15:35:28 GMT	EKSWSV1
eksctl-eksworkshop-eksctl-nodegroup-0	CREATE_COMPLETE	Wed, 21 Nov 2018 15:16:40 GMT	EKS nodes (AMI family: AmazonLinux2, SS...
eksctl-eksworkshop-eksctl-cluster	CREATE_COMPLETE	Wed, 21 Nov 2018 15:05:59 GMT	EKS cluster (dedicated VPC: true, dedicate...
aws-cloud9-eksworkshop-148a5100e7d7...	CREATE_COMPLETE	Wed, 21 Nov 2018 14:59:55 GMT	-

برای مشاهده جزئیات فرآیند دیپلوی کردن باید به CodePipeline در Management Console مراجعه کنیم و جزئیات را مشاهده کنیم.



در نتیجه با کلیک کردن روی اسم پایپ لاین می توانیم از موفقیت آمیز طی شدن همه ی فرآیند های پایپ لاین مطمئن شویم که در صورت موفقیت آمیز بودن همه چیز تصویر به شکل زیر خواهد بود.




حال به سادگی و بدون نیاز به اینکه بخواهیم به صورت کامل از ابتدا سرویس هایی مثل گیتلر را راه اندازی کنیم، می توانیم با هر کامیتی که در گیت هاب می کنیم پروژه را به صورت دیپلوی شده روی سرور مشاهده کنیم.




```

12 func main() {
13     http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
14
15         f := fib()
16
17         res := &response{Message: "Hello World"}
18
19         for _, e := range os.Environ() {
20             pair := strings.Split(e, "=")
21             res.EnvVars = append(res.EnvVars, pair[0]+"="+pair[1])
22         }
23
24         for i := 1; i <= 90; i++ {
25             res.Fib = append(res.Fib, f())
26         }
27
28         // Beautify the JSON output
29         out, _ := json.MarshalIndent(res, "", " ")
30
31         // Normally this would be application/json, but we don't want to prompt downloads
32         w.Header().Set("Content-Type", "text/plain")
33
34         io.WriteString(w, string(out))
35
36         fmt.Println("Hello world - the log message")
37     })
38     http.ListenAndServe(":8080", nil)
39 }

```

 **Change text**

**Commit changes**

Change the Message text  **Commit message**

Add an optional extended description...

☒ Commit directly to the master branch.

☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

**Commit changes** **Cancel**

به محض کامیت کردن می توانیم در پنل AWS از موفقیت آمیز بودن فرآیند دیپلوی به صورت زیر مطمئن شویم.

Developer Tools

**CodePipeline**

Developer Tools > CodePipeline > Pipelines > eks-workshop-codepipeline-b-CodePipelineGitHub-2EJVLCK65DS

eks-workshop-codepipeline-b-CodePipelineGitHub-2EJVLCK65DS

Edit View history Release change

View current revisions

**Source**

App

GitHub

In progress - Just now

Disable transition

**Build**

Build

CodeBuild

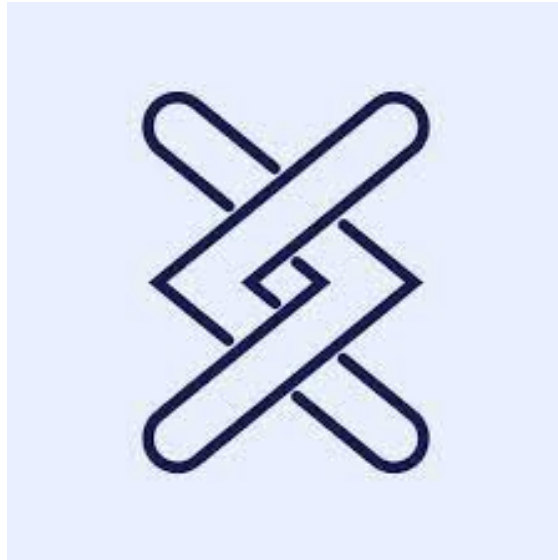
Succeeded - 8 hours ago

Details

App: Test build 173948aa

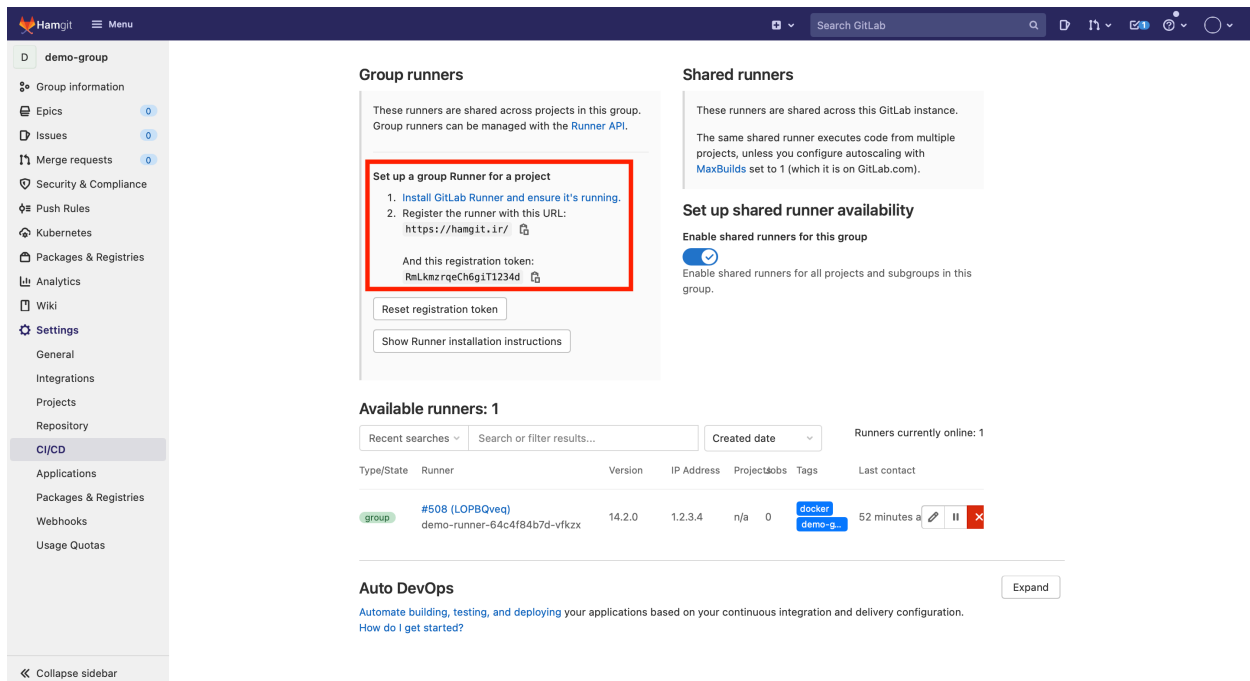
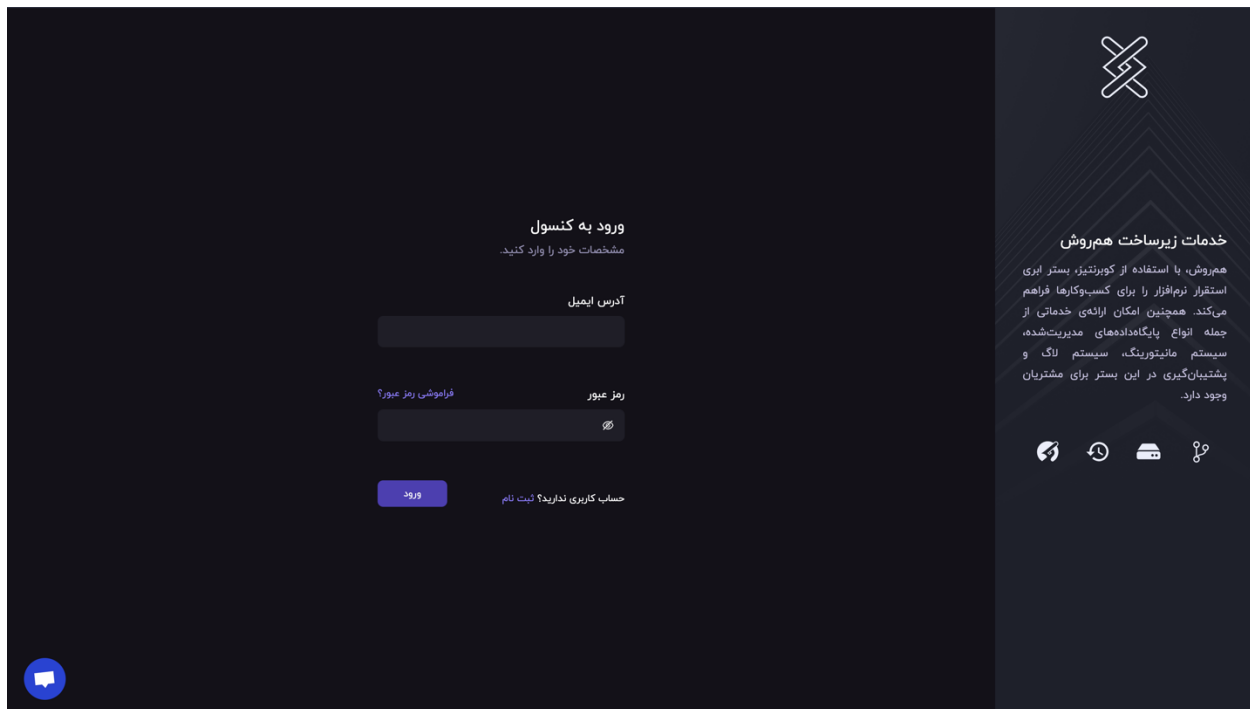
در این صورت فرآیند Build، Deploy و Test به صورت اتوماتیک همگی انجام می شود.

## - سرویس دارکوب (از محصولات همروش)

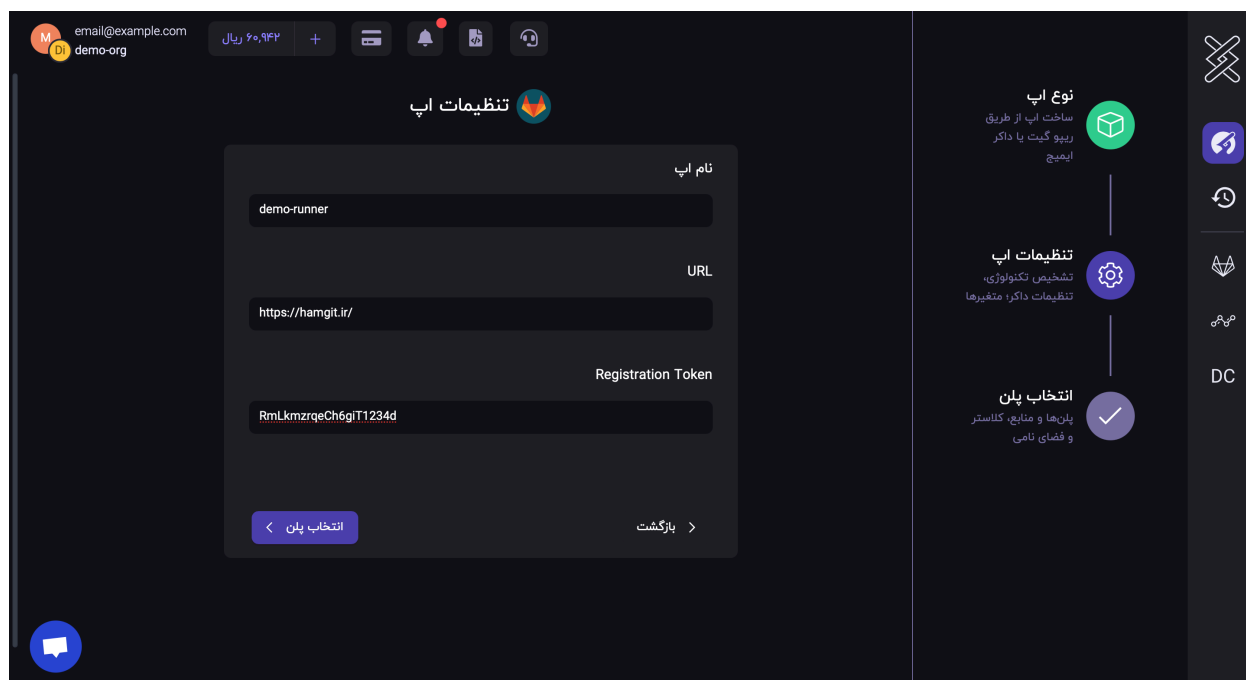


یکی از سرویس های جایگزین برای AWS را می توان سرویس دارکوب را که توسط همروش ارائه شده است دانست. این سرویس هم یک Platform as a Service است که به عنوان pipeline های CI/CD ارائه می شود. کافیسیت ابتدا وارد [سایت همروش](#) شویم و یک حساب کاربری ایجاد کنیم. سپس در مرحله بعد کافیسیت که ریپازیتوری هایی که می خواهیم پایپ لاین ها را روی آن اجرا کنیم، به سرویس گیتلب رانر اضافه کنیم. در نتیجه کافیسیت از منوباری که در سمت چپ سایت وجود دارد CI/CD را به عنوان سرویس انتخاب کنیم و این ریپوها را به آن اضافه نماییم.

این فرآیند در سرویس CI/CD دارکوب و شرکت همروش به صورت ساده تر از AWS هم انجام می شود و به صرت ساده می توان این فرآیند شامل چرخه ی Build، Deploy و Test را به صورت اتوماتیک انجام داد و با هر کامیت می توانیم خروجی را به صورت همزمان روی سرور مشاهده کنیم.



سپس وارد کنسول همروش سرویس دارکوب می شویم و با ساخت اپ آماده GitLab Runner یک Paas ایجاد می کنیم. در نهایت کافیسیت نام اپ، URL و همچنین Registration Token را به درستی در فیلدهای ورودی وارد کنیم.



در نهایت کافیه با انتخاب پلن مورد نظر ساخت اپ تکمیل شود و در نهایت به سادگی GitLab Runner قابل استفاده خواهد بود. می بینیم که فرآیند استفاده از دارکوب بسیار ساده تر از استفاده از گیتلب رانر به صورت خام است و بدون اینکه بخواهیم فرآیند استقرار را از صفر انجام دهیم، می توانیم با استفاده از رابط کاربری این پیاده سازی را انجام دهیم.

## (F) منابع

[https://docs.gitlab.com/ee/ci/quick\\_start/](https://docs.gitlab.com/ee/ci/quick_start/)

<https://www.katalon.com/resources-center/blog/ci-cd-tools/>

<https://www.jetbrains.com/help/teamcity/configure-and-run-your-first-build.html#Create+your+first+project>

<https://www.jetbrains.com/help/teamcity/configure-and-run-your-first-build.html#Artifacts>

<https://www.synopsys.com/glossary/what-is-cicd.html>

<https://www.synopsys.com/glossary/what-is-continuous-integration.html>

<https://www.synopsys.com/glossary/what-is-continuous-delivery.html>

<https://www.redhat.com/en/topics/devops/what-is-ci-cd>

<https://developers.redhat.com/blog/2017/09/06/continuous-integration-a-typical-process>

<https://hamraves.com/gitlab-runner>

<https://hamraves.hs3.ir/landing/hamraves/hamraves-logo-banner.jpg>

<https://www.stackscale.com/wp-content/uploads/2020/04/cloud-service-models-iaas-paas-saas-stackscale.jpg>

<https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSo1LNq92EZNb9T0L2sj-tYH0rh7SC0QPTsg3OWQ310w5NMxQM1Jqll1xsI94IFzMGwYLk&usqp=CAU>

[https://www.eksworkshop.com/intermediate/220\\_codepipeline/change/](https://www.eksworkshop.com/intermediate/220_codepipeline/change/)

<https://aws.amazon.com/codepipeline/>