**GitLabRunner**

خیلی جالب گفته. حتما سایتش رو پیدا کنم و بقیش و صفحه 8 که در مورد .yml صحبت کرده به انجا برم و آنها رو نیز بیارم اینجا

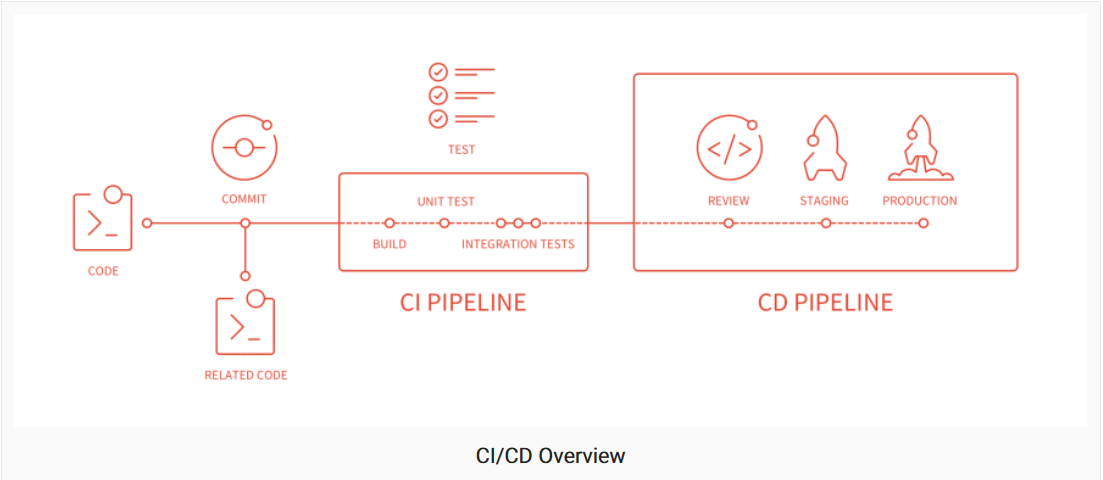
برای اجرای رویکردهای مدرن برای استقرار کد(CI/CD) ابزارهایی وجود دارد از جمله : Jenkins (این مورد بسیار پیچیده و دشوار است) ، [CodeDeploy](https://aws.amazon.com/codedeploy/) AWS ، GitLab ، و در انتها نیز گیت‌هاب [Actions](https://developer.github.com/actions/) .

اگر سازمان شما بر روی GitLab فعالیت می‌کند، بهتر است که از همین مورد کارتان را شروع کنید، چرا که به خوبی با سایر بخش‌های GitLab ادغام شده است.

تیم برنامه نویسی به CI و CD احتیاج دارد تا بتواند به صورت یکپارچه فعالیت کند  و GitLab CI / CD رتبه 1 در Forrester CI Wave rated را دارا می باشد.

در واقع CI/CD میاد source و روال build شما رو با هم integrate می­کنه و در نهایت اون رو به محیط production شما deliver می­کنه.

فرض کنید اپلیکیشنی دارید که به تازگی بالا اومده و یک سری مشتری دارن باهاش کار میکنند . حالا اگه روال نسخه گذاریتون وقت گیر باشه و یه باگی هم توی سیستم به وجود بیاد ، چقدر روال توسعه و پشتیبانی سیستم برای رفع مشکلات به وجود آمده میتونه ترسناک باشه.



gitlab runner یک برنامه باینری به زبان Go است و نیازی ندارد که کتابخانه ی اضافی روی سیستم نصب شود (مگر اینکه از سورس کامپایل شود). cross platform است و اگر در کنار آن می خواهید از docker استفاده کنید آخرین ورژن آن را نصب کند زیرا gitlab runner  حداقل به ورژن 1.13.0 داکر نیاز دارد.

**ویژگی های Gitlab Runner :**

* **اجازه اجرا:**
  + چندین کار همزمان
  + از چندین token با چندین سرور (حتی برای هر پروژه).
  + تعداد محدود مشاغل همزمان به ازای هر token
* **job ها می تواند صورت زیر اجرا شود :**
  + به صورت local
  + به کمک  داکر کانتینر
  + به کمک  داکر کانتینر و اجرای job ها از طریق ssh
  + به کمک  داکر کانتینر و پخش برو روی ابرهای مختلف و hypervisor های مجازی سازی.
  + اتصال به سرور SSH از راه دور.
* **پشتیبانی از bash – windows  batch و powershell**
* **بارگذاری تنظیمات به ورت خودکار بون نیاز به ریستارت**
* **فعال سازی کشینگ کانتینر های داکر**
* **نصب آسان در همه توزیع ها و سیستم عامل ها**
* **قابلیت شخصی سازی job  ها**

**نکته:** ورژن Gitlab runner می­بایست با ورژن gitlab شما یکی باشد. شاید تفاوت ورژن در آن ها کار کند ولی بعضی از ویژگی­ها و feature ها ممکن است به صورت درست فعالیت نکنند.

**مراحل پیاده سازی Gitlab Runner**

* نصب gitlab runner
* register کردن gitlab runner
* پیکربندی و تنظیم یک پروژه

**نصب gitlab runner**

GitLab Runner را می­توان در GNU / Linux ، macOS ، FreeBSD و Windows نصب و مورد استفاده قرار داد.

می توانید آن را با استفاده از Docker نصب کنید و یا  باینری را به صورت دستی بارگیری کنید یا هم  از مخزن بسته های rpm / deb استفاده کنید که GitLab ارائه می دهد.

روش پیشنهادی استفاده از رپازیتوری است. رپازیتوری را با دستورات زیر اضافه و gitlab-runner را اجرا می­کنیم

# For Debian/Ubuntu/Mint

$ curl -L https://packages.gitlab.com/install/repositories/runner/gitlab-runner/script.deb.sh | sudo bash

$ sudo apt-get install gitlab-runner

# For RHEL/CentOS/Fedora

$ curl -L https://packages.gitlab.com/install/repositories/runner/gitlab-runner/script.rpm.sh

$ sudo yum install gitlab-runner

شما می توانید برای امنیت بالاتر یک یوزر به نام gitlab-runner ایجاد و مسوولیت اجرای این ابزار را به آن بدهید ولی در این مقاله من می خوام از یوزر روت به عنوان اجرا کننده job ها در runner استفاده کنم (در محیط عملی حتما یوزر بسازین. من حال ندارم . خخخ) بعد ار انتصاب یوزر استارتش رو میزنیم

**$ gitlab-runner install –user=root –working-directory=/root**

**$ gitlab-runner start**

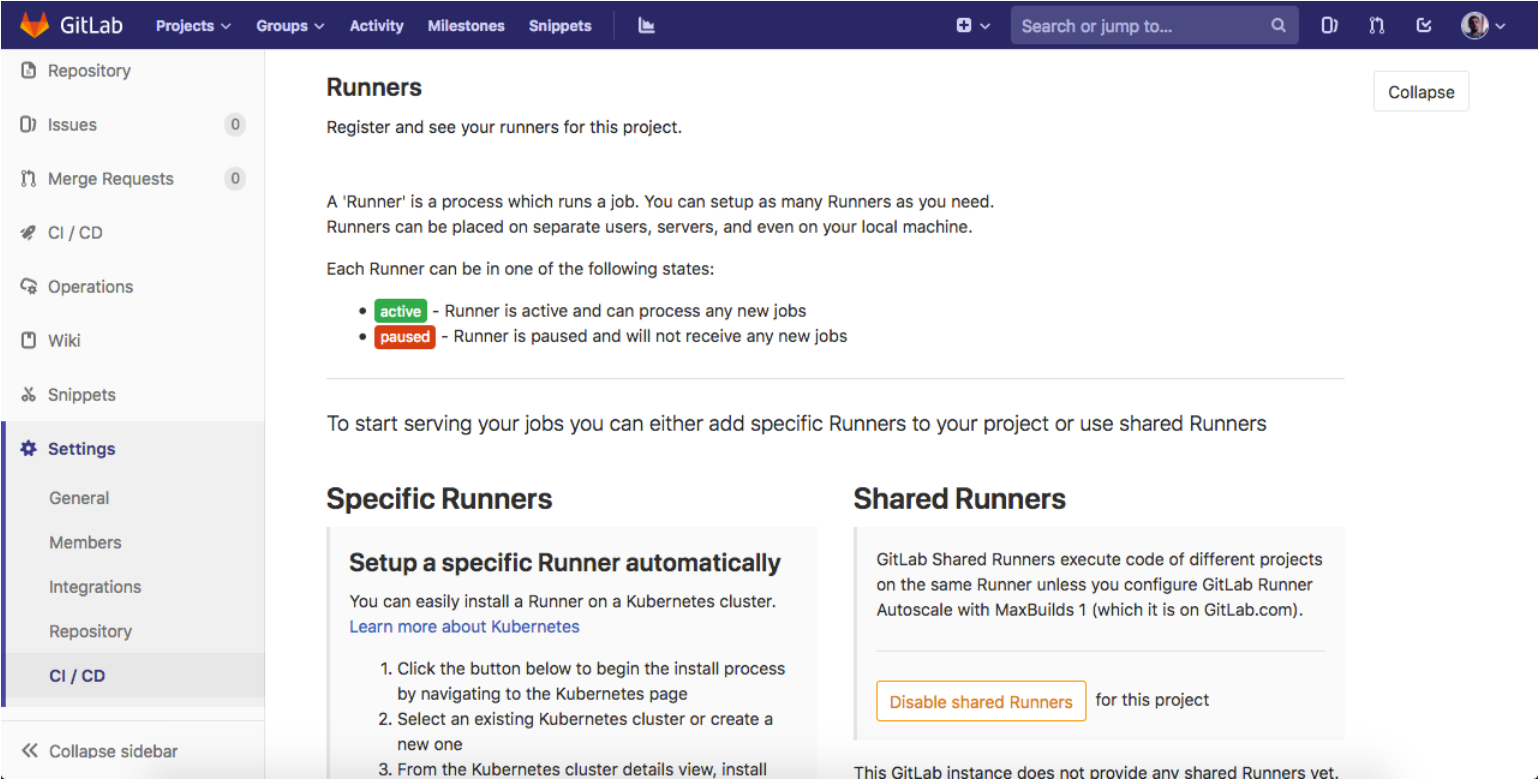
خب در اینجا نصب Gitlab Runner به پایان رسید. می تونید با کامند زیر از صحت نصب مطمئن بشید

$ gitlab-runner –version

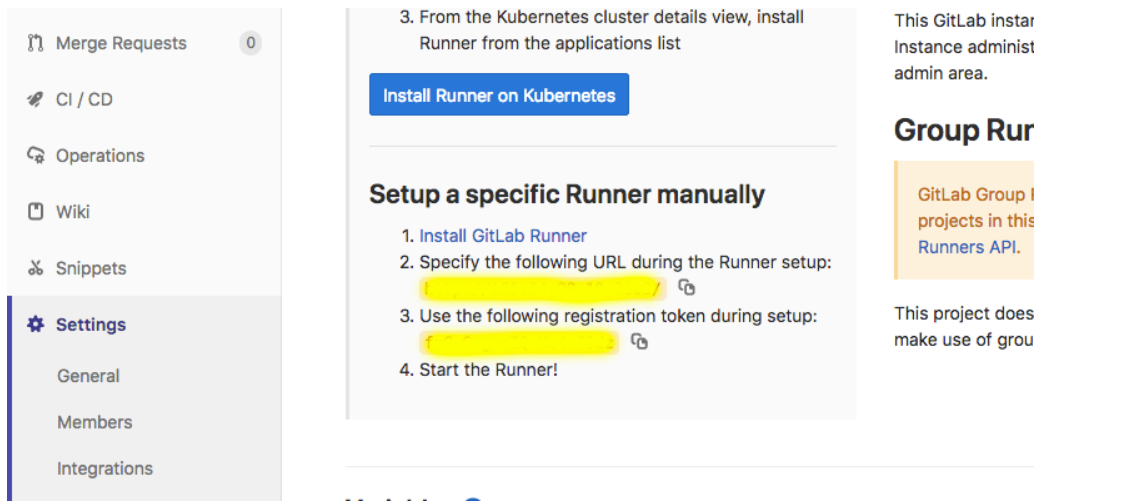
**register کردن gitlab-runner**

register کردن یک Runner فرایندی است که Runner را با نمونه GitLab پیوند می دهد.

بدین منظور شما نیاز به ۲ مورد دارین: آدرس url سرور گیت لب و token  که می توانید به صفحه­ی setting CICD در گیت لب آن پروژه رفته و در قسمت runner مشاهده کنید.



پایین تر برید تا به بخش Setup a specific Runner manually برسید



الان بریم سراغ register یا شناسوندن ( معادل فارسی بهتری به ذهنم نرسید )

**$ gitlab-runner register**

بعدش میاد آدرس و توکن و نام و تگ و نوع script اجرایی رو می پرسه و شما هم اطلاعات رو وارد می کنید و سپس اینتر رو می زنید. البته حواس تون هست که نام و تگ runner هم مهمه چون بعدا لازمش داریم (پس یه اسم معنی دار و درست و حسابی براش تعیین کنید مثلا development یا production یا … !!!)

Running in system-mode.

Please enter the gitlab-ci coordinator URL (e.g. https://gitlab.com/):>

آدرسی که توی گیت لب دارید

Please enter the gitlab-ci token for this runner:>

توکن پایینش توی همون گیت لب

Please enter the gitlab-ci description for this runner: >

اضافه کردن یک توضیح برای آن رانر

Please enter the gitlab-ci tags for this runner (comma separated): >

اضافه کردن یکسری تگ به رانر production,root

Registering runner… succeeded runner=fx-sadGwkx Please enter the executor: docker-ssh, docker-ssh+machine, ssh, virtualbox, docker+machine, kubernetes, docker, parallels, shell: >

  مشخص کننده اجرا کننده shell

Runner registered successfully. Feel free to start it, but if it’s running already the config should be automatically reloaded!

همون جا توی گیت لب اگر صفحه رو رفرش کنید یه بخشی دقیقا پایین عکس قبلی به نام Runners activated for this project ایجاد می شه که وضعیت اتصال runner ها رو نشون می ده (برای هر پروژه می تونید چندین runner داشته باشید) و حدودا بعد ۳ الی ۴ دقیقه وقتی که سبز شد یعنی که همه چیز درست انجام شده.

**پیکربندی و تنظیم یک پروژه**

برمیگردیم به gitlab که source countrol ماست. gitlab میگه که من میتونم به شما CI/CD ارائه بدم و در قالب یک سری stage بیام source شما رو بخونم و بسته به نوع پروژه ، مثلا اگر node js  هست با npm بیلد رو انجام بدم .

خود گیت لب برای اینکه بتونه این کار رو انجام بده داره از مکانیزم داکر استفاده میکنه ، یعنی داخل خودش یک سری container های داکری رو داره اجرا میکنه برای اینکار از یک فایل به اسم ‘gitlab-ci.yml.’ استفاده میکنه که در اون stage های مختلف رو میخونه و اجرا میکنه .

این stage ها در واقع مراحل انجام کار هستند مثلا اولین مرحله build کردن ، دومین مرحله ساخت image و سومین مرحله اجرا ، روی سرور ما است.

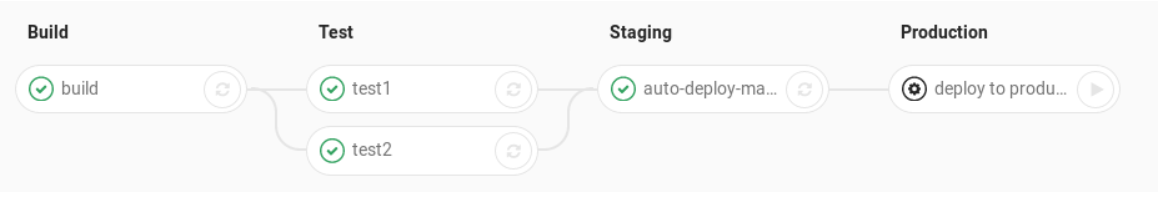
مکانیزم گیت لب به این شکل هست که وقتی یک stage اجرا میشه و complete میشه همه چیزش پاک میشه یعنی یه جورایی هر stage یک temp محسوب میشه. که در ادامه میگم که این به چه معنی هست و برای ارتباط برقرار کردن بین این stage ها و ردل و بدل کردن دیتا چه کار هایی رو باید انجام بدیم

**پایپ لاین ها (Pipelines)**

یک pipeline مجموعه ای از job هایی هست که در قالب چندین stage اجرا میشن و از طریق منو pipeline قابل دسترسی هستند. اگر تمامی job ها با موفقیت اجرا بشن ، pipeline به stage بعدی میره اما اگه یکی از job ها fail بشه ، pipeline همونجا متوقف میشه

یه فایلی رو بایستی با نام .gitlab-ci.yml  توی روت پروژه تون ایجاد کنید

در عکس زیر عکس یک pipeline رو مشاهده میکنید که متشکل از چهار stage هست که هر کدوم حداقل یکی یا بیش از یک job دارن



خود GitLab در [اینجا](https://docs.gitlab.com/ee/ci/yaml/) یه داکیومنت فوق العاده داره برای ایجاد و تنظیم این فایل و البته به با کلی مثال در [اینجا](https://docs.gitlab.com/ee/ci/examples/) ولی من خودم از کدهای زیر برای تنظیم این فایل استفاده کردم.

در واقع میام برای هر پوش یه گیت clone و سپس npm install و ادامه که مشخصه.development اسم همون runner هست که مشخص کردیم و when یا مقدار always میاد روی هر git push این script رو انجام می ده.

**مثال برای ساخت یک فایل .gitlab-ci.yml**

یک پروژه react داریم که قراره اون رو build و پابلیش کنیم مراحل رو می خوایم در دو stage مشخص کنیم

**gitlab-ci.yml.**

stages:  
– build  
– deploy  
before\_script:  
– git -C /docker/compose/pwa-react/source pull [repo git address]  
build:  
stage: build  
script:  
– docker build -t pwa-react /docker/compose/pwa-react/  
tags:  
– docker  
deploy:  
stage: deploy  
script:  
– cd /docker/compose/pwa-react ; docker-compose up -d  
– docker rmi $(docker images -q –filter “dangling=true”)  
tags:  
– docker

Dockerfile  در محل /docker/compose/pwa-react/

بقیش موند.