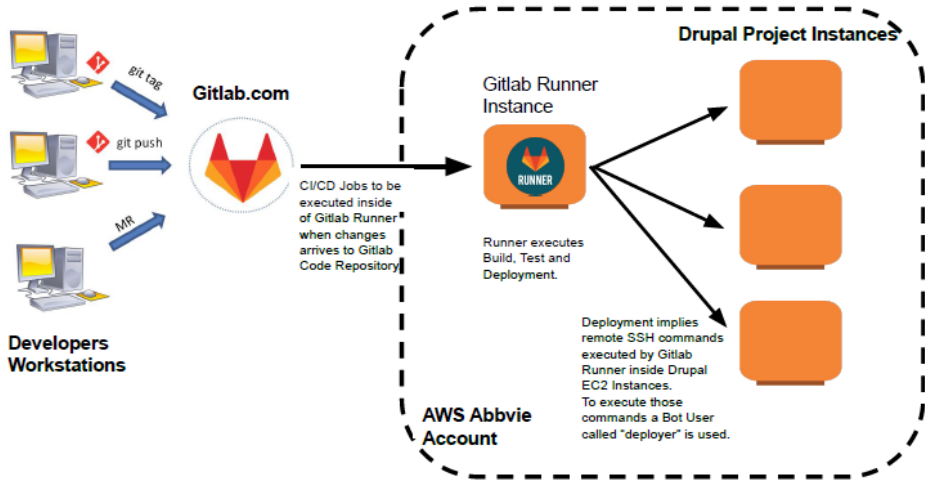


GitLab - Procurements from Academy Team

Important Emails

Attachment	Type	Content
<ul style="list-style-type: none"> PDF and the GitLab yaml Documentation link 	Email	 <p>The diagram illustrates the GitLab CI/CD workflow. On the left, 'Developers Workstations' are shown with actions: 'git tag', 'git push', and 'MR' (Merge Request). These actions trigger updates to the 'Gitlab.com' repository. A note states: 'CI/CD Jobs to be executed inside of Gitlab Runner when changes arrives to Gitlab Code Repository'. An arrow points from the repository to a 'Gitlab Runner Instance' (represented by a blue circle with a 'RUNNER' icon). A note next to the runner says: 'Runner executes Build, Test and Deployment.' This runner is connected to a dashed box labeled 'AWS Abbvie Account'. Inside this box, there are three orange squares representing 'Drupal Project Instances'. A note explains: 'Deployment implies remote SSH commands executed by Gitlab Runner inside Drupal EC2 Instances. To execute those commands a Bot User called "deployer" is used.'</p> <p>https://docs.gitlab.com/ee/ci/yaml/index.html</p>
<ul style="list-style-type: none"> Installed Software's/Tools on Runners and Docker 	Email	<p>Content added below this Table</p>
<ul style="list-style-type: none"> Tool information 	Email	<p>As I said in the previous email, all the commands that you can see in the .yaml file are launched in containers that are based on some Docker Image as a base which have those tools installed. The Dockerfiles that we use to create the custom images are in the folder: https://gitlab.com/abbviebea/bea/-/tree/development/docker for example: php command is launched using bea:latest image, so bea-base Dockerfile should show the version of php that we are installing/using:</p> <p>https://gitlab.com/abbviebea/bea/-/blob/development/docker/bea-base/Dockerfile#L5 This is php7.4</p> <p>Another example: cap command belongs to Capistrano tool, we are using in this job the ruby base image. Ruby Docker image comes with bundle tool preinstalled, so we launch "bundle install" and bundle will install the Capistrano Version specified in the Gemfile file. The content of Gemfile file is:</p> <pre>source 'https://rubygems.org' group :development do gem 'capistrano', '~> 3.5', '>= 3.5.0' gem 'capistrano-composer', '~> 0.0.6' end</pre>

<ul style="list-style-type: none"> • Why are Multiple Docker Containers being used? 	Email	<p>Having one image with all the required tools could be another possibility, but many times is easier to use a different base image that comes exactly with the tool that you need at this moment, for example in the job sonarqube-check, the official image from sonarqube comes with the tool that we need directly to be used, so we only need to use it. The same for the Deployment jobs, the base image ruby, comes with bundler preinstalled, so it is very easy to use it and launch "bundle install" to get the Capistrano tool that we need to launch "cap deploy" command.</p> <p>This is the main reason, but if a base image can be built with all the required tools, it is useful too.</p>
<ul style="list-style-type: none"> • GitLab CI-CD Explained .pdf 	PDF	
<ul style="list-style-type: none"> • A walk through by the BEA Team on Architecture and Yaml - Video 	Meeting Recording	

Installed tools/software's installed in GitLab Runners

GitLab Runner Base OS is Ubuntu 18.04, we are using an AWS AMI provided by Abbvie. In this instance we have installed:

- Docker: <https://docs.docker.com/engine/install/ubuntu/>
- GitLab Runner Ubuntu Package: <https://docs.gitlab.com/runner/install/linux-repository.html>

Once installed, we have configured GitLab-runner to run using Docker Executor and we have registered it against the Workspace launching a concrete command using a private token created by the [Gitlab.com](https://docs.gitlab.com/runner/register/#linux) Workspace Admin: <https://docs.gitlab.com/runner/register/#linux>

Once installed and registered, GitLab Runner is ready to pick up available CI jobs created on [Gitlab.com](https://gitlab.com). Every Job will be launched inside the runner, inside an Isolated container based on the Image you specify at the beginning of each job. This image should have the required tools that you need to launch successfully the commands specified in *before_script* and script parts.

If some job contains the tag "services" it means that this job will need some extra services to run. Every service will create a container that is linked with the main container of the job.

- What type of Images used and customizations performed

To take in mind: We are not using Docker in remote environments, we are only using Docker in the Pipeline, to execute tests, linters and execute deployments. The Application code is deployed in the instances more traditionally. Apache2, Php-fpm is directly installed in the server to serve the application.

As you can see in the gitlab-ci.yml, at the beginning of each job we are specifying the needed image for this concrete job:

Some of them are custom images, some others are used directly from public images from DockerHub:

The *Dockerfiles* for the custom images are located at <https://gitlab.com/abbviebea/bea/-/tree/development/docker>
Custom:

registry.gitlab.com/abbviebea/bea/bea --> Custom image with php, composer and node tools which contains the application code and preloaded application files for testing proposal.

registry.gitlab.com/abbviebea/bea/bd --> Custom image with a preloaded database for testing proposal.

registry.gitlab.com/abbviebea/bea/solr --> Custom image with a solr database preloaded for testing proposal.

registry.gitlab.com/abbviebea/bea/cypress --> Custom Image with concrete Cypress configuration for testing proposal.
Directly used from DockerHub:

sonarsource/sonar-scanner-cli:latest --> Image to launch sonar tool to test the code.

ruby:2.6.6-buster --> Image with ruby, to install Capistrano tool and use it in the deployment jobs.

docker:19.03.13 --> docker image which contains docker tool and allows us to launch docker commands, like build, pull and push.

memcached:1.5.6 --> memcached image which contains memcached tool.