Continuous Delivery with Containers

Elizabeth K. Joseph
@pleia2

All Things Open 2018
Spent the past 2 years working containers with Apache Mesos and Kubernetes

4 years working on CI/CD for OpenStack

10+ years in Linux systems administration and engineering roles

Author of *The Official Ubuntu Book* and *Common OpenStack Deployments*
Continuous Delivery (CD) is a software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time.

Via https://en.wikipedia.org/wiki/Continuous_delivery
Traditional Delivery

---

Months (or years!) between releases

Customers don’t see new features quickly

Developers lose track of features they worked on
Goal: A Modern Release Process with CD

---

Week 1: Project planning and release

Develop > Test > Stage > Release

Week 2: Customer Feedback and release

Develop > Test > Stage > Release

Week 3: Customer Feedback and release

Develop > Test > Stage > Release

...
The CI/CD Pipeline

git repository → CI/CD → staging → production

image

artifacts

git push

test
How?
Run in containers!

CC BY 2.0: https://www.flickr.com/photos/rubbermaid/6909787969/
Organize everything efficiently!
Can’t I just use VMs for testing?

Sure! But consider...

- VMs take longer to provision
- You may not need all they provide (kernel, system libraries...)
- An identical container image is simple to run in development, testing, staging and production
Bare Metal/VMs vs. Containers

---

**Traditional Datacenter**

- Siloed servers
- Low utilization
  - 12-15% for bare metal
  - 30% for virtual machines

**Containerization Platform**

- Integrated cluster (service discovery, authentication, etc)
- Workload multiplexing on the same machines
Does everything go in containers? Up to you.
Everything Running in Containers

Services & Containers
- GitLab
- Jenkins
- Microservice
- Artifactory
- Microservice
- Microservice
- Prometheus
- Microservice
- Microservice

Kubernetes, Docker Swarm, Apache Mesos
- Container Orchestration
- Security & Governance
- Monitoring & Operations
- User Interface & Command Line

Bare Metal, OpenStack, AWS, Azure, GCP
- Linux Server
- Linux Server
- Linux Server
- Linux Server
Everything Running in Containers

Traditional Datacenter

Jenkins-1

Jenkins-2

GitLab

Tests

Containerization Platform

Tests
Legacy Infrastructure + Containers

Continue running your legacy systems on Bare Metal or VMs

Send all tests to a new, independent containerization platform

GitLab
Legacy Infrastructure + hosted + Containers

Continue running your legacy systems on Bare Metal or VMs

Send all tests to a new, independent containerization platform
Fully hosted with GitHub

Hosted CI/CD system

CircleCI
Travis CI
TeamCity

Test on cloud-hosted Kubernetes service:

- Google Kubernetes Engine (GKE)
- Azure Kubernetes Service (AKS)
- Amazon Elastic Container Service for Kubernetes (Amazon EKS)
Fully hosted with GitLab.com

Use GitLab.com (hosted GitLab) for repository, artifact store, test runner

And connect it to a cloud-hosted Kubernetes service:

- Google Kubernetes Engine (GKE)
- Azure Kubernetes Service (AKS)
- Amazon Elastic Container Service for Kubernetes (Amazon EKS)
Or a combination of these!
Walkthrough:

Setting up a pipeline on GitLab with Auto DevOps and Kubernetes

Visit https://docs.gitlab.com/ee/topics/autodevops/quick_start_guide.html
Select a project template

Projects

New project
A project is where you house your files (repository), plan your work (issues), and publish your documentation (wiki), among other things.

All features are enabled for blank projects, from templates, or when importing, but you can disable them afterward in the project settings.

To only use CI/CD features for an external repository, choose CI/CD for external repo.

Tip: You can also create a project from the command line. Show command

<table>
<thead>
<tr>
<th>Blank project</th>
<th>Create from template</th>
<th>Import project</th>
<th>CI/CD for external repo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ruby on Rails</td>
<td></td>
<td></td>
<td>Use template, Preview</td>
</tr>
<tr>
<td>Spring</td>
<td></td>
<td></td>
<td>Use template, Preview</td>
</tr>
<tr>
<td>NodeJS Express</td>
<td></td>
<td></td>
<td>Use template, Preview</td>
</tr>
</tbody>
</table>

Ruby on Rails
Includes an MVC structure, Gemfile, Rakefile, along with many others, to help you get started.

Spring
Includes an MVC structure, mvnw and pom.xml to help you get started.

NodeJS Express
Includes an MVC structure to help you get started.
Set up project with template

New project
A project is where you house your files (repository), plan your work (issues), and publish your documentation (wiki), among other things.

All features are enabled for blank projects, from templates, or when importing, but you can disable them afterward in the project settings.

To only use CI/CD features for an external repository, choose CI/CD for external repo.

Tip: You can also create a project from the command line. Show command

Template
Ruby on Rails
Change template

Project name
rails-autodevops

Project URL
https://gitlab.com/pleia2
Project slug
rails-autodevops

Want to house several dependent projects under the same namespace? Create a group.

Project description (optional)
Description Format

Visibility Level
- Private
  Project access must be granted explicitly to each user.
- Internal
  The project can be accessed by any logged in user.
- Public
  The project can be accessed without any authentication.

Create project
Cancel
Enable Kubernetes in Google Cloud

Quickstart

This quickstart shows you how to deploy a containerized application with Google Kubernetes Engine.

Before you begin

Take the following steps to enable the Kubernetes Engine API:

2. Create or select a project.
3. Wait for the API and related services to be enabled. This can take several minutes.
4. Make sure that billing is enabled for your project.

LEARN HOW TO ENABLE BILLING
“Add Kubernetes Cluster”
Create new Cluster on GKE

Kubernetes cluster integration

With a Kubernetes cluster associated to this project, you can use review apps, deploy your applications, run your pipelines, and much more in an easy way.

Learn more about Kubernetes.

If you are setting up multiple clusters and are using Auto DevOps, read this first.

Enter the details for your Kubernetes cluster

Please make sure that your Google account meets the following requirements:

- Your account must have access to Google Kubernetes Engine
- Make sure your account meets the requirements to create Kubernetes clusters
- This account must have permissions to create a Kubernetes cluster in the Google Kubernetes Engine project specified below

Sign in with Google

or create a new Google account
Enter the details for your Kubernetes cluster

Please make sure that your Google account meets the following requirements:

- Your account must have access to Google Kubernetes Engine
- Make sure your account meets the requirements to create Kubernetes clusters
- This account must have permissions to create a Kubernetes cluster in the Google Kubernetes Engine project specified below

Read our help page on Kubernetes cluster integration.

Select a different Google account

Kubernetes cluster name
- autodevops

Environment scope
- *

Google Cloud Platform project
- My Project 71888
To use a new project, first create one on Google Cloud Platform.

Zone
- us-central1-a
Learn more about zones.

Number of nodes
- 3

Machine type
- n1-standard-2
Learn more about machine types and pricing.
The cluster exists!
Enable Applications

<table>
<thead>
<tr>
<th>Application</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Helm Tiller</td>
<td>Helm streamlines installing and managing Kubernetes applications. Tiller runs inside of your Kubernetes Cluster, and manages releases of your charts.</td>
</tr>
<tr>
<td>Ingress</td>
<td>Ingress gives you a way to route requests to services based on the request host or path, centralizing a number of services into a single entrypoint. Note: This will add some extra resources like a load balancer, which may incur additional costs depending on the hosting provider your Kubernetes cluster is installed on. If you are using Google Kubernetes Engine, you can check the pricing here. After installing ingress, you will need to point your wildcard DNS at the generated external IP address in order to view your app after it is deployed. More information</td>
</tr>
<tr>
<td>Prometheus</td>
<td>Prometheus is an open-source monitoring system with GitLab Integration to monitor deployed applications.</td>
</tr>
<tr>
<td>GitLab Runner</td>
<td>GitLab Runner connects to this project's repository and executes CI/CD jobs, pushing results back and deploying applications to production.</td>
</tr>
<tr>
<td>JupyterHub</td>
<td>JupyterHub, a multi-user Hub, spawns, manages, and proxies multiple instances of the single-user Jupyter notebook server. JupyterHub can be used to serve notebooks to a class of students, a corporate data science group, or a scientific research group.</td>
</tr>
</tbody>
</table>
Enable GitLab Auto DevOps

**General pipelines**
Customize your pipeline configuration, view your pipeline status and coverage report.

**Auto DevOps**
Auto DevOps will automatically build, test, and deploy your application based on a predefined Continuous Integration and Delivery configuration. Learn more about Auto DevOps

Auto Review Apps and Auto Deploy need a domain name to work correctly.

- **Default to Auto DevOps pipeline**
The Auto DevOps pipeline will run if no alternative CI configuration file is found. [More information]

**Domain**
35.194.19.243.nip.io
You need to specify a domain if you want to use Auto Review Apps and Auto Deploy stages. 35.194.19.243.nip.io can be used as an alternative to a custom domain.
Do not set up a domain here if you are setting up multiple Kubernetes clusters with Auto DevOps.

**Deployment strategy**
Deployment strategy needs a domain name to work correctly.
- Continuous deployment to production
- Continuous deployment to production using timed incremental rollout
- Automatic deployment to staging, manual deployment to production

Save changes
View pipelines
A pipeline!
What do these all mean?

In the **build** stage, the application is built into a Docker image and then uploaded to your project’s Container Registry (Auto Build).

In the **test** stage, GitLab runs various checks on the application:

- The **test** job runs unit and integration tests by detecting the language and framework (Auto Test)
- The **code_quality** job checks the code quality and is allowed to fail (Auto Code Quality)
- The **container_scanning** job checks the Docker container if it has any vulnerabilities and is allowed to fail (Auto Container Scanning)
- The **dependency_scanning** job checks if the application has any dependencies susceptible to vulnerabilities and is allowed to fail (Auto Dependency Scanning)
- The **sast** job runs static analysis on the current code to check for potential security issues and is allowed to fail (Auto SAST)
- The **license_management** job searches the application’s dependencies to determine each of their licenses and is allowed to fail (Auto License Management)

**Note:** As you might have noticed, all jobs except **test** are allowed to fail in the test stage.

The **production** stage is run after the tests and checks finish, and it automatically deploys the application in Kubernetes (Auto Deploy).

Lastly, in the **performance** stage, some performance tests will run on the deployed application (Auto Browser Performance Testing).

Via: https://docs.gitlab.com/ee/topics/autodevops/quick_start_guide.html#deploying-the-application
Navigate to deployed application

[Image of a GitLab interface with an arrow pointing to the deployed application environment]
Simple site in production!
Metrics! Via Prometheus in the GitLab UI
Make a change
Tests run again!
View console details of jobs running
<table>
<thead>
<tr>
<th>Status</th>
<th>Pipeline</th>
<th>Commit</th>
<th>Stages</th>
</tr>
</thead>
<tbody>
<tr>
<td>• passed</td>
<td>Latest</td>
<td>Update welcome_controller...</td>
<td>✔ ✔ ✔ ✔ ✔ ✔ ✔ ✔ ✔ ✔ ✔ ✔</td>
</tr>
<tr>
<td>#33703055</td>
<td>by Auto DevOps</td>
<td></td>
<td>00:15:43</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>19 seconds ago</td>
</tr>
</tbody>
</table>

Greetings, fellow All Things Open attendees! You're on Rails!
Advanced CD Strategies
Advanced Strategies: Canary Deployments

“Canary release is a technique to reduce the risk of introducing a new software version in production by slowly rolling out the change to a small subset of users before rolling it out to the entire infrastructure and making it available to everybody.”

https://martinfowler.com/bliki/CanaryRelease.html
Advanced Strategies: Blue/Green Deployments

“One of the challenges with automating deployment is the cut-over itself, taking software from the final stage of testing to live production. You usually need to do this quickly in order to minimize downtime. The blue-green deployment approach does this by ensuring you have two production environments, as identical as possible. At any time one of them, let's say blue for the example, is live. As you prepare a new release of your software you do your final stage of testing in the green environment. Once the software is working in the green environment, you switch the router so that all incoming requests go to the green environment – the blue one is now idle.”

https://martinfowler.com/bliki/BlueGreenDeployment.html
Advanced tooling exists!

---

Many containerization platforms allow for enough deployment strategies to define a canary or blue/green strategy yourself.

Tooling like Vamp can also help.

https://vamp.io/documentation/installation/v1.0.0/kubernetes/

https://vamp.io/documentation/installation/v1.0.0/dcos/
Questions?

Elizabeth K. Joseph

@pleia2

lyz@princessleia.com

https://princessleia.com/