Directory structure for project with Dockerfile, Jekinsfile, Kubernetes deployment yaml, pip requirements.txt, and test scripts?

Asked 4 years, 11 months ago Modified 4 years, 10 months ago Viewed 3k times

2

2

Would the following directory structure work?

The goal is to have Jenkins trigger off GitHub commits and run Multi-branch Pipelines that build and test containers. (I have everything running on Kubernetes, including Jenkins)

```
/project
  .git
  README.md
  Jenkinsfile
  /docker_image_1
    Dockerfile
    app1.py
    requirements.txt
    /unit_tests
      unit_test1.py
      unit_test2.py
  /docker_image_2
    Dockerfile
    app2.py
    requirements.txt
    /unit_tests
      unit_test1.py
      unit_test2.py
  /k8s
    /dev
      deployment.yaml
    /production
      deployment.yaml
  /component_tests
    component_tests.py
```

1. Is the k8s folder that has the deployment.yamls in the right place?
2. Are the test folders in good locations? The tests in "component_tests" will ideally be doing more end-to-end integrated testing that involve multiple containers
3. I see a lot of repos have Jenkins file and Dockerfile in the same directory level. What are the pros and cons of that?

docker jenkins pip kubernetes jenkins-pipeline

ShareImprove this questionFollow

edited Jul 26, 2017 at 18:12

asked Jul 26, 2017 at 17:13
gunit
3,262 2 26 41

2 Answers

Sorted by:
Trending sort available
Highest score (default)

5

There's no good answer to this question currently.

Ku eployment, but as a technology it relies on additional 3rd party tooling manage the build part of the ALM workflow. There a lots of options available for turning your source code into a container running on Kubernetes. Each has it's own

consequences for how your source code is organised and how a deployment might be invoked from a CI/CD server like Jenkins.

I provide the following collection of options for your consideration, roughly categorized. Represents my current evaluation list.

"Platform as a service" tools

Tooling the manages the entire ALM lifecycle of your code. Powerful but more complex and opinionated.

- Deis workflow
- Openshift
- Fabric8 (See also Openshift.io)

Build and deploy tools

Tools useful for the code/test/code/retest workflow common during development. Can also be invoked from Jenkins to abstract your build process.

- Draft
- Forge
- Kcompose
- Fabric8 Maven plugin (Java)
- Psykube

YAML templating tools

The kubernetes YAML was never designed to be used by human beings. Several initatives to make this process simpler and more standardized.

- Helm
- Ksonnet

Deployment monitoring tools

These tools have conventions where they expect to find Kubernetes manifest files (or helm charts) located in your source code repository.

- Keel
- Kube-applier
- Kubediff
- Landscaper
- Kit

CI/CD tools with k8s support

- Spinnaker
- Gitlab
- Jenkins + Kubernetes CI plugin
- Jenkins + Kubernetes plugin

ShareImprove this answerFollow

edited Sep 7, 2017 at 9:03

answered Jul 26, 2017 at 18:24

Mark O'Connor
74.6k 10 133 178

4

This is really left much to your preference. In our projects we tend to split services into separate repositories not subfolders, but we also had a

This is really left much to your preference. In our projects we tend to split services into separate repositories not subfolders, but we also had a case where we had a bunch of Scala microserviced managed in similar way (although dockers were built with sbt plugin for docker)

One big advice I would give you is that in the long run managing your kubernetes manifests like that might become serious pain in the back. I went through this, and my suggestion is to use `helm` charts from day one.

I assume that your "component_tests" are end-to-end tests. Other then naming I see no problem with that. For cases where we test solutions that span multiple repos we keep them in a separate repo as well though.

Share Improve this answer Follow

answered Jul 26, 2017 at 17:59

Radek 'Goblin' Pieczonka

19.4k 5 46 45

Definitely split services into separate repositories when possible. Then you put the deployment stuff in a separate repos as well that uses the built versions of the images from your image repository. It's so much cleaner. – Grimmy Jul 26, 2017 at 18:14

... at least when you work with more than a couple of people. I also don't want to see commits about configuration files for live mixed in with the apps. – Grimmy Jul 26, 2017 at 18:20

Splitting services up in the separate repositories is a good idea and inline with the concept of microservices and the 12 Factor app recommendations. This is where Helm is a good candidate for future success with the emergence of repositories for hosting helm charts. Should make the deployment of complex eco-systems of micro services more palatible – Mark O'Connor Jul 26, 2017 at 18:37