Developing a Software Management and Testing Platform-as-a-Service for Research Software

THE UNIVERSITY of EDINBURGH

Motivation

There is a growing requirement to incorporate sustainable software practices into High Energy Physics experiments. Widely supported tools offering **source code management**, **continuous integration (CI)**, **unit testing** and **software quality assurance** can greatly help improve standards. However, for resource-limited projects there is an understandable inertia in deviating effort to cover systems maintenance and application support for additional tools. Although basic *free* code hosting options are widely available, *premium* support and hosting costs may become prohibitive as projects increase their software testing footprint.

Here we describe the development of a **Platform-as-a-Service (PaaS) solution** that combines software management and testing applications into a consolidated service. This solution provides an easy to deploy sandbox environment for projects wishing to evaluate the functionality of a full code management and testing suite without needing an initial outlay for an on-premise or hosted solution. The approach also provides a pathfinder exercise to create a federated network of software projects that can take advantage of cost-effective resource pooling and facilitate the reuse of common software testing patterns.

Prototype Development

A functioning prototype was built to demonstrate how a PaaS solution could meet the desired objectives. All service components were hosted in the **Microsoft Azure** cloud. **Gitlab** was chosen as a candidate software management platform based the availability of a native CI pipeline and the relative ease of deployment. **Kubernetes** (and by extension the **Azure Kubernetes Service**) provided the orchestration of **Gitlab CI Runner** containers executed as part of the CI testing workflow (Figure 2).

A schematic of the prototype and the proposed development lifecycle is shown in Figure 1. In a cloud computing environment seemingly complex deployment requests can be performed in relatively few steps. Figure 3 demonstrates how a new kubernetes cluster deployment can be performed on the command line [1]. A scale testing suite was developed to simulate code submissions and review actions sent to a running platform instance (Figure 4). This enabled scaling response, load balancing and resource accounting to be accurately evaluated.

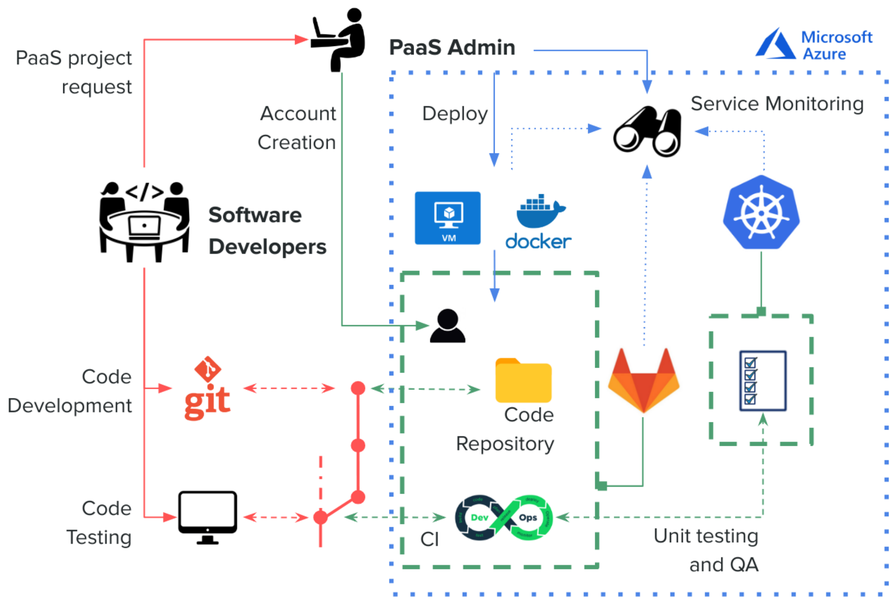

Figure 1 Deployment and development lifecycle of a PaaS Research Software instance

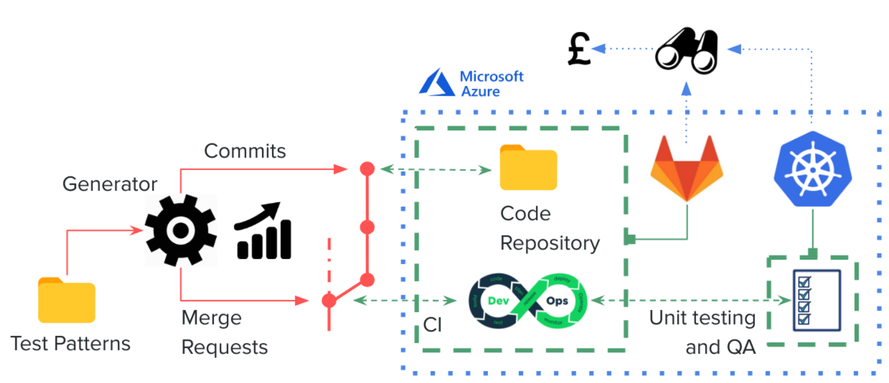

Figure 4 Scale testing suite to monitor PaaS performance

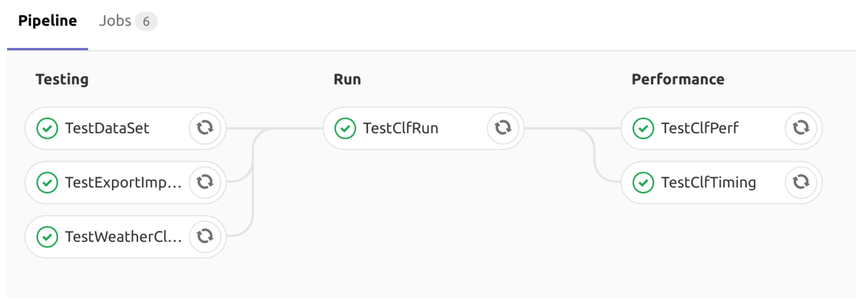

Figure 2 Representative Gitlab CI pipeline for code in the prototype

```
az aks create \
    --name myAKSCluster \
    --resource-group myResourceGroup \
    --node-count 1 \
    --generate-ssh-keys \
    --service-principal <appId> \
    --client-secret <password>
```

Figure 3 Azure CLI command to create a new kubernetes cluster

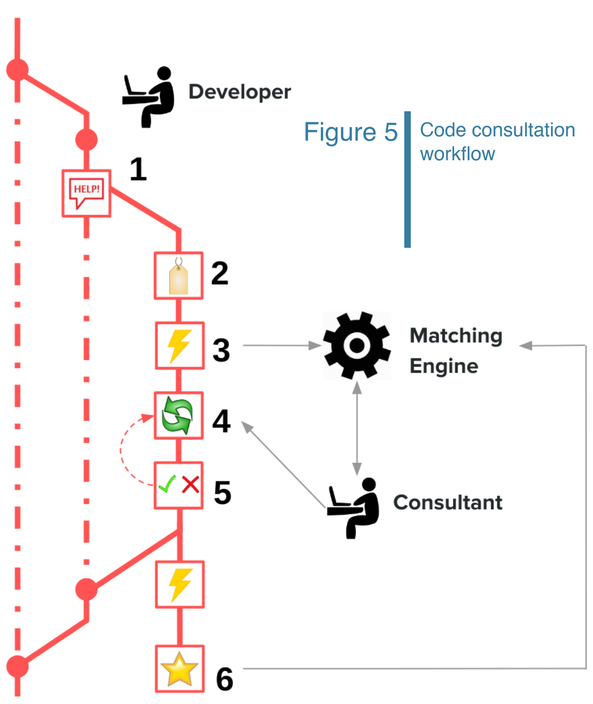

Figure 5 Code consultation workflow

Cross-Project Collaboration

A multi-tenancy platform could be used to encourage cross-project collaboration on software development, code review, and the sharing of common testing patterns. A *naive* **consultation framework** was created to determine how a collaborative approach could be devised in practice by leveraging the chosen technologies chosen in our PaaS prototype (Figure 5). This approach has been inspired by the code review process adopted by the ATLAS experiment [2].

To engage with a consultant a developer would first create a new fork from their active code branch (**1**) and initiate a merge request discussion using **labels** to designate the scope of their request (**2**). This triggers an action to poll a matching engine which assigns the request to a suitable consultant from an available pool (**3**). The assigned consultant then provides effort to satisfy the request (**4**). Once the outcome is agreed the developer merges changes back into their branch (**5**). Feedback on the consultation is then solicited from the developer triggered by the closure of the merge request (**6**). Effort could be assessed through evaluation metrics ranging from a simple *karma* increment to a more involved template to qualify items such as impact, quality and time to delivery.

Outlook

The success of a proposed research software PaaS relies upon agile deployment, accurate resource accounting and seamless service orchestration with infrastructure implementation details hidden from the end developers. A working prototype is in place and early adopters based in the UK Research Software community will be encouraged to navigate the platform and provide feedback on how the service can be further developed. At the end of the evaluation period the deployment and user experiences will be reviewed in anticipation of the development a more robust and community supported platform.

Andrew Washbrook[1]
[1]University of Edinburgh

[1] Deploy an Azure Kubernetes Service (AKS) cluster
https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-deploy-cluster

[2] Modernising ATLAS Software Build Infrastructure
https://indico.cern.ch/event/567550/contributions/2627121/

CHEP 2018
Sofia, Bulgaria