PaaS for Web apps as key to the evolution of CERN Web Services

Alexandre Lossent, Alberto Rodriguez Peon, Andreas Wagner, Nikos Kasioumis (CERN)

Main Goals

Modernize web central hosting at CERN
- Support modern development frameworks
- Provide more flexibility for users
- Reduce/Eliminate the need for "locally managed" web servers

Improve the offering of tools to Web developers
- Make it easier for developers to get started
- Allow automation of application deployment
- Integration with CERN code hosting (GitLab)
- Automation with CI/CD pipelines (GitLab/Jenkins)

Facilitate deployment & operation of web applications
- Allows fast prototyping
- Facilitates hosting of central services
- Self-service templates for application instances
- Application managers don't need to maintain OS

CERN Web Frameworks strategy

Use containers to host everything and to widen the service options. Use OpenShift to orchestrate all.

CERN Integration

Persistent Storage
- EOS, CVMFS, CephFS, NFS
- Dynamic Volume Provisioning

Authentication & Authorisation
- SSO
- E-groups
- Kerberos Support

Central Monitoring
- Using CERN Central Monitoring solution based on Hadoop and Elasticsearch

Code hosting and CI/CD pipelines
- Provided by Gitlab

CERN Web Services:
- Name allocation, management of ownership of project and lifecycle, quota management

Custom CERN build of OpenShift
- Allows deployment of relevant patches while OpenShift Origin provides few point releases
- Minor customisations to support CERN storage with flexvolume (until CSI storage driver is available in OpenShift 3.10)

Network visibility of hosted applications
- Intranet, Internet and CERN technical network

All integration components are implemented in form of Kubernetes out-of-tree custom controllers in Go using the Go-client libraries for OpenShift and Kubernetes.

Operations

Automation
- Ansible Playbooks allow to reuse all primitives available from OpenShift-Ansible

Updates
- Grace-period for all application for OpenShift update deployment enables application owners to perform a graceful redeployment of their applications

Service and Usage Statistics

Production cluster
- 300 projects
- 1000 pods

Development cluster
- 150 projects
- 500 pods

Infrastructure (July 2018)
- Worker node VMs: 30GB RAM, 10 CPU cores
- Production
  - 25worker nodes
  - 5 dedicated Gitlab worker nodes
- Devevelopment
  - 17 worker nodes
  - 3 dedicated Gitlab worker nodes

Service Usage
- 10 million requests per day (without Gitlab)
- 12 million requests per day for Gitlab
- Daily traffic 1TB (about 0.5TB for GitLab and 0.5TB for other apps)

GitLab use case: increased operational efficiency

Technologies

Container orchestration with OpenShift
- Focus on application itself, delegate infrastructure management
- Portable deployments (commercial cloud...)

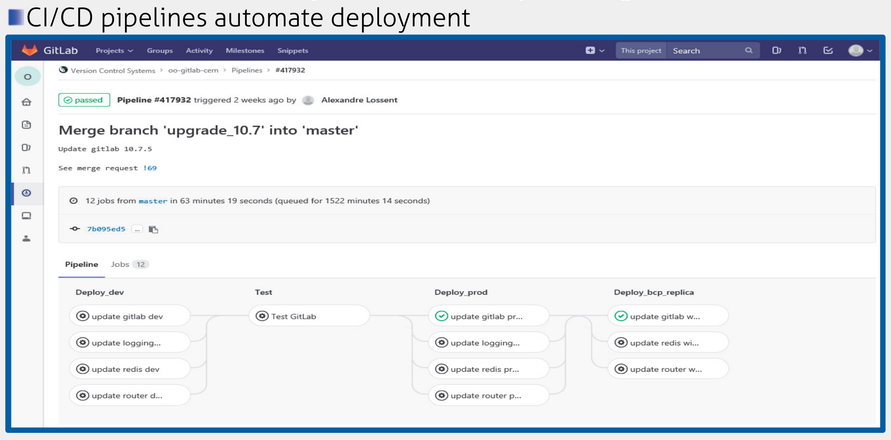
CI/CD pipelines
- Purpose
  - Fully automate deployment, upgrades, configuration changes
  - Automated integration tests
- Properties
  - Application deployment fully described as code, under version control
  - Keeps track of what has been deployed, where and when; supports rollback, redeploy, retries

GitLab in OpenShift

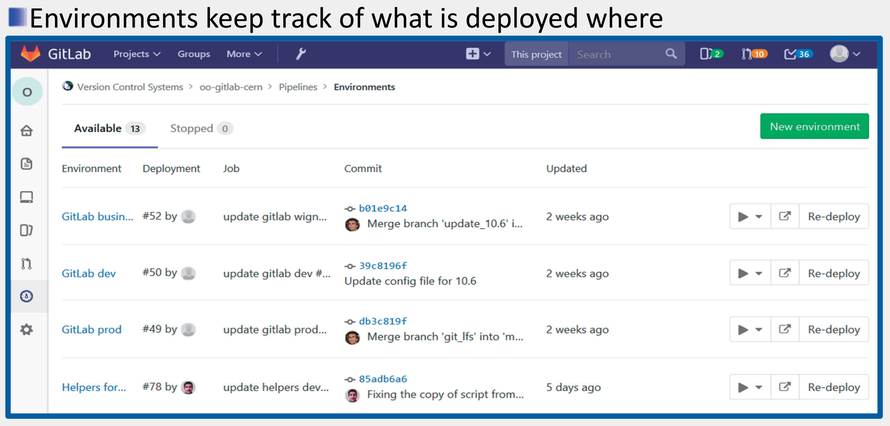
- Fully automate deployment, upgrades, configuration changes
- Improved service uptime
- Reduced time to recovery in case of incidents
- Considerably reduced operational effort
- GitLab deploys and updates itself via CI/CD pipelines

CI/CD example - GitLab updating itself

- CI/CD pipelines automate deployment



- Environments keep track of what is deployed where



Leverage Helm for app deployment

- "package manager for Kubernetes" (and Openshift)
- Describes an application and its multiple instances (dev, staging, prod...) as code (YAML)
- Server-side Tiller component handles deployments and updates of each instance
- Helm "charts" readily available for lots of apps
- Native support in GitLab CI/CD ("auto devops")

CI/CD pipelines benefits

- Automated integration tests: confidence not breaking most important workflows
- Deploy changes faster, more often, with less effort
- Simple procedure, can be followed by non-experts

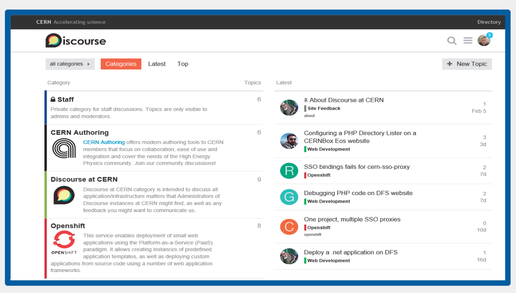
Automation of operational tasks

- Move from detailed manual procedures and ad-hoc scripts to automated Ansible playbooks
- Package playbooks, the entire management toolkit and dependencies in a single Docker image that can be used by all operators, anywhere
- Leverage Prometheus and AlertManager for automated responses
- Changes in operational tasks can be tracked and reviewed like code changes

Other Use Cases

Application templates
- Enables multiple instances of an application
- Self-service creation from a centrally maintained template
- Moving from Openshift templates to more powerful Ansible Playbook Bundles (APBs)

New use cases
- Discourse Community Forum



- Sentry Error Tracking

Future plans

Generic web site hosting:
- Serve static & CGI Web content from shared EOS file system

Drupal
- Future Drupal Service based on containers

https://home.cern