Continuous Integration for Research Software

Dr Christopher Cave-Ayland

c.cave-ayland@imperial.ac.uk
@ImperialRSE
Imperial College Research Computing Service, DOI: 10.14469/hpc/2232
What is Continuous Integration

- Continuous integration (CI) is the practice of automating the integration of code changes from multiple contributors into a single software project – Atlassian

https://cloud.google.com/solutions/continuous-integration/
Lots of options!

- Azure Pipelines
- circleci
- Bamboo
- ANVIL
- Google Cloud
- GitHub Actions
- Bitbucket Pipelines
- Jenkins
More usefully

Cloud hosted services (usually including compute environments)

Software

- ANVIL
- Jenkins
- Buildbot
- Azure Pipelines
- GitHub Actions
- CI/CD
- CircleCI
- Bamboo

https://www.imperial.ac.uk/ict/rcs
Challenges for Research Software and CI

- Computationally intensive – cpu/memory
- Use of accelerators
- Complex dependencies
- Multi-platform
- Specialist compilers + operating systems
- Multi-node execution

How do these interact with available CI implementations?
Nektar++ - www.nektar.info

- Finite element/computational fluid dynamics code
- ~15 years old
- Open-source C++
- 2 full time developers – Imperial + Exeter
- Variable number of PhD/project student developers

- Computationally intensive (compile + test)
- Multi-platform
- Complex dependencies
Existing Nektar++ CI Setup
Criteria

- Reduced maintenance burden
- Work with on-premise GitLab code repository
- Greater reproducibility
- Test on Windows, Mac and 6 Linux distros
- Optimised build times (build cache)
- Rapid debugging of failures
- Infrastructure-as-code
- Easy to setup new environments
- No recurrent costs – preferably will make use of existing infrastructure
Review some alternatives

- ANVIL
  Research Software Testing Platform
- Buildbot (with tweaks)
- Azure Pipelines
- CI/CD
ANVIL
Research Software Testing Platform

- Specialised CI service for research software
- STFC hosted (project restrictions)
- Based on Jenkins
- Can run workloads on SCARF (HPC cluster)
- Scientific software + compilers available in environment
  - Intel compilers
Front-end vs Back-end

Cloud

On-premise

GitLab

Front-end

Azure Pipelines

Back-end

Microsoft-hosted agents

Self-hosted agents
Back-end alternatives

On-premise  "Cloud"

On-premise alternatives:
- VirtualBox
- Docker

"Cloud" alternatives:
- ANVIL
- Azure Pipelines
- CI/CD
### Scores

#### Front-End

<table>
<thead>
<tr>
<th></th>
<th>Heterogeneous work loads</th>
<th>Gitlab Integration</th>
<th>Sustainability</th>
<th>Ease of Use</th>
<th>IaC</th>
<th>Advanced Gitlab Integration</th>
<th>CD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buildbot</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Gitlab CI</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Azure Devops</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Anvil</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

#### Back-End

<table>
<thead>
<tr>
<th></th>
<th>Multi-platform</th>
<th>Sustainability</th>
<th>Ease of Use</th>
<th>Infrastructure</th>
<th>Cost</th>
<th>IaC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gitlab CI</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>On-prem VMs</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>On-prem Docker</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Gitlab CI</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Azure Devops</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Anvil</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

#### Front-End

<table>
<thead>
<tr>
<th>Front-End</th>
<th>Back-End</th>
<th>Total Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gitlab CI</td>
<td>On-premise Docker</td>
<td>26</td>
</tr>
<tr>
<td>Buildbot</td>
<td>On-premise Docker</td>
<td>24</td>
</tr>
<tr>
<td>Azure</td>
<td>Azure</td>
<td>21</td>
</tr>
<tr>
<td>Anvil</td>
<td>Anvil</td>
<td>20</td>
</tr>
</tbody>
</table>
Beyond the scores

Azure Pipelines + Microsoft agents
- A good offering
- Every platform
- 10 concurrent free builds
- Lowest maintenance
- Held back by GitLab integration
- Unclear what cost would be

Buildbot + On-premise Docker
- Swapping VMs for Docker is a no-brainer
- CI configuration is separate from code base
- Separate server to maintain
- Support for building rpms/debs
- Custom integration with GitLab

GitLab CI + On-premise Docker
- Integrated with code hosting
- GitLab.com runners would be expensive
- Container registry
- Conditional pipeline execution

Anvil + Anvil
- No container support
- Specialised environments not relevant to Nektar++
- No relevant dependencies available
- Questionable longevity
Our work-in-progress solution

Gitlab Server

- Docker Registry
- CI Controller

Execution hosts

- Windows VM
  - gitlab-runner
- Docker daemon
  - gitlab-runner
  - Local build caches

MacOS Host

- gitlab-runner

Developer

Pull failed builds

Environment images
Failed build images
Job workloads

Job workloads
The benefits

- Reduced maintenance – 12 VMs down to 1
- CI configuration is under version control
- Non-admins can change the CI configuration
- Non-admins have access to rapid debugging workflow
- Linux builds are now fully reproducible
- Adding new Linux distros is easy
- Much more agnostic to execution host
- Faster and more flexible execution
- All in part of GitLab
Insights

• One size does not fit all
  – Individual project requirements
  – Existing constraints
• Not much to choose between different CI workflow languages – you’re going to write a yml file
• Use Docker
• Don’t underestimate time required to maintain infrastructure
• Existing cloud CI services still don’t meet all use cases for research software
Cloud based Possibilities

Developer

Auto scaling Kubernetes

Gitlab Server

Docker Registry

CI/CD Controller

Pull failed builds

Azure Pipelines

Environment images

Failed build images

Job workloads

Job workloads
Thank you!

- Nektar++ development team
  - Chris Cantwell
  - Dave Moxey
  - Spencer Sherwin

- Research Software Reactor
  - Tania Allard
  - Sarah Gibson
  - Gerard Gorman
  - Microsoft

- Research Computing Service
  - Diego Alonso Alvarez
  - Mayeul d’Avezac de Castera
  - Mark Woodbridge

Questions?