

workload-sharing cluster

**Onderzoek**

**Klas: HA5C**

**Groepsleden:** Bastiaan

Links:

<https://github.com/medranSolus/fractal_service>

<https://blender.dtmc.ca/docs/>

<https://kubernetes.io/>

<https://flamenco.blender.org/>

**General Data dump :**

**Blender render farm:**

The [Blender](https://blender.dtmc.ca/docs/blender.org) Render Cluster is a [Kubernetes](https://kubernetes.io/) example render cluster that can use standard Desktop PCs to run in a render farm with the help of scheduling project called [Flamenco](https://flamenco.blender.org/).

The overall project goal is to be able to easily add any type of compute nodes to a Flamenco rendering cluster using DevOps best-practises.

Check out this short [introduction video](https://youtu.be/9covNqEJd9Y) or guide on [medium.com](https://medium.com/@michael.cook.talk/farming-with-flamenco-part-one-d53ca7b682c1).

Context

There have been four typical ways of rendering Blender projects:

1. On your workstation with a nice graphics card.
2. Custom job scripts that dispatch Blender projects to adhoc group of workstations.
3. A group of workstations using [Flamenco](https://flamenco.blender.org/) using shared storage.
4. Online commercial render farms with Pay-As-You-Go compute resources, paywalls etc.

This project is a more standardised way of doing (2) and a more complicated version of (3). The render farm runs [Flamenco](https://flamenco.blender.org/) Workers on a [Kubernetes](https://kubernetes.io/) cluster simplifying the maintenance, monitoring and management of the underlying hardware. Even if you have a powerful workstation, this kind of edge-cluster may be a welcome addition and you can still use your workstation to render projects based on how it is configured with Flamenco.

Learn More

Here is an unofficial brief history of Blender rendering landscape from an opensource perspective.

Scripts

Circa ~2012-2022 there have been several attempts to create scripts that send parts of Blender project to more than one rendering workstation. These were usually fire-and-forget or remote-desktop type approaches using NAS.

Then there were scripts and clients built around the inbuilt Blender ‘network renderer’ feature. These were similar to [Flamenco](https://flamenco.blender.org/), where they replicate the .blend file for a Project across workstations running agents or clients. Then their render output sent to a NAS. This worked for Blender 2.x pre 2020.

The inbuilt ‘network renderer’ feature doesn’t seem to be available in Blender v3+ and those opensource projects all seem to have been stopped in their tracks.

Flamenco v3

[Flamenco](https://flamenco.blender.org/) v3 (circa 2021+) is an opensource solution which is built by Blender Foundation and not only distributes a Blender project to be rendered but also adds some intelligence around Job priority, retries, and logging under one manager with a web-console and a Flamenco Blender Add-On (plugin) on each workstation that wishes to submit Flamenco Jobs.

The other main advantage of [Flamenco](https://flamenco.blender.org/) v3 is its simplicity and “just works out of the box” goal across multiple platforms for end-users with a few computers at home or a small-scale team.

Learn more about common deployment scenarios using Flamenco:

* [Just Flamenco](https://blender.dtmc.ca/docs/documentation/ways_of_using/flamenco_only/) - How [Flamenco](https://flamenco.blender.org/) works for a small team.
* [Flamenco Worker Tags](https://blender.dtmc.ca/docs/documentation/ways_of_using/flamenco_fwc.md) - Using Flamenco Worker Tags (FWTs) for Development vs Production compute resources.
* [Flamenco in an Edge Cluster](https://blender.dtmc.ca/docs/documentation/ways_of_using/flamenco_cluster/) - Using Kubernetes to build a blue-green pipeline with zero interruptions.
* [Best of Both Worlds](https://blender.dtmc.ca/docs/documentation/ways_of_using/flamenco_bobw/) - Mixing Workstations and Cluster resources.

Cloud Rendering?

Hang on… why isn’t this project about a cloud-based cluster? - [some thoughts](https://blender.dtmc.ca/docs/documentation/ways_of_using/flamenco_cloud/)

There are also numerous cloud-rendering solutions springing up each with their own business model. This project is not using cloud-rendering solutions. This project uses Flamenco in a scalable edge-cluster situated in a small office or home environment.

Disclosure

This project is not affiliated with [Blender](https://blender.dtmc.ca/docs/blender.org), [Flamenco](https://flamenco.blender.org/) or [Kubernetes](https://kubernetes.io/).

**Kubernetes:**

[Kubernetes](https://kubernetes.io/docs/concepts/overview/), also known as K8s, is an open source system for automating deployment, scaling, and management of containerized applications.

It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon [15 years of experience of running production workloads at Google](http://queue.acm.org/detail.cfm?id=2898444), combined with best-of-breed ideas and practices from the community.

Planet Scale

Designed on the same principles that allow Google to run billions of containers a week, Kubernetes can scale without increasing your operations team.

Never Outgrow

Whether testing locally or running a global enterprise, Kubernetes flexibility grows with you to deliver your applications consistently and easily no matter how complex your need is.

Run K8s Anywhere

Kubernetes is open source giving you the freedom to take advantage of on-premises, hybrid, or public cloud infrastructure, letting you effortlessly move workloads to where it matters to you.

To download Kubernetes, visit the [download](https://kubernetes.io/releases/download/) section.

**Flamenco:**

Features overview

Free and Open Source

Released under GPL 3.0, every component of Flamenco is Free and Open Source software. Development is supported by the Blender project.

Simple and Portable

Flamenco consists of a few components and requires almost no configuration in order to be used in production.

Easy Customization

Designed to be customizable, Flamenco allows TDs to specify Job Types are specified using the JavaScript language and seamlessly fit into a pipeline.

Cross-platform and Self-hosted

Flamenco runs on all major operating systems, and is self hostable. Your data is yours, and yours alone.

Robust Technology

The core of Flamenco is built using Go and SQLite. Check out the source code on [projects.blender.org](https://projects.blender.org/studio/flamenco).

In Development

Flamenco v3 is in active development at Blender Studio. Join [the chat](https://chat.blender.org/#/room/#flamenco:blender.org) to see what’s happening!

**Cannonial:**

Cannonial “The Ubuntu people” staat bekend als de maker van Ubuntu, maar werkt ook vooral in de enterprise kant van server clustering en Cloud-beheer. Maar heeft ook kleinschalinge oplossingen

**Ubuntu server:**

Ubuntu server is een headless (zonder GUI: een grafical user interface. Dit betekend dat er alleen een terminal is) en lichtere versie van Ubuntu desktop. Ubuntu is een distributie van Linux wat een open-source OS is.

**Cannonial kubernetes:**

**Charmed kubernetes:**



Composable, operator-based Kubernetes for the enterprise

Afbeelding met cirkel, schermopname, Graphics, logo

Automatisch gegenereerde beschrijving

* Model-driven Kubernetes for fully customisable deployments
* Pluggable CNI, CSI, CRI and monitoring components
* Carrier-grade and hardware acceleration features
* Support for third-party components and services
* Full life cycle management for host and in-cluster with [Juju](https://juju.is/)
* [Install Charmed K8 ›](https://ubuntu.com/kubernetes/install#multi-node)
* [Learn more about Charmed K8s ›](https://ubuntu.com/kubernetes/features)

**MicroK8s:**

**Juju:**

Shift from configuration management to distributed system management

Afbeelding met schermopname, cirkel

Automatisch gegenereerde beschrijving

Applications never run in isolation. Adding a workload to production requires adding essential services like observability, patching, identity, secret management, and many more

But it doesn’t stop there. 1000 days from now this complex system needs to work just as well as it does today. This means that lifecycle events like upgrades, migrations or scaling have to work seamlessly without downtime.

Without a comprehensive understanding of how to deploy, integrate and operate software, there is often too much perceived risk to adopt it in large scale, production settings. This is especially true for open source software, where operational knowledge is concentrated in a small number of hard-to-recruit individuals.

It’s time to open source the operations code

To unlock the true potential of open source, we must normalise open sourcing not just the application code, but also the operational frameworks required to take the most out of it

That’s where Juju and charms come in.

Charms take all the domain knowledge required to operate software effectively and distil it into clean, maintainable, testable Python code that can be used across clouds.

Juju is the orchestrator that helps to deploy, manage and integrate that software across Kubernetes, VMs, bare metal and public clouds.

What are Juju and Charms?

Juju, the orchestrator engine

Juju is an open source orchestration engine for software operators that enables the deployment, integration and lifecycle management of applications at any scale, on any infrastructure.

Charms, the software operators

A charm is an operator: business logic encapsulated in reusable software packages that automate every aspect of an application’s life.

Charms are developed with the Charm SDK which comprises the:

* Ops library: a Python framework for developing and testing charms
* Charmcraft: a tool for building, packaging and publishing charms.

Afbeelding met tekst, schermopname, cirkel, logo

Automatisch gegenereerde beschrijving

What makes juju and charms different?

Afbeelding met tekst, schermopname, Graphics, cirkel

Automatisch gegenereerde beschrijving

Automate operations across cloud, metal, VMs and Kubernetes

Charms allow you to use a single codebase to automate the lifecycle management of your system across public clouds, Kubernetes, virtual (VM) and bare metal machines.

Afbeelding met schermopname, cirkel, tekst, Graphics

Automatisch gegenereerde beschrijving

Seamless integrations across different clouds

Integrations simplify the process of connecting two applications by automatically applying required networking and configuration changes, even if the workloads are deployed across different clouds (e.g. Openstack and AWS).

Afbeelding met schermopname, tekst, Graphics, diagram

Automatisch gegenereerde beschrijving

Lifecycle management for your system, not individual applications

Because applications integrations are a first class citizen in Juju, the system knows how to optimise the lifecycle management of the entire system, rather than individual applications.

How does Juju compare to similar tools

Juju’s focus is on system lifecycle management

Most existing tools are focused on solving the problem of deployment or configuration management, while juju’s main focus is on Day 2 operations like backup, upgrades, migrations, or scaling.

Juju integrates well with Terraform, extending its infrastructure as code capabilities to our application modelling framework.

Afbeelding met tekst, schermopname, software, Multimediasoftware

Automatisch gegenereerde beschrijving