CIQ

Rocky, Apptainer, Warewulf, and Fuzzball / HPC-2.0

OpenFabrics Alliance

Innovations in HPC Software Infrastructure

CIQ IS...

Global Hybrid Optimized Software Infrastructure

Infrastructure-2.0

ROCKY LINUX

APPTAINER

WAREWULF

FUZZBALL

Empowering

Innovating software infrastructure solutions for extreme performance and scale, including service models designed to meet the needs of your organization.

Improving

Our passion is to always do better. Always provide more value. Always drive intelligent innovation. Always provide what the customer & community needs.

Growing

CIQ believes in constant and never-ending growth. We are growing fast and looking for like-minded people with an unwavering passion for always doing better for ourselves and others.

CIQ

Community Enterprise Linux

Rocky Linux tm

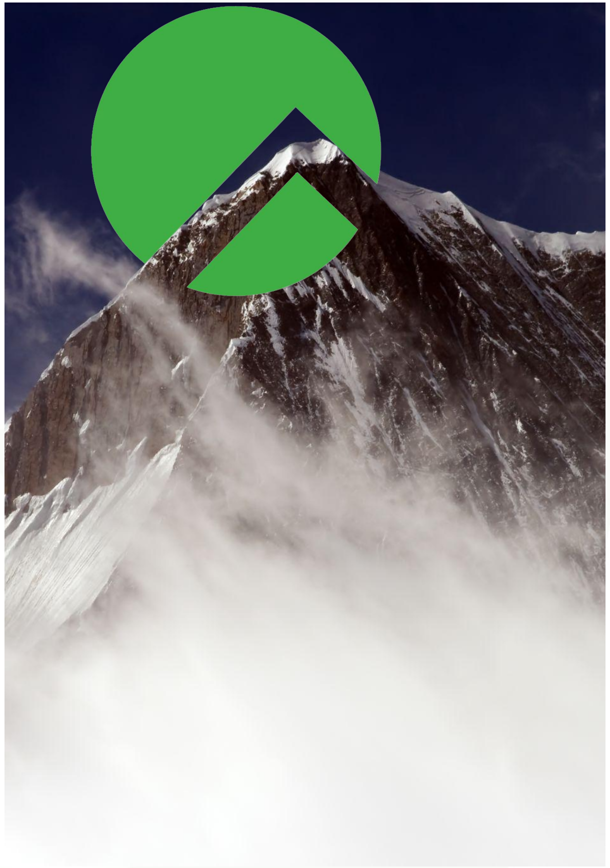
Rocky Linux

Community Enterprise Linux

As CentOS comes to its end of life, Rocky Linux is there as the natural successor. By design it is a freely available, community driven, bug-for-bug compatible version of Enterprise Linux.

Rocky Linux is quickly becoming the dominant operating system for the enterprise & HPC.

Google NAVER CLOUD PLATFORM vmware fastly Microsoft Azure Mattermost ORACLE CLOUD SUPERMICRO NVIDIA AMD SEAGATE GOVERNMENT SOLUTIONS arm PENGUIN COMPUTING A subsidiary of SMART Global Holdings, Inc. amazon web services GitLab



Rocky Linux

Features

- **Community Enterprise Linux:** The Rocky Enterprise Software Foundation is designed to be a community driven, controlled, and architected OS that will stand the test of time.
- **Compatibility:** Rocky Linux is 100% compatible with Enterprise Linux which makes it a simple, drop-in solution.
- **Stability:** A community of vested individuals, organizations and enterprises of every shape and size provides True Stability.
- **Every Cloud, Every OEM, Every ISV:** Our vision from the beginning is to have every cloud, every OEM, every component and every ISV vested, compatible and partnering for the good of everyone.
- **The Ethos of CIQ & The Name of Rocky:** CIQ is committed to empowering everyone to do what they do great. Rocky McGaugh, CentOS co-founder and namesake of Rocky Linux, was great at Linux and it is important that everyone know their contributions make a difference in this world.

Compatible

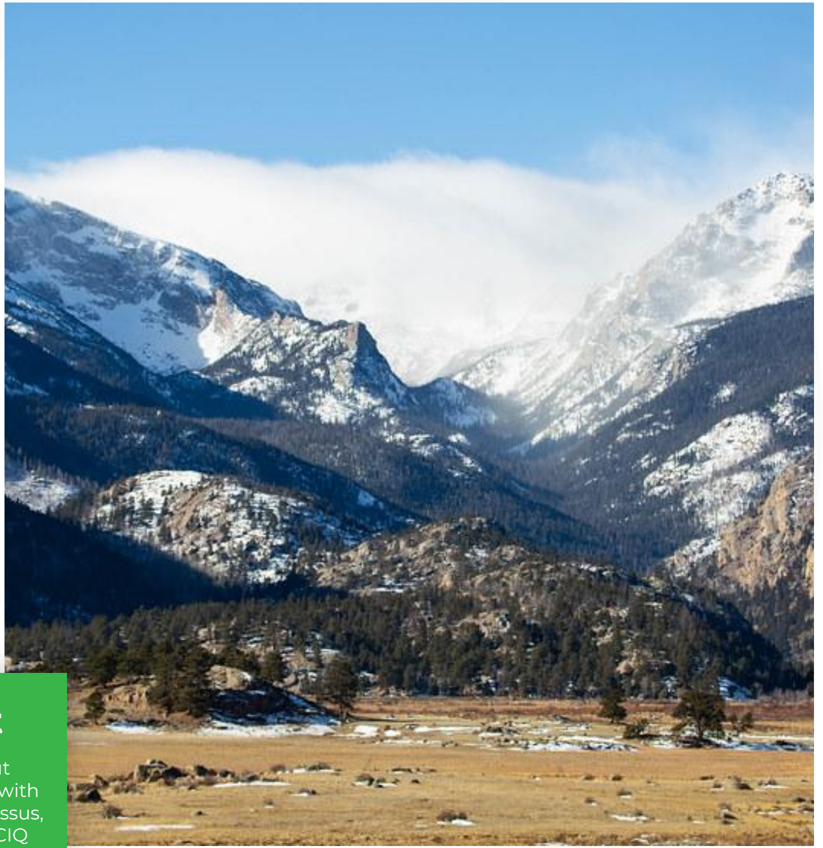
Rocky Linux is designed to be 100% compatible with the Enterprise Linux family of distributions

Open Source

The Rocky Enterprise Software Foundation (RESF) is the organization behind Rocky Linux, community led, supported, and maintained.

Compliant

Rocky Linux is all about security and compliance with accreditations like CIS, Nessus, RESF Secure Boot, and CIQ sponsoring FIPS certification.



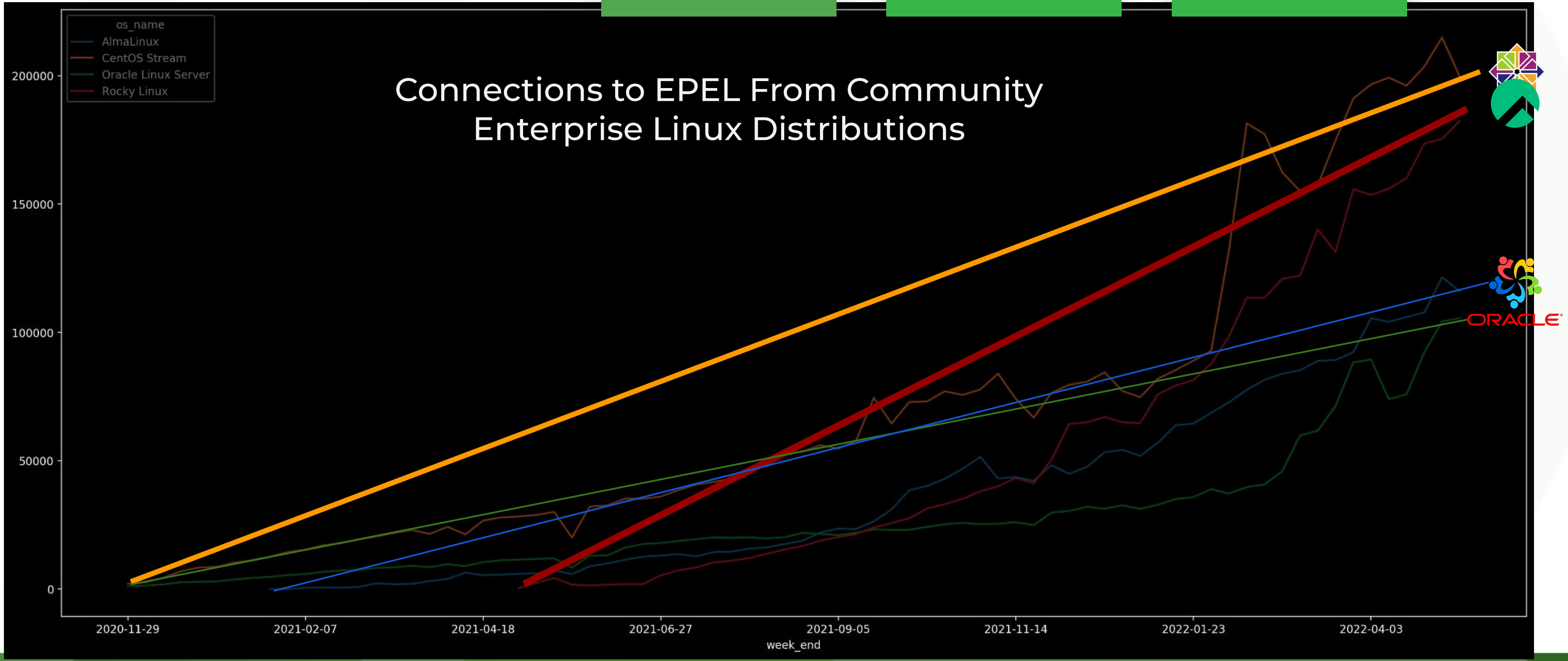
Rocky Linux

Growth

Already over 6% of enterprise infrastructure

Tens of thousands of community members

Google + CIQ Supports Rocky Linux



Rocky Linux was first released in June, 2021

In less than a year...

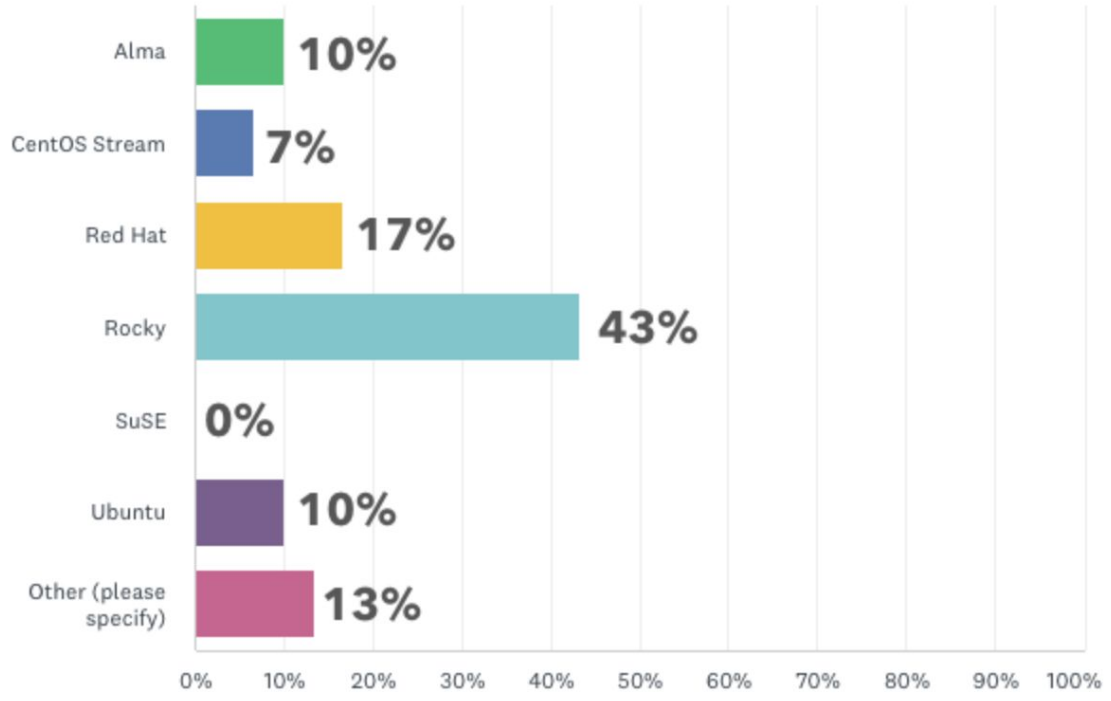
“Among all respondents,...

7.7% indicated Rocky Linux was a “primary” operating system and 12.9% “secondary.”

That gives Rocky Linux a presence among more than one in five respondents.”

– Intersect360 Research, 2022

Usage in HPC is Even Higher!



https://www.advancedclustering.com/centos-migration/

CIQ

Apptainer / Singularity

Containers for HPC

Kurtzer founded Singularity in 2015, and since then, it has risen to being one of the most utilized tools in HPC.

It has now been moved into the Linux Foundation and renamed to Apptainer.

CIQ, is the official commercial support arm of Apptainer.

As Docker brought containers into enterprise, Singularity brought containers into HPC with a containerization strategy that just works for HPC architectures, security models, and use-cases.

Now Singularity is part of the Linux Foundation and re-released under a new name for the open source project, Apptainer.

NVIDIA. intel. SUSE openHPC AMD arm

CIQ

Warewulf

Operating System Imaging and Provisioning

Kurtzer founded Warewulf in 2001, and it has become the most successful and long lived cluster management solution in HPC.

The project lives on today, with Kurtzer still leading the effort as it has evolved to become the core of OpenHPC, a joint Intel and Linux Foundation project.

Warewulf today provisions containers to bare metal servers statelessly.

Warewulf is a bare metal, stateless, open source cluster provisioning solution to facilitate the creation and operating system management of large quantities of hardware resources.

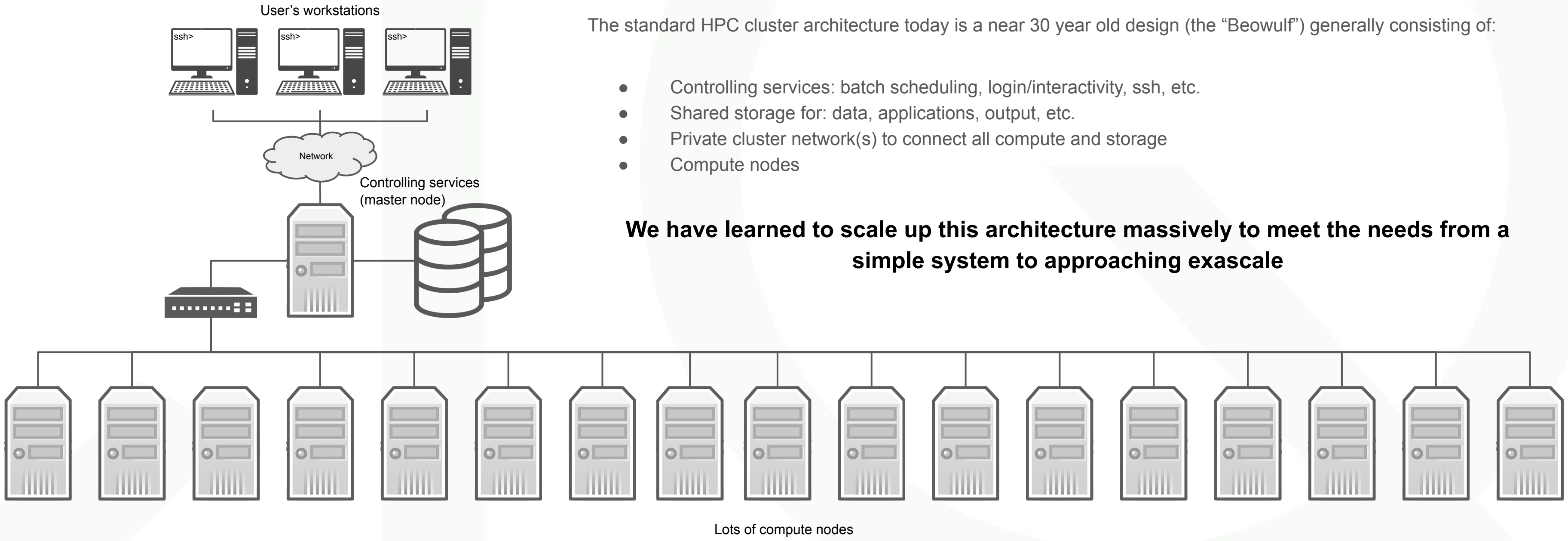
arm AMD SUSE NVIDIA openHPC intel

Fuzzball / HPC-2.0

Cloud Native, Federated, Compute Orchestration

Traditional HPC (legacy)

Nearly Every HPC System Today is Based on a Legacy Architecture



The standard HPC cluster architecture today is a near 30 year old design (the "Beowulf") generally consisting of:

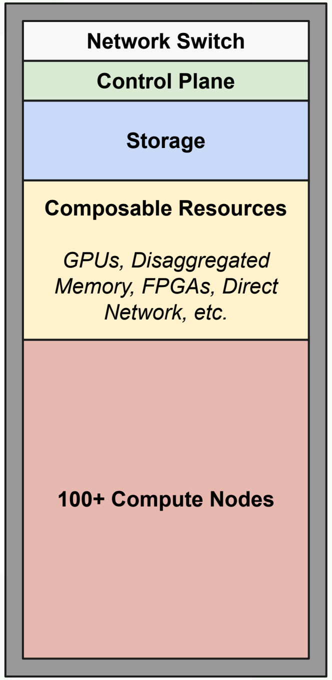
- Controlling services: batch scheduling, login/interactivity, ssh, etc.
- Shared storage for: data, applications, output, etc.
- Private cluster network(s) to connect all compute and storage
- Compute nodes

We have learned to scale up this architecture massively to meet the needs from a simple system to approaching exascale

HPC-2.0 Driving Forces

1. **HPC:** Use-cases are getting more complicated
2. **Pandora's Container:** Containers have opened the door to the benefits of HPC modernization and cross pollination with enterprise, cloud, and hyperscale capabilities
3. **Job Diversity:** We are seeing a much greater breadth of applications, requirements, and dependencies
4. **Usage Models: "**SSH is for system administrators, not users" -- there are many desires to get users off of the systems and instead working through controlled and monitored APIs and IAM policies
5. **Cloud**: There has been a consistent and constant push for cloud/multi-cloud and distributed computing
6. **Supply Chain Security:** Due to the increase in supply chain exploits, security, confidence, and integrity is of the utmost importance from companies to executive orders
7. **Democratization:** Consumers of HPC (including non-traditional HPC users) are becoming more interested in the production, velocity, and utility of HPC, rather than software engineering
8. **Enterprise Computing:** Organizations who have never considered HPC before are now becoming consumers of HPC and HPC-like capabilities and refuse to build a "legacy" Beowulf architecture

A Cloud Scale Hybrid HPC System

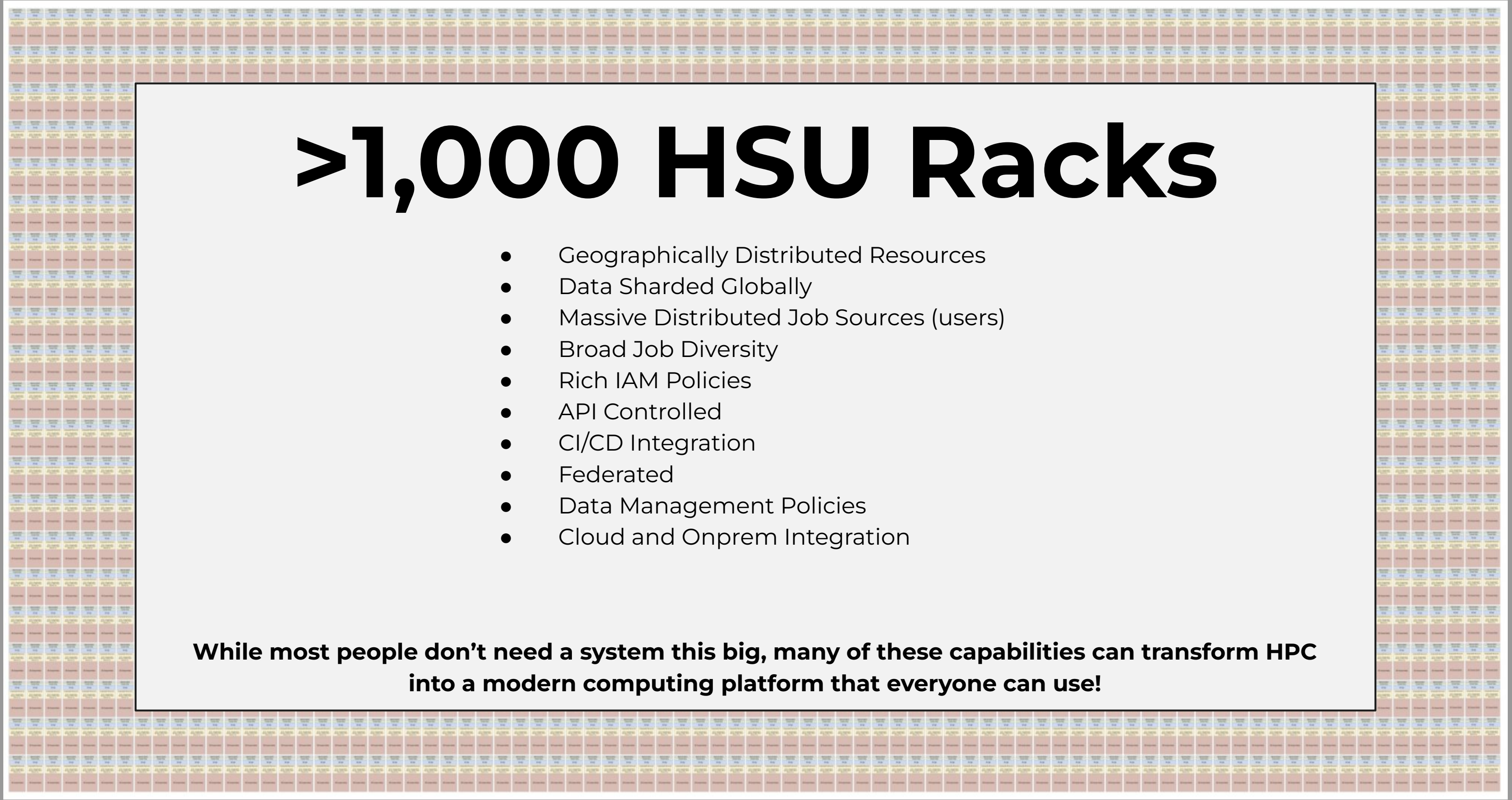


HPC systems today are flat and monolithic; cloud scale systems are not designed this way to make better use of resources, distribution, and scale.

To build a Hyperscale HPC system, we would start from the basic building blocks and work upward.

The single cluster building block:

"Hyper-Scalable Unit" (HSU)



>1,000 HSU Racks

- Geographically Distributed Resources
- Data Sharded Globally
- Massive Distributed Job Sources (users)
- Broad Job Diversity
- Rich IAM Policies
- API Controlled
- CI/CD Integration
- Federated
- Data Management Policies
- Cloud and Onprem Integration

While most people don't need a system this big, many of these capabilities can transform HPC into a modern computing platform that everyone can use!

HPC-2.0 Additional Features and Capabilities

- **Traditional HPC:** Supports and optimized for typical HPC workloads
- **Enterprise Computing:** Also designed for AI/ML, ML-ops, as well as compute and data driven analytics
- **Performance:** Get totally out of the way, enable parallelization, and ensure micro-architecture optimization
- **Scale:** The system must support not only horizontal and vertical scaling, but also "*Z-axis scaling*" (meta-orchestration)
- **Data Management:** Automatic staging/unstaging of data according to data locality, mobility, gravity, and security
- **Composable Workflows:** Workflows consist of job graphs, data, dependencies, and fully composable
- **Advanced Federated Scheduling and Orchestration:** Driven by an intelligent AI enabled policy engine
- **Interface Forwarding:** JupyterHub, Remote desktops, VSCode, Remote debugging/profiling, Matlab
- **Cloud Integration:** Multi-cloud computing and/or bursting model with automatic elastic scaling up and down
- **Security:** Credentials, users, access is all distributed and managed by the system via IAM policies
- **Auditing/Compliance:** All workflows and resources are 100% auditable, reproducible, and policy controlled
- **API Driven:** Enterprises don't want individuals having local Unix accounts on resources (SSH is so 1990's)

While an astute HPC system engineer can build *some* of these things on top of traditional HPC, much of this is out of alignment and thus adds layers of indirection and complexity

While enterprise orchestration systems can do *some* of this, they don't generally meet the needs of modern HPC where data awareness, advanced resource allocation, and rich policy controls are required.

We need to cross-pollinate and innovate

This Doesn’t Exist…

So We Built It.

Fuzzball: Workflows

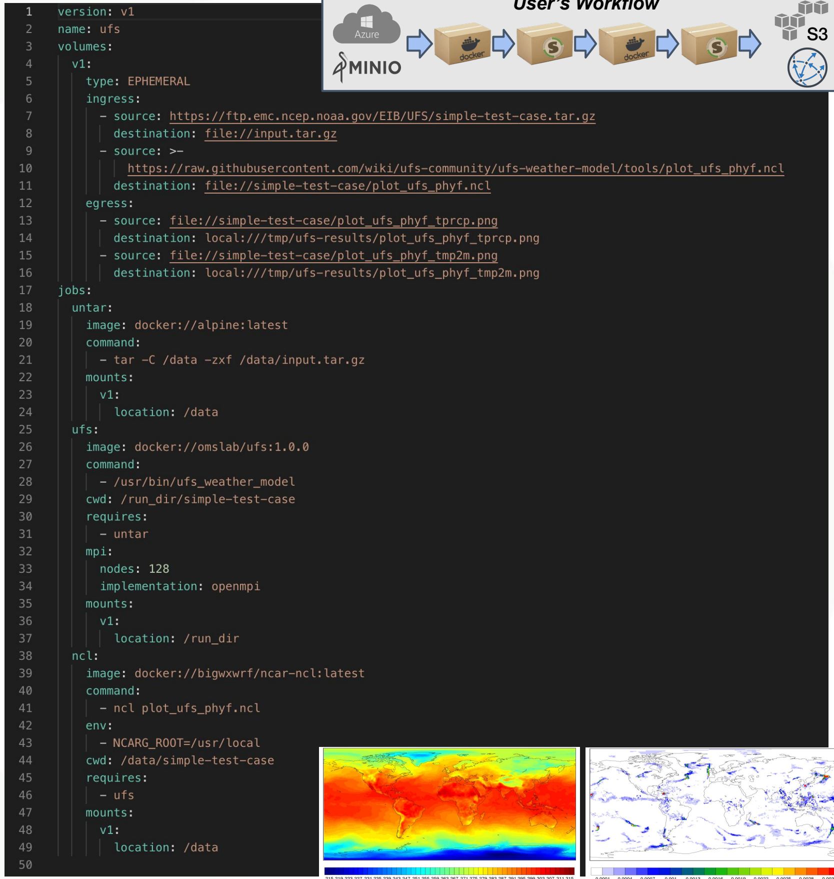
Workflows need to support three primary aspects:

1. **Data/Volumes:** All necessary input, output, configuration, and container data must be defined so Fuzzball can properly manage all data, volumes, and access security within a composed system

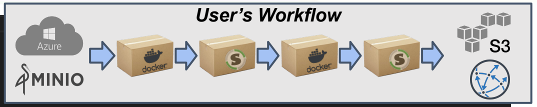
2. **Compute:** Each job step is defined within an acyclic graph and runs within a defined container and supports arrays, parallelization, and dependencies

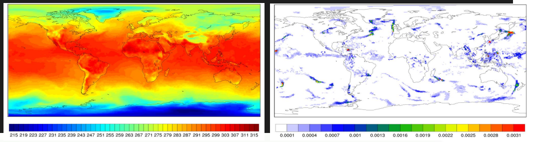
3. **Resources:** Any resource dependencies for a workflow and/or jobs (e.g. GPUs, memory, cores/sockets) are defined such that the workflow can be scheduled to compute resources as well as federated clusters and auto-scaling cloud resources

With this information, we can develop completely composable compute environments on demand for each pipeline, and those workflows will be orchestrated to the most efficient resources in an automated manner.



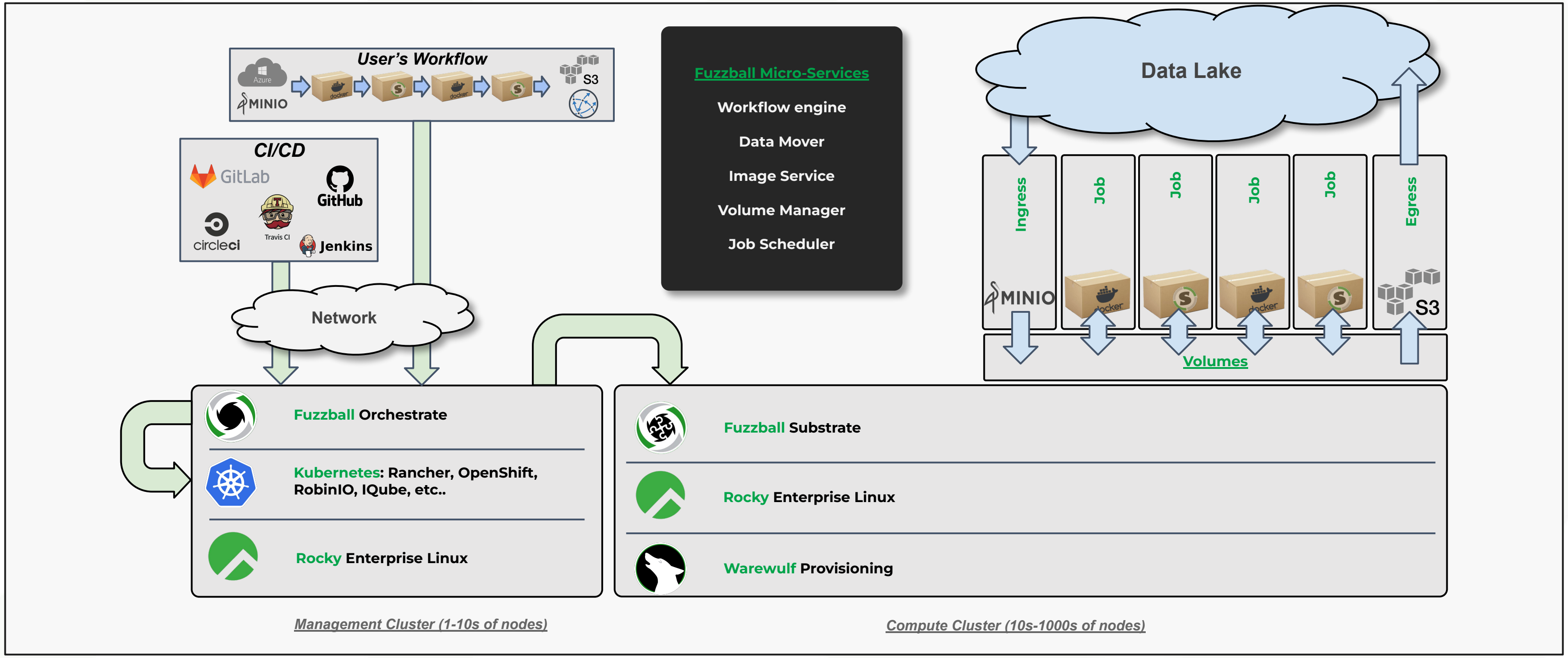
```yaml
version: v1
name: ufs
volumes:
  v1:
    type: EPHEMERAL
    ingress:
      - source: https://ftp.emc.ncep.noaa.gov/EIB/UFS/simple-test-case.tar.gz
        destination: file://input.tar.gz
      - source: >-
          https://raw.githubusercontent.com/wiki/ufs-community/ufs-weather-model/tools/plot_ufs_phyf.ncl
        destination: file://simple-test-case/plot_ufs_phyf.ncl
    egress:
      - source: file://simple-test-case/plot_ufs_phyf_tprcp.png
        destination: local:///tmp/ufs-results/plot_ufs_phyf_tprcp.png
      - source: file://simple-test-case/plot_ufs_phyf_tmp2m.png
        destination: local:///tmp/ufs-results/plot_ufs_phyf_tmp2m.png
jobs:
  untar:
    image: docker://alpine:latest
    command:
      - tar -C /data -zxf /data/input.tar.gz
    mounts:
      v1:
        location: /data
  ufs:
    image: docker://omslab/ufs:1.0.0
    command:
      - /usr/bin/ufs_weather_model
    cwd: /run_dir/simple-test-case
    requires:
      - untar
    mpi:
      nodes: 128
      implementation: openmpi
    mounts:
      v1:
        location: /run_dir
  ncl:
    image: docker://bigwxwrf/ncar-ncl:latest
    command:
      - ncl plot_ufs_phyf.ncl
    env:
      - NCARG_ROOT=/usr/local
    cwd: /data/simple-test-case
    requires:
      - ufs
    mounts:
      v1:
        location: /data
```





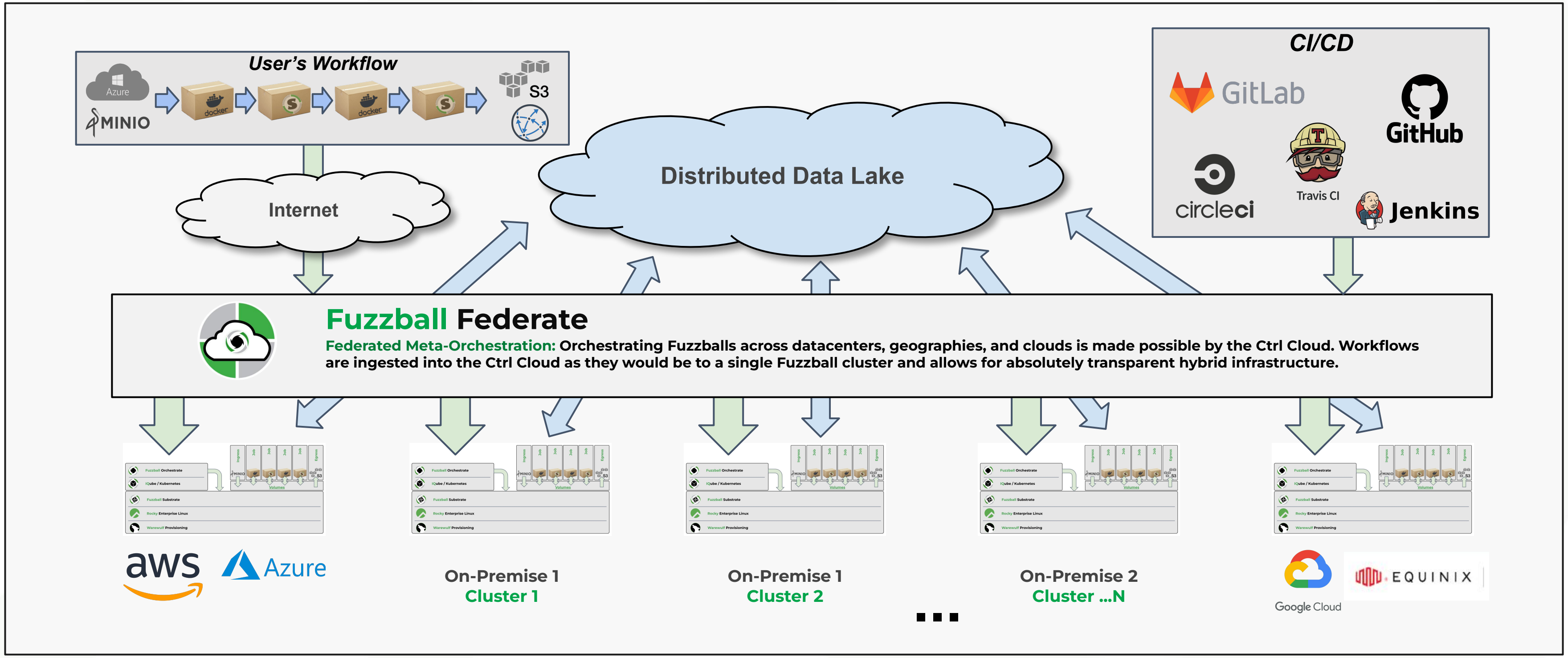
Fuzzball: Orchestrate

A **Fuzzball cluster** allows users to submit workflows to the **Fuzzball Orchestration services,** which then manages the data, container image management, and volumes and then submits the job graph to the scheduler to reserve resource space on the compute resources. The jobs are run via **Fuzzball Substrate service** which manages those workflows on thousands of compute nodes!



Fuzzball: Federate

Fuzzball Federate allows us to **scale out** and **unite** Fuzzball clusters at massive scale into a **single seamless platform** for users and CI/CD to interact with over **geographically distributed** sites and clouds.



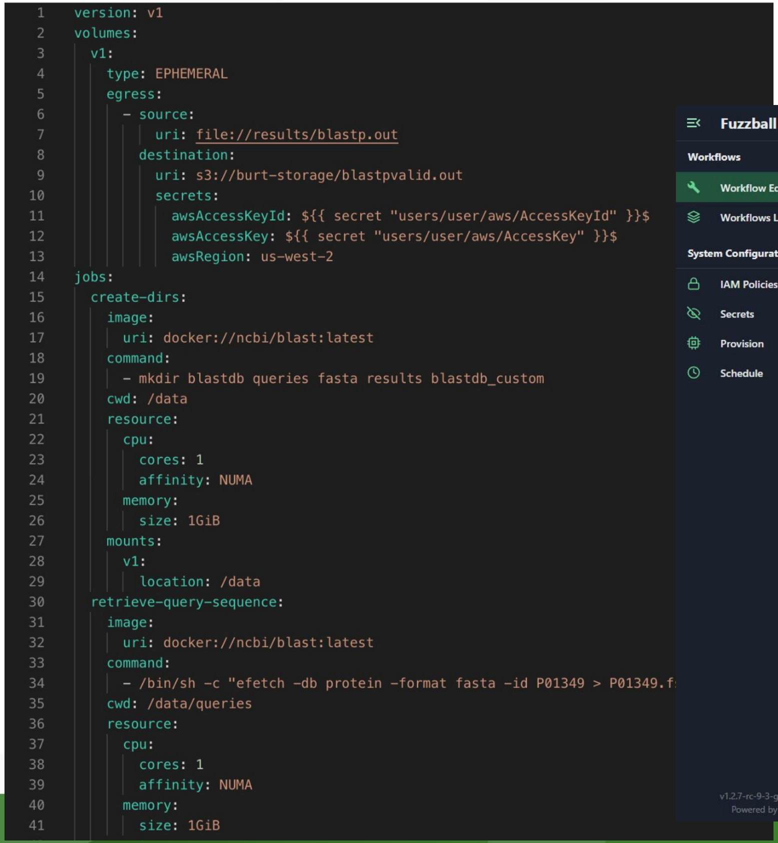
Fuzzball Federate

Federated Meta-Orchestration: Orchestrating Fuzzballs across datacenters, geographies, and clouds is made possible by the Ctrl Cloud. Workflows are ingested into the Ctrl Cloud as they would be to a single Fuzzball cluster and allows for absolutely transparent hybrid infrastructure.

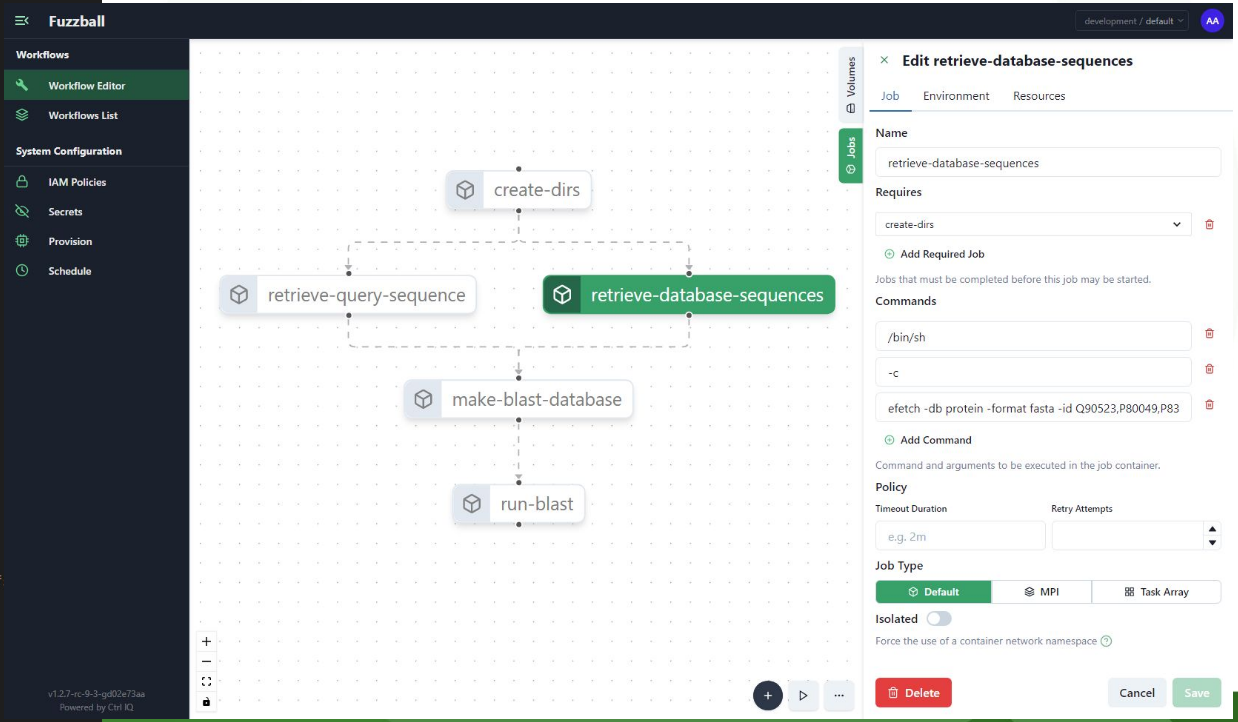
Fuzzball: UI

Unified Common Interfaces to Simplify Workflow Development and Users

Unifying the experience across clouds, sites, clusters, and nodes while maintaining advanced numa level node placement of CPUs, GPUs, FPGAs, IPUs, etc



```yaml
version: v1
volumes:
  v1:
    type: EPHEMERAL
    egress:
      - source:
          uri: file://results/blastp.out
        destination:
          uri: s3://burt-storage/blastpvalid.out
          secrets:
            awsAccessKeyId: ${{ secret "users/user/aws/AccessKeyId" }}$
            awsAccessKey: ${{ secret "users/user/aws/AccessKey" }}$
            awsRegion: us-west-2
jobs:
  create-dirs:
    image:
      uri: docker://ncbi/blast:latest
    command:
      - mkdir blastdb queries fasta results blastdb_custom
    cwd: /data
    resource:
      cpu:
        cores: 1
        affinity: NUMA
      memory:
        size: 1GiB
    mounts:
      v1:
        location: /data
  retrieve-query-sequence:
    image:
      uri: docker://ncbi/blast:latest
    command:
      - /bin/sh -c "efetch -db protein -format fasta -id P01349 > P01349.f
    cwd: /data/queries
    resource:
      cpu:
        cores: 1
        affinity: NUMA
      memory:
        size: 1GiB
```



Edit retrieve-database-sequences

Job | Environment | Resources

Name: retrieve-database-sequences

Requires: create-dirs

Add Required Job

Jobs that must be completed before this job may be started.

Commands:
/bin/sh
-c
efetch -db protein -format fasta -id Q90523,P80049,P83

Add Command

Command and arguments to be executed in the job container.

Policy

Timeout Duration: e.g. 2m

Retry Attempts:

Job Type: [✓ Default] MPI Task Array

Isolated

Force the use of a container network namespace

Delete Cancel Save

Fuzzball: CLI

Fuzzball is 100% API driven; users don’t use SSH or obtain access to a non-secured container or VM instance.

This is an example of the Fuzzball CLI which runs locally on the user’s or admin’s workstation. Other interfaces are in development.

```
$ fuzzball context create cloud fuzzball.cluster:443


$ fuzzball context list
ACTIVE  NAME   ADDRESS
*       cloud  fuzzball.cluster:443
        demo   demo.cluster:7331


$ fuzzball context use demo
Configuration for "demo" now in use.


$ fuzzball workflow start counter-demo ./counter.yaml
Workflow “counter-demo” started.


$ fuzzball workflow list
NAME                        | STATUS    | STARTED
counter-demo                  | Started  | 2022-02-01 12:02:28PM


$ fuzzball workflow attach counter-demo counter_job
22
23
...


$ fuzzball workflow exec --tty counter-demo counter_job /bin/sh
Fuzzball> ps aux
...
```

Fuzzball: Workflows

Task Arrays

Jobs within a workflow are highly extensible acyclic graphs, and within each individual job, you can apply arrays and dependencies to create needed job shapes to achieve very complex tasks.

```
version: v1

jobs:
  task-sleeper:
    image:
      uri: docker://alpine:latest
    command: ["/bin/sleep", "$FB_TASK_ID"]

    policy:
      timeout:
        execute: 2m
      retry:
        attempts: 1

    resource:
      cpu:
        cores: 1
        affinity: NUMA
      memory:
        size: 1GB

    task-array:
      start: 1
      end: 32
      concurrency: 8
```

Fuzzball: Workflows

MPI

Fuzzball is fully MPI aware and manages the multi-node wire-up for multiple MPI implementations as well as Infiniband and advanced numa architecture job placement.

```
version: v1

jobs:
  hello:
    image:
      uri: docker://mpi_example:latest
    command: ["hello-mpi"]

    policy:
      timeout:
        execute: 20m
      retry:
        attempts: 1

    resource:
      cpu:
        cores: 2
        affinity: NUMA
      memory:
        size: 1GB

    mpi:
      nodes: 64
      implementation: openmpi
```

Fuzzball: Workflows

Persistent Volumes

Persistent volumes indicate a volume that is configured on a specific resource where the data is already staged and available (e.g. genomic, astrophysics, etc.).

The system administrator can make these volume available as a read-only or read-write depending on IAM policies.

```
version: v1

volumes:
  v1:
    type: PERSISTENT
    namedVolume: default

jobs:
  touch:
    image:
      uri: docker://alpine:latest
    command: ["/bin/touch", "/volume/foo"]

    mounts:
      v1:
        location: /volume

 test:
    image:
      uri: docker://alpine:latest
    command: ["/bin/test", "-f", "/volume/foo"]

    mounts:
      v1:
        location: /volume

    requires: [touch]
```

Fuzzball: Workflows

Secrets and GPUs

This workflow is the HPL-AI benchmark from Nvidia and it confirms the expected performance on Nvidia DGX A100.

Also, notice the embedded secrets and credentials that Fuzzball supports internally.

```
version: v1
jobs:
  run-hpl-ai:
    image:
      uri: docker://nvcr.io/nvidia/hpc-benchmarks:21.4-hpl
      secrets:
        username: $oauthtoken
        password: ${{ secret "path/to/ngc-secret" }}$

    command: [hpl.sh, --xhpl-ai, --config, dgx-a100, --dat, /workspace/hpl-ai-linux-x86_64/sample-dat/HPL-dgx-a100-1N.dat]

    resource:
      cpu:
        cores: 128
        affinity: NUMA
      memory:
        size: 2000GB
        by-core: false
      devices:
        nvidia.com/gpu: 8
    mpi:
      nodes: 1
      implementation: openmpi
```

Fuzzball: Workflows

Ethereum Mining / T-Rex

Fuzzball easily supports high level of job and workflow diversity.

This is an example of Ethereum cryptocurrency mining using a T-Rex container that was built and updated using a private GitLab CI pipeline.

```
version: v1
jobs:
  mine-ethereum:
    image:
      uri: docker://registry.gitlab.com/workflows/trex-miner:docker-stable
      secrets:
        username: gmk
        password: ${{ secret "users/user/gitlab/PAT" }}$
    command: ["/bin/sh", "-c", "./ETH-ethermine.sh"]
    network:
      isolated: true
    cwd: /trex
    policy:
      timeout:
        execute: 5m
    resource:
      cpu:
        cores: 2
        affinity: NUMA
      memory:
        size: 12GB
      devices:
        nvidia.com/gpu: 1
```

Fuzzball: Workflows

Breaking GPU Vendor Lock-in

By Changing these two lines, you can go from Nvidia/CUDA to AMD/ROCM (or other GPU implementation) and Fuzzball will locate and execute on the right hardware (if it exists) to drive optimal TCO.

```
version: v1
volumes:
  v1:
    type: EPHEMERAL
    ingress:
      - source:
          uri: https://lammps.sandia.gov/inputs/in.lj.txt
        destination:
          uri: file://in.lj.txt

jobs:
  run-lammps:
    image:
      uri: docker://nvcr.io/hpc/lammps:29Sep2021
    command: [lmp, -k, on, g, 1 -sf, kk, -pk, kokkos, cuda/aware, on, neigh, full, comm, device, binsize, 2.8, -var, x, 8, -var, y, 4, -var, z, 8, -in, /data/in.lj.txt]
    cwd: /data

    resource:
      cpu:
        cores: 1
        affinity: NUMA
      memory:
        size: 12GiB
        by-core: false
      devices:
        nvidia.cpm/gpu: 1

    mounts:
      v1:
        location: /data

    mpi:
      nodes: 32
      implementation: openmpi
```

Fuzzball: Workflows

Jupyter / Port Forwarding

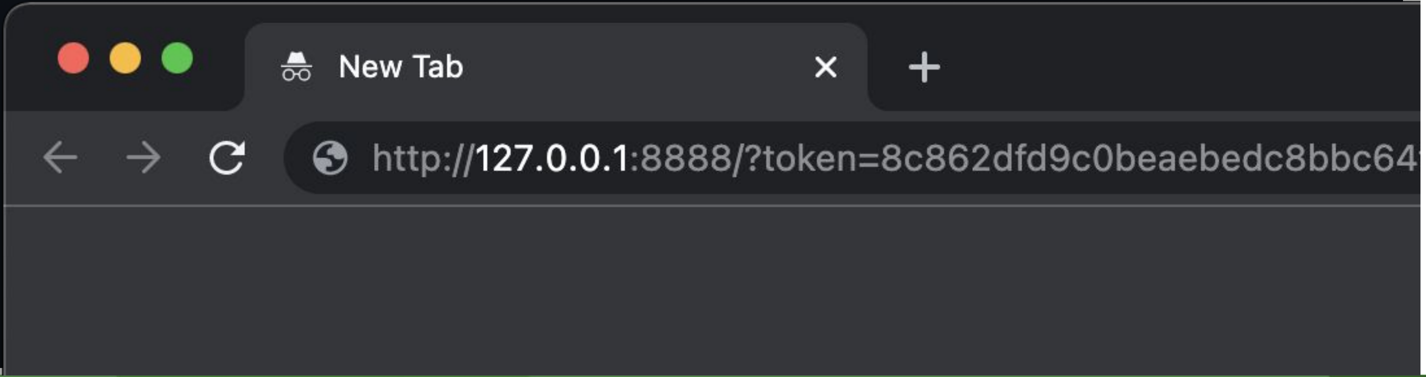
Absolute native integration with graphical Jupyter notebooks, remote desktop virtualization, or any other port forwarding use-cases:

```
$ fuzzball workflow start jupyter-notebook.yaml
Workflow Started with id fae7c506-cbdc-4ab1-80fd-5167a5f6a250

$ fuzzball workflow log fae7c506-cbdc-4ab1-80fd-5167a5f6a250 | tail -n 1
jupyter              |      or http://127.0.0.1:8888/?token=8c862dfd9c0beaebedc8bbc64f5d17f23af36d3f8d0b6dad

$ fuzzball workflow port-forward fae7c506-cbdc-4ab1-80fd-5167a5f6a250 jupyter 8888:8888
2021/08/18 13:54:52 Listening on 127.0.0.1:8888
```

```
version: v1
jobs:
  jupyter:
    image:
      uri: docker://jupyter/minimal-notebook
    command: ["jupyter", "notebook", "--port", "8888", "--no-browser", "--allow-root"]
    network:
      isolated: true
    resource:
      cpu:
        cores: 16
        affinity: NUMA
      memory:
        size: 256GB
        by-core: false
      devices:
        nvidia.com/gpu: 1
```



CIQ

Rocky, Apptainer, Warewulf, and Fuzzball / HPC-2.0

OpenFabrics Alliance

Innovations in HPC Software Infrastructure