



The Clinical Data
Science Conference

EU 2025

16–19 November

CCH – Congress Center Hamburg

Paper TT-13

Validated Statistical Computing: A Kubernetes Framework for Regulatory- Compliant R, Python and SAS Deployments

Randy Betancourt, Siemens, Portland OR, USA

Tim Fantuzzo, Siemens, Troy MI, USA

Agenda

- SAS Workload Processing Architectures
- Single Server Deployment Model
- SLC Hub Architecture
- Kubernetes Deployment Model
- Comparative Analysis
- Decision Framework
- Recommendations

The Challenge

Pharmaceutical Industry Pressures

- Adopting open-source tools (R, Python) alongside SAS
- Maintaining FDA compliance (21 CFR Part 11)
- Managing multiple software and hardware stacks
- Administrative burden of disparate systems

Key Requirements

- Data integrity and traceability
- Audit trails and electronic signatures
- System validation documentation
- Change control processes
- Reproducibility of statistical analyses

Decision Framework & Priorities

1. Infrastructure Stability & Auditability

- FDA compliance and validation requirements
- Change control and reproducibility needs
- Validation against a known benchmark

2. Administrative Skills at Hand

- Traditional IT vs. cloud-native expertise
- Training investment and learning curves

3. Infrastructure Architecture & Deployment Choices

- Single server
- SLC Hub
- Kubernetes
- Hybrid
- Agreements on HA requirements and scalability needed

4. Cloud Migration & Modernization

- On-premises to cloud transition actions
- Hybrid and multi-cloud considerations
- Migration impacts on stock of users' code

5. Programming Language Preferences

- Shifting from SAS to R and Python
- Impact on infrastructure requirements

6. SaaS Analytics Integration

- Databricks, Snowflake adoption
- Integrating SAS workflows with ML pipelines

Three Deployment Models

- **Single Server Deployment**

- Traditional client-server model
- Suitable for <20 concurrent users

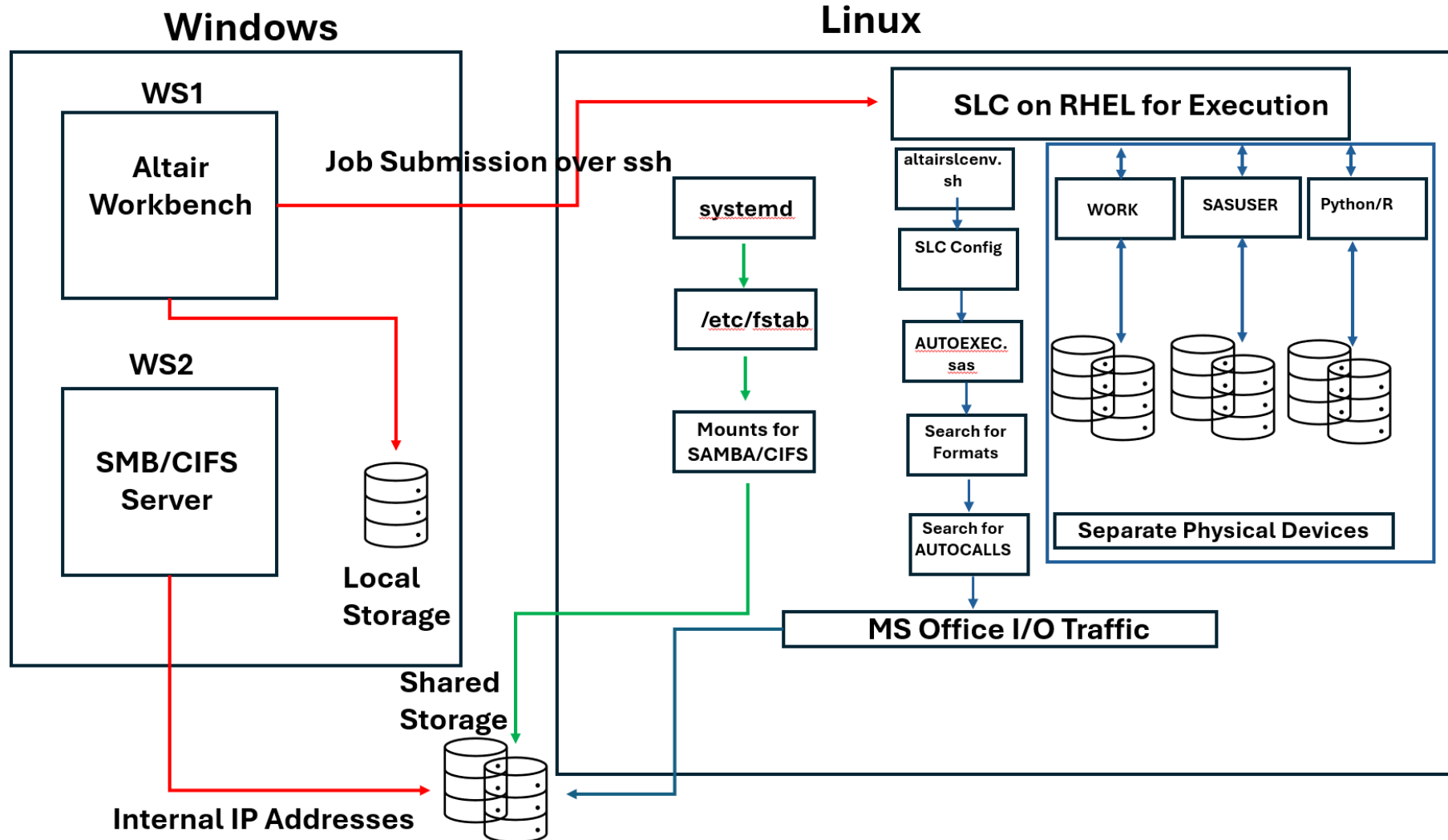
- **SLC Hub Architecture and/or SAS Grid**

- Vendor supported enterprise-class platform
- Built-in orchestration and scheduling
- Provides governance and controls

- **Kubernetes Deployment**

- Non-vendor supported enterprise-class platform
- Cloud-native platform
- Maximum flexibility and portability

Single Server Deployment



Single Server Deployment

Advantages

- Vendor supported
- Well-understood provisioning
- Low ongoing overhead
- Simple storage requirements

Disadvantages

- Single point of failure (SPOF)
- No horizontal scaling
- No load balancing
- Limited to vertical scaling
- No enterprise features (authdomains, built-in scheduler)

Single Server Deployment: Use Cases

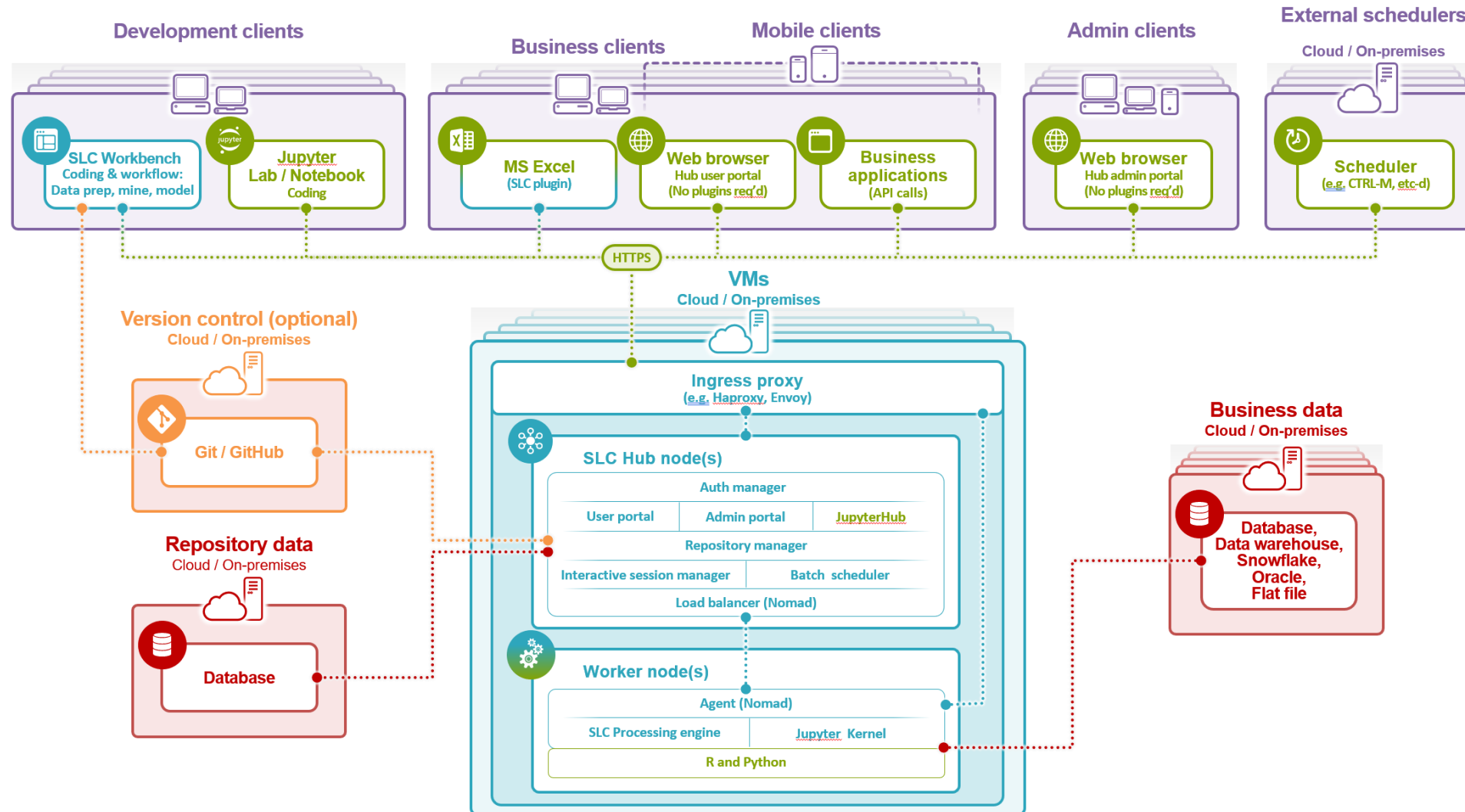
Ideal Use Case

- Line-of-business deployments
- Less than 20 concurrent users
- Development and test environments
- Non-HA requirements
- Budget-constrained projects

Not Suitable For

- Meeting high-availability requirements
- DR/failover scenarios
- Enterprise-scale deployments
- Thousands of users

SLC Hub Architecture Overview



SLC Hub Advantages

Simplified Operations

- Cohesive, pre-integrated software stack
- Web-based GUI for all admin tasks
- Point-and-click configuration for users
- Documented installation procedures

Vendor Support

- Direct access to support engineers
- Tested component interactions
- Validated upgrade paths
- FDA audit support

Lower Skill Requirements

- Familiar to IT teams
- Standard system administration skills
- No container/Kubernetes expertise needed
- Gentler learning curve

Enterprise Features

- Auth domains for credential management
- Abstract database connections
- Simplified infrastructure changes
- Reduced validation burden

SLC Hub Disadvantages

Limited Cloud Integration

- VM-based worker nodes (not containers)
- Cannot leverage Kubernetes-native tools
- Limited service-mesh integration
- Coarser scaling granularity (VM vs. pod level)
- Non-portable deployment model to other public cloud providers

High Availability Gaps

- Native HA not available until early 2027
- Requires third-party clustering (e.g., Veritas)
- Additional complexity and licensing costs

Kubernetes Architecture Overview

Cloud-Native Cluster Management

- Container-based deployment with self-healing properties
- Declarative configuration (YAML)
- Infrastructure as Code
- Specifies desired state, not procedures
- Native cloud integration
- Version control with Git

Administrative Requirements

- Build separate containerized images (SLC, R, Python)
- Manage image lifecycle
- Dependencies are constrained to container images

Reproducible Environment

- Identical configurations across dev/test/prod
- Historical configuration reconstruction
- Easy and consistent validation evidence for validation

Kubernetes - High Availability

Self Healing

- Automatic pod replacement on failure
- No manual intervention required
- Reduced MTTR
- Can perform auto-scaling in response to changes in workload characteristics

Pod Replication

- Workloads distributed across nodes
- No single points of failure
- Automatic rescheduling on node failure

Built-in Load Balancing

- Fixed & stable service endpoints
- Automatic traffic routing
- Health-based pod selection
- No external load balancers needed

Kubernetes - Change Control

Immutable Container Images

- Cannot modify validated images
- Changes require new image versions
- Prevents configuration drift
- Ensures reproducibility & auditability

Versioned Deployment

- Complete rollout history
- Revision tracking
- Integration with on-going validation protocols
- Audit-ready deployment evidence

Rollback Capabilities

- Instant reversion to previous deployment states
- Single-command rollback and recovery
- Maintains previous configurations
- A risk mitigant to change impacts

Kubernetes Disadvantages—It Takes a Team

Steep Learning Curve

- Container technologies
- Understand networking topologies
- Storage orchestration
- Adhering to security policies
- Understand pod lifecycle management
- Significant training investment
- Significant cross-domain collaboration needed

Complex Troubleshooting

- Multiple abstraction layers
- Application vs. infrastructure issues
- Container runtime problems
- Network debugging
- Storage subsystem issues

Kubernetes Disadvantages—(Continued)

No Single Vendor Support

- Fragmented support relationships
- Container runtime, K8s distribution, storage, networking
- Multiple support contracts
- Complex issue diagnosis and escalation

Ongoing Maintenance

- Kubernetes version upgrades
-
- TLS/SSL certificate rotations & secrets rotation
- etcd backups
- Rapid release cycle
- Frequent upgrade requirements

Summary: Comparative Analysis

Factor	SLC Hub	Kubernetes
Deployment Complexity	Low	High
Skill Requirements	Standard IT	Cloud-native expertise
Vendor Support	Single vendor	Multiple vendors
Time to Production	Faster	Slower
Cloud Integration	Limited	Extensive
Portability	Low	High
Resource Efficiency	Moderate	High
HA Availability	2027 (native)	Built-in
Change Control	GUI-based	Git-based IaC
Learning Curve	Gentle	Steep