NEST testing and deployment pipelines

Dennis Terhorst for the NEST Community

Institute of Neuroscience and Medicine (INM-6 Computational and Systems Neuroscience & Theoretical Neuroscience, Institute for Advanced Simulation (IAS-6) Jülich Research Centre, Member of the Helmholz Association and JARA, Jülich

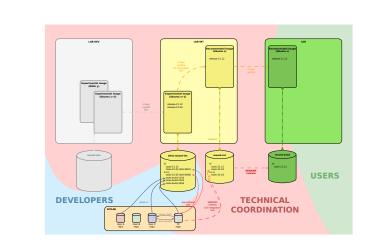


Technical Setup

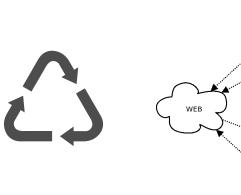
Level	Result	Process Location	Underlying Resources	Environment	Processes	Development Level
Full-stack testing	Tools work in cooperation with external services	framework central CI chain	all	"cron" jobs, uptimerobot, etc.	front-end and service interactions (robotframework)	service & operation
Larger framework integration	Installation in larger software stack with additional constraints on dependencies	framework central CI chain	"cloud" resources, HPC	OpenStack; OpenShift-Jobs gitlab.ebrains.eu//technical-coordination	spack install full environment run use-cases and workflows	framework integration
Packaging & deployments	Different distribution channels are prepared, images and containers are built	package repo CI chain	VMs	github.com//conda-feedstock, dockerhub, obs, *.ebrains.eu	build and test-install various packaging options (deb, rpm, conda, docker,)	ecosystem integration
Local integration tests	Software works in cooperation with other tools	framework-fork CI chain		github.com/nest/nest-simulator, gitlab.ebrains.eu/nest/nest-simulator	run more complex examples, small analyses and workflows	interoperability & interfaces
Benchmark test	no adverse effects on performance	forks near resources	HPC or special hardware	jugit.fz-juelich.de/nest/nest-simulator	beNNch, vTune, perf-tools,	architecture & model independence
Functional tests Regression tests	examples work as described functionality tests simple validation	main central repo CI chain	VMs	github.com/nest/nest-simulator, github.com/ <user>/nest-simulator</user>	own test framework (SLI and Python), boost test suite, pydoctest, pytest, mypy, vale, lychee,	functionality & validation
Unit tests	methods behave as expected	forked repo CI chain				
Static checks	code avoids simple smells	developer-local machine or resource	laptop or workstation	github.com/ <user>/nest-simulator, ~/nest-simulator</user>	cppcheck, clang-format pylint, flake8, pydocstyle, rstcheck	software features & performance

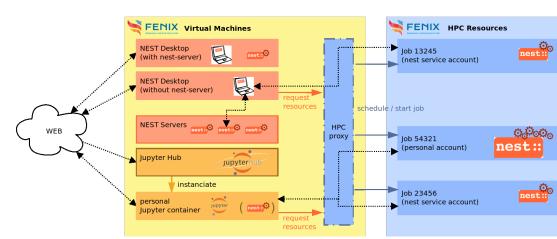
Administrative and Organizational Questions





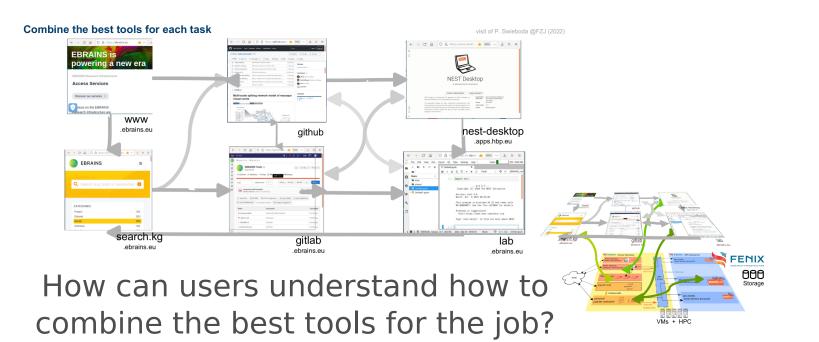
What mechanisms of collaboration and protection are required?



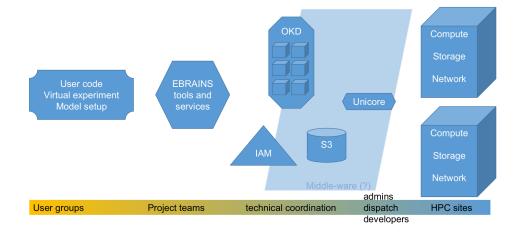


How can different available resources efficiently and effectively be used together?









Responsibility for "middleware"?









