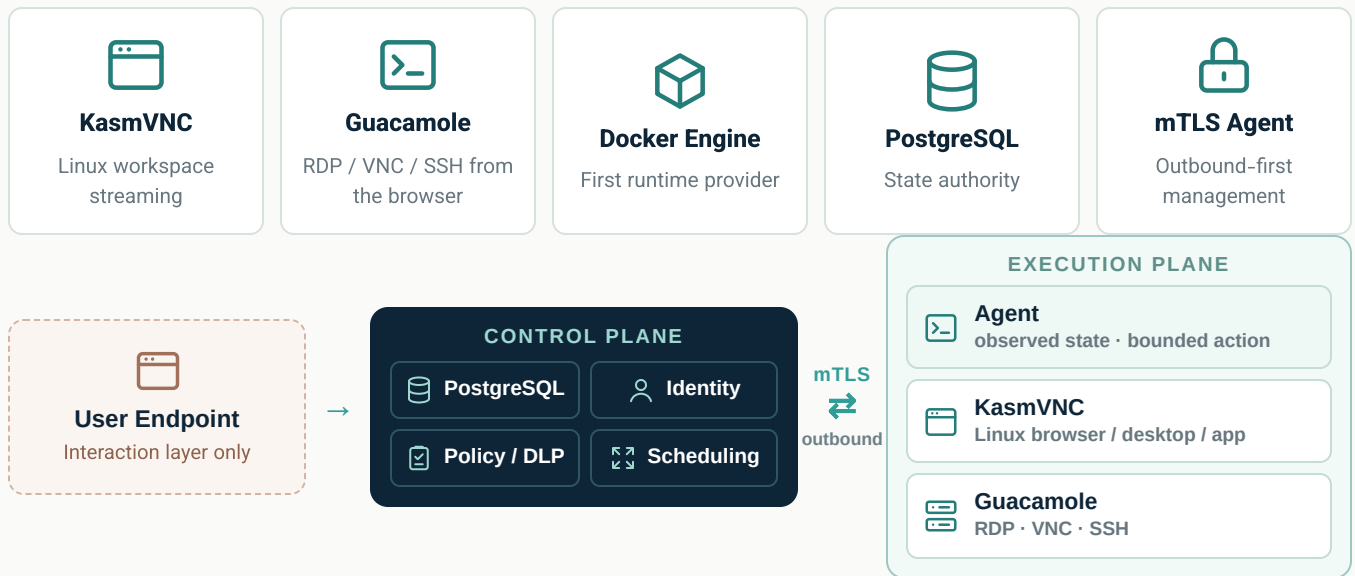


4SO Workspace

Technical Brochure

Secure Browser Workspaces with Separated Control & Execution Planes

Linux Browser, Desktop and Application workspaces plus
RDP/VNC/SSH access from one web experience; with Identity,
Policy and DLP, Scheduling, a durable Session lifecycle, and
protected execution kept separate from the user's device.



The endpoint is not the place of trust

The session and workload execution stay in a controlled environment; the endpoint is only the interaction layer.

Single-node is first-class

Control and Execution can share one host; Kubernetes is a deployment option, not a prerequisite for the experience.

Separate the workspace from the endpoint; keep the user experience simple.

CAPABILITY SNAPSHOT

Capability Snapshot

A simple user experience, serious infrastructure behind it

Workspace keeps User Experience and Infrastructure Operations capabilities side by side in one product.



Browser, Desktop and App

Disposable, controlled Linux environments



RDP / VNC / SSH

Browser access without scattered clients



Enterprise identity

Authentication, Authorization and scope



DLP and Policy

File, Clipboard, Peripheral and Network Path



Scheduling and Capacity

Execution node selection based on state



Multi-zone

Extend Execution without moving Control



Recording

Session-level observation, collaboration and Audit



Observability

Health, Capacity and Audit in one view

Product ownership boundary

- PostgreSQL** – Intent, Session, Policy, Node and Capacity 
- Control ≠ Execution** – two independent roles 
- Agent** – observed state, not authority 
- Single-node** – a first-class deployment model 

Operating principles

- Outbound-first, mTLS-authenticated node management 
- The Browser UI goes only through the Product API 
- Session state is observed from the runtime 
- Kubernetes is optional, not a prerequisite 

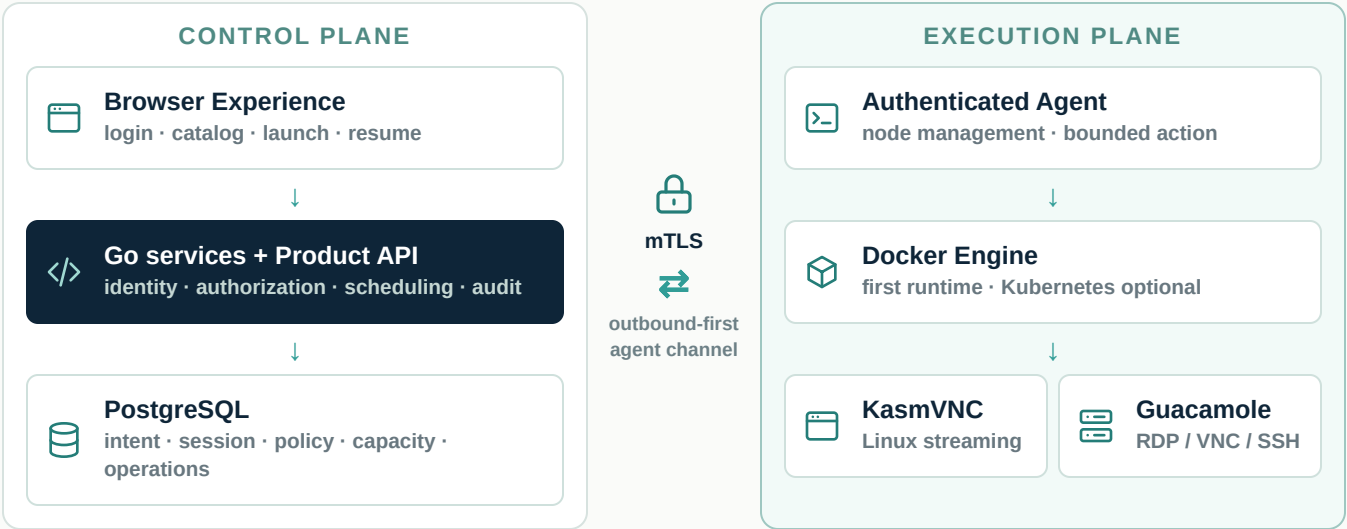
 **Core contract:** Session = Intent + Durable Operation + Runtime Observation – no single stage on its own counts as success.

ARCHITECTURE OVERVIEW

Architecture Overview

Control Plane and Execution Plane are deliberately not the same

Intent and Policy stay in the Control Plane; agents carry out management, and the real session runs in the Execution Plane.



Layer	Component	Contract
State Authority	PostgreSQL	UI and Agent do not create independent sources of state
Control Plane	Go services + Product API	Identity, Scheduling and audit; separate from Execution
Agent Plane	Outbound-first mTLS agents	Authority passes through the Product API and RBAC
Linux Streaming	KasmVNC	Protected execution stays on the Execution Plane
Remote Access	Apache Guacamole	Remote protocols sit behind the same Identity and Policy

Deterministic release
API, Controller, Edge, CLI and migrations ship in one artifact

No direct mutation
The Browser UI never mutates the runtime directly

Observed, not assumed
Session state is read from the runtime

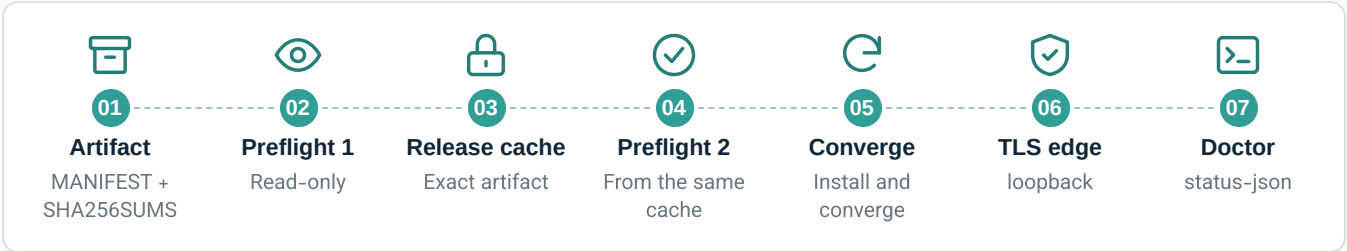
TOPOLOGY / DEPLOYMENT MODEL

Deployment & Install Contract

From a single server to a multi-zone Execution Plane

Profile	Control Plane	Execution	Use
Single node	Same server	Same server	Simple start and small environments
Control + execution nodes	Dedicated	Multiple execution nodes	Separate management from workspace execution
Multi-zone	Central	Execution in multiple zones	Proximity to users and distributed capacity
HA Control Plane	Multi-node	Independent Execution Pool	Control Plane redundancy

ARTIFACT-BOUND INSTALL



Loopback TLS edge

PostgreSQL and the Control API are not host-published

Unsafe cache rejected

Unsafe hard links and permissions are rejected

Safe resume

resume_required and last_safe_resume_point from the owner boundary

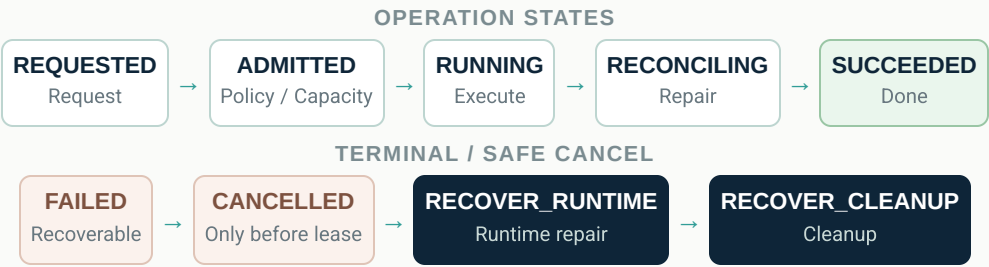
No Kubernetes tax: the runtime provider is an implementation detail beneath the product lifecycle; the normal Workspace experience does not depend on Kubernetes.

ACTION & WORKFLOW CONTRACT

Action & Workflow Contract

A session is not a momentary request

Launch, Stop, Restart and Recovery go from the Product API to Operation and Outbox, and session state is observed separately from request state.



Action	Admission	Execution / Lock	Success	Failure / Recovery
Launch	Actor/Project permission, Catalog, Policy and Capacity	Idempotency-Key → Operation / Outbox → Scheduler / Agent	Session ID and observed runtime	Retry or recovery event; the UI does not guess the outcome
Stop	Existing session and valid permission	Separate Operation to Agent / Runtime	Session is no longer running	Cancellable before lease; Reconcile after that
Restart	Valid source session	Restart attempt + successor binding	Successor observed and usable	Source/successor correlation is preserved
Recover Runtime	Failure / recovery condition	Explicit RECOVER_RUNTIME, not a generic restart	Runtime Ready again	Attempt count and timing kept in History
Recover Cleanup	Provably half-finished runtime	RECOVER_CLEANUP on the same Session	Orphaned resources cleaned and converged	No false successor success is reported
Node Capacity	Health and capacity before Add / Drain / Remove	Scheduling coordinated with node lifecycle	Placement from observed state	Active sessions are never silently orphaned

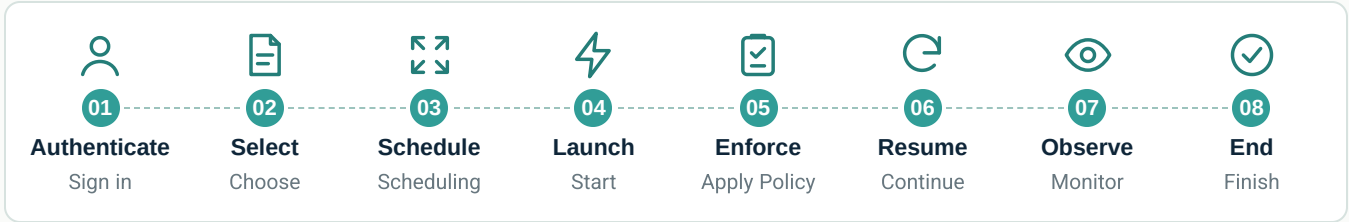
**Safe cancel:** only in REQUESTED or ADMITTED, with no lease owner and before the side-effect boundary is crossed; History keeps the attempts and the latest recovery.

LIFECYCLE

Lifecycle & Workspace Profiles

A session is more than a launch

From authentication to Scheduling, Policy enforcement, Resume, Recording and End, everything is part of the user and infrastructure lifecycle.



Browser Workspace

Isolated browser, separate from the endpoint

Linux Desktop / App

Containerized and Catalog-based

Remote Systems

RDP / VNC / SSH with unified Audit

Single-node

Control and Execution on one host

Distributed

Capacity pool under one Control Plane

Multi-zone

Location-aware Scheduling

Endpoint untrusted

Sensitive execution never stays on the user's device

Control ≠ Execution

Runtime is separate from the state authority

No K8s tax

Kubernetes is optional

Observed session

Readiness comes from the runtime

Deliver

Control

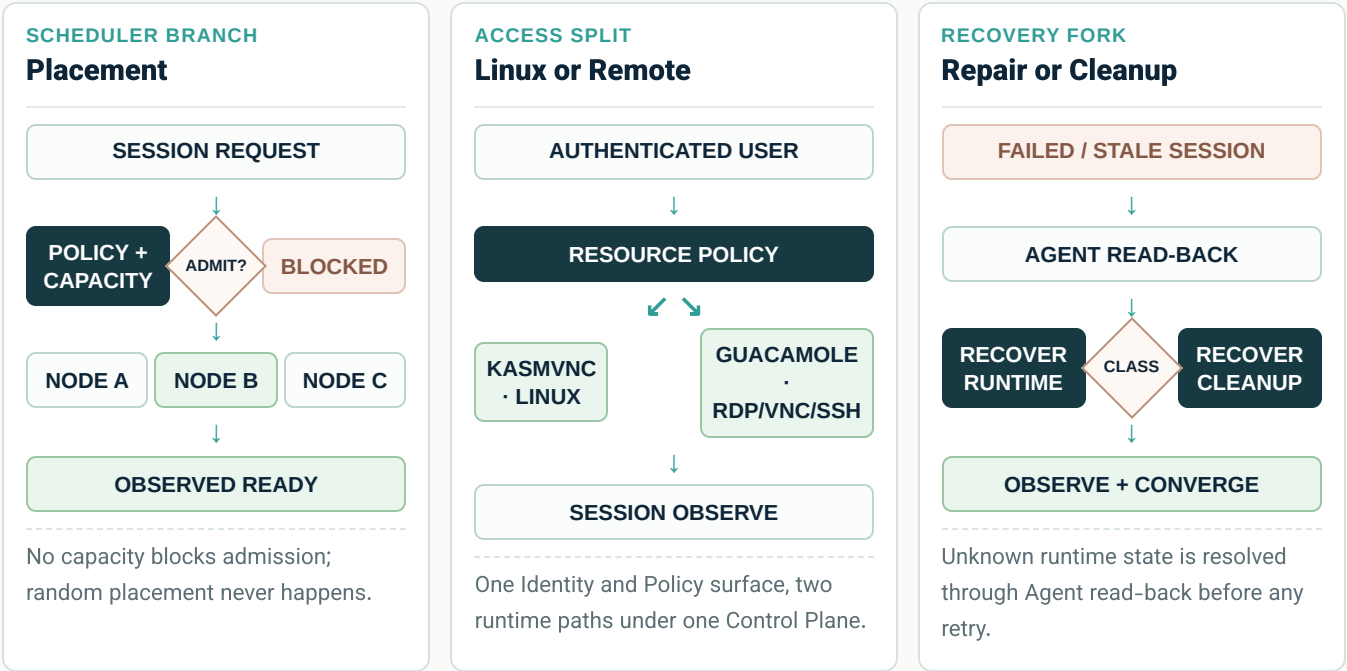
Scale

Recover

DEEP TECH WORKFLOWS

Failure & Recovery Semantics

Browser, Control, Agent and Runtime fail in different ways



Failure Domain	Detection	Product Behavior	Recovery / Fence
Browser disconnect	Client and Session telemetry	The runtime can keep running	Reattach without rebuilding authority
Agent unavailable	Heartbeat and observed state	New mutations blocked or degraded	Agent reconnect and Reconcile
Runtime crash	Failure state on the Session	Session needs attention or recovery	RECOVER_RUNTIME or Restart
Partial cleanup	Orphaned resources discovered	No new success is reported	RECOVER_CLEANUP
Capacity loss	Scheduler and node state	Admission changes	Drain and Relaunch per Policy

Data protection: an ephemeral session is not the user's persistent data; node replacement must not abandon session identity without Reconcile.

ACTION & OPERATIONS CATALOG

Action & Operations Catalog

User Journey and Infrastructure Journey in one product

Workspace •

catalog-list

launch

status

attach

Node •

register

preflight

drain

remove

replace

Policy •

DLP

network policy

access scope

Session •

stop

restart

recover-runtime

recover-cleanup

Capacity •

observe

schedule

rebalance

Diagnostics •

doctor

status-json

operation history

Identity and Trust

Identity and RBAC before admission

mTLS for Agent connection and node trust

Single-use credentials; no secrets in JSON

Observability and Evidence

Status, attention and recovery attempts on the Session

Node and Capacity state for the operator

Unknown instead of a guessed Healthy

Action safety: the UI and Agent consume the same Product API; a shell or runtime bypass has no place in the normal workflow.

A secure workspace, without turning the endpoint into the trust boundary.

4SO Workspace turns secure access from scattered connections into one manageable product.