# Configurable Stack & Runtime Specialization — Addendum v0.1

*Last updated: 2025-08-22*

This addendum extends **Autonomous Agentic SDLC System — Master Plan (v0.1)** to make the stack fully configurable and to allow runtime specialization/assignment of agents per request.

## 1) Configuration Artifacts

### 1.1 STACK\_CONFIG.yaml

languages: [python, typescript]  
frameworks:  
 web: [fastapi, express, nextjs]  
 data: [pandas, polars]  
db: [postgres, sqlite]  
queue: [redis, rabbitmq]  
deploy: [docker, k8s, ecs]  
ci: [github\_actions]  
observability: [otel, prometheus]  
security: [semgrep, trivy, gitleaks]  
constraints:  
 budget\_usd: 50  
 latency\_ms\_target: 500  
 regions: [us-east-1]

### 1.2 TEAM\_PROFILE.yaml

name: "Lean Web"  
description: "Fast iteration for CRUD web apps"  
agents:  
 pm: { model: gpt-5, tools: [kb, tracker] }  
 architect: { model: gpt-5, tools: [mermaid, openapi] }  
 dev\_backend: { model: gpt-5, tools: [pytest, uvicorn], count: 1 }  
 dev\_frontend: { model: gpt-5, tools: [node, playwright], count: 1 }  
 qa: { model: gpt-5, tools: [pytest, coverage, playwright] }  
 secops: { model: gpt-5, tools: [semgrep, trivy, gitleaks] }  
 devops: { model: gpt-5, tools: [terraform, kubectl, gha] }  
policies:  
 approvals: { prod\_deploy: "auto" }  
 coverage\_gate: 0.8  
 risk\_threshold: medium

## 2) Orchestration Changes

* **Agent Registry**: a catalog of agent types, their tools, costs, and rate limits.
* **Capability Graph**: skills → tools → policies; tasks are matched to agents at runtime.
* **Profile Loader**: loads selected TEAM\_PROFILE.yaml and binds tools/models.
* **Dynamic Assignment**: planner assigns agents per task; can override by policy (e.g., high risk → enable HITL critic).
* **Budget/Latency Controller**: enforces constraints and triggers graceful degradation or tool swaps.

## 3) Runtime Selection Flow

1. User creates a **Workspace** and picks a **Profile** or uploads STACK\_CONFIG.yaml.
2. Orchestrator validates config against support matrix; scaffolds project accordingly.
3. Tasks enter the DAG; the **Scheduler** picks agents based on capability/constraints.
4. Policies (coverage, approvals, security) are enforced as CI/CD and OPA gates.

## 4) Repo Additions

repo/  
 configs/  
 STACK\_CONFIG.yaml # per-workspace  
 TEAM\_PROFILE.yaml # selected team profile  
 orchestrator/  
 registry.yaml # agent registry  
 capability\_graph.yaml # task→skills→tools mapping

## 5) APIs (Draft)

* POST /workspaces { name, profile\_name?, stack\_config? }
* POST /workspaces/{id}/requests { text, files[] }
* GET /workspaces/{id}/status
* PUT /workspaces/{id}/profile { profile\_name }
* PUT /workspaces/{id}/stack { stack\_config }

## 6) Acceptance Tests (Examples)

* Given a **Lean Web** profile, the system scaffolds FastAPI + Next.js and deploys to staging.
* Switching to **Enterprise Secure** profile enables OPA checks and artifact signing; deployments require approval.
* Increasing coverage\_gate to 0.9 blocks merges until tests meet the threshold.

## 7) Next Steps

* Implement config parser + JSONSchema validation for both YAMLs.
* Build the **Profile Loader** and **Capability Graph** evaluator.
* Extend CI to read gates from TEAM\_PROFILE.yaml.
* Ship 3 default profiles: Lean Web, Data+ML, Enterprise Secure.