AgentForge: Cloud-Native Multi-Agent AI Platform

Product Requirements & Architecture Document

Project Title: AgentForge
Watermark: Aditya Shenvi © 2025-26
Version: 1.0 (Final)
Date: December 2025
Status: Production-Ready Specification

Page 1: Executive Summary

Vision

AgentForge is an open-source, production-ready platform that democratizes AI agent deployment to the cloud. It enables developers, ML engineers, and enterprises to design, deploy, monitor, and scale multi-agent AI systems across AWS, GCP, and Azure with GitOps automation, full observability, and multi-tenant SaaS architecture—all without vendor lock-in.

Core Value Proposition

- **Drag-and-drop agent builder** with LangGraph backend (no code needed for 80% of use cases).
- **One-click cloud deployment** to Kubernetes (EKS/GKE/DOKS) via Terraform + ArgoCD.
- **Production observability** with OpenTelemetry → Grafana LGTM stack (traces, metrics, logs, dashboards).
- **ML lifecycle tracking** with MLflow (experiments, model registry, artifact management).
- **Multi-tenant SaaS ready** with auth (Clerk), org management, usage metering, and billing hooks.

Target Users

1. **Startups/Scaleups**: Deploy AI prototypes to production in hours, not months.
2. **Enterprise ML teams**: Standardize agent development with governance, audit trails, and cost controls.
3. **Individual developers**: Build portfolio projects that showcase full-stack + cloud + AI/ML skills.

Success Metrics

- Deploy first agent: < 2 minutes (UI $\rightarrow$ Live in K8s).
- Agent uptime: 99.8% SLA with auto-failover.
- Cost visibility: Real-time per-agent cost breakdown (compute, model calls, storage).
- Community: 500+ GitHub stars, 50+ public agent templates, 10k+ deployed agents.

Page 2: Product Overview & Features

Core Features (MVP Phase 1: Weeks 1–3)

1.1 Visual Agent Designer

- **Canvas**: React Flow-based graph editor for building agent workflows.
- **Node Types**:
  - **Input**: File upload, API webhook, Kafka stream, database query.
  - **LLM**: Choose model (Ollama local, Mistral, Claude, GPT-4 via API keys).
  - **Tool**: Code executor, REST API caller, database connector, file writer.
  - **Decision**: If/else logic, loop, parallel branches.
  - **Output**: Return JSON, send Slack notification, write to S3, email report.
- **Templates**: Pre-built patterns (PR reviewer, anomaly detector, sentiment analyzer, SQL debugger).
- **Versioning**: Each agent graph saved as version; rollback to previous.

1.2 Multi-Agent Orchestration (LangGraph)

- **Graph Topology**: Supervisor + worker agents (e.g., manager talks to coder + reviewer).
- **State Management**: Persistent agent state across runs (memory, context, decisions).
- **Tool Binding**: Agents can call tools (code execution, API, DB) natively.
- **Error Handling**: Automatic retries with exponential backoff, dead-letter queues.
- **Human-in-the-Loop**: Pause at decision points, allow manual override, log approval.

1.3 Cloud Deployment & GitOps

- **Infrastructure as Code**: Terraform modules for AWS EKS, GCP GKE, DigitalOcean DOKS.

- **Git-Driven**: Push agent YAML → GitHub → ArgoCD auto-syncs to cluster (true GitOps).
- **Helm Charts**: Pre-built charts for AgentForge core (frontend, backend, workers, Redis).
- **Auto-Scaling**: Horizontal Pod Autoscaler rules based on agent queue depth and CPU/memory.

1.4 Observability (OpenTelemetry + Grafana)

- **Trace Collection**:
  - MLflow automatic tracing for LLM calls, tool execution.
  - OpenTelemetry instrumentation in FastAPI backend.
  - Spans: Agent start → LLM call → Tool → Decision → Output.
- **Metrics** (Prometheus):
  - Per-agent: latency (p50, p95, p99), throughput, error rate, token usage, cost.
  - System: pod CPU/memory, request queue depth, database connections.
- **Logs** (Grafana Loki):
  - Structured logs from agents, LLM calls, tool errors; queryable by agent ID, run ID.
- **Dashboards** (Grafana):
  - Real-time agent health, per-model accuracy, cost breakdown by tenant, infrastructure status.
  - Custom: Tenant-specific views, SLA tracking.

1.5 ML Lifecycle (MLflow)

- **Experiment Tracking**:
  - Each agent run logged: parameters (model, temperature), metrics (accuracy, latency), artifacts (output, logs).
- **Model Registry**:
  - Track fine-tuned models per agent; stage (dev/staging/prod); auto-promote on accuracy threshold.
- **Artifact Storage**:
  - S3/GCS backend for datasets, trained models, agent templates.

1.6 Multi-Tenant SaaS Architecture

- **Authentication**: Clerk OAuth (GitHub, Google, email).
- **Organizations**: Create org, invite users, manage roles (admin, developer, viewer).
- **Usage Metering**:
  - Track: API calls, LLM tokens, compute minutes, storage.
  - Per-tenant quotas and alerts.
- **Billing-Ready**: Hook for Stripe integration (future); free tier: 10k monthly tokens, 5 agents.

Page 3: Technical Architecture

High-Level System Design

```
                    USER LAYER

  Frontend (Next.js 15 + React)
  - Agent Builder (React Flow)
  - Dashboard (real-time metrics, logs, cost)
  - Templates Gallery

                      HTTPS/WebSocket

                  API GATEWAY LAYER

  FastAPI Backend (Python)
  - Auth (Clerk middleware)
  - Agent CRUD endpoints
  - Real-time WebSocket for run status
  - Webhook handlers (GitHub, Slack, etc.)
```

AGENT CORE	DATA LAYER	INFRA
LangGraph	PostgreSQL	Kubernetes
Orchestration	Redis Cache	Helm Charts
Supervisor	S3/GCS	ArgoCD
Workers	MLflow	Terraform

LLM LAYER	MONITOR	LOGS
Ollama	Prometheus	Loki
Mistral API	Grafana	Fluent-Bit
Bedrock/GPT	OpenTelemetry	ELK-alt

Component Details

3.1 Frontend (Next.js 15 + React 19)

- **Tech Stack**:
  - Framework: Next.js 15 (App Router, Server Components).
  - UI Library: React 19 + Headless UI (Radix).
  - State: TanStack Query (server state) + Zustand (client state).
  - Graph Editor: React Flow for agent canvas.
  - Charting: Recharts for real-time metrics.
  - Styling: Tailwind CSS v4.
- **Key Pages**:
  - `/dashboard`: Overview, recent agents, system health.
  - `/agents`: List, create, edit agents.
  - `/agents/[id]`: Detail page with canvas editor, runs, metrics, logs.
  - `/runs/[runId]`: Trace viewer, span timeline, error details.
  - `/templates`: Marketplace of agent templates.
  - `/settings`: Auth, org, billing, API keys.
- **Real-Time Updates**: WebSocket connection to backend for live run status, logs, metrics.

3.2 Backend (FastAPI + Python)

- **Tech Stack**:
  - Framework: FastAPI (async/await native).
  - Async Job Queue: Celery + Redis.
  - ORM: SQLAlchemy + Alembic for migrations.
  - Auth: Clerk SDK middleware.
  - LLM Orchestration: LangGraph + LLM interface.
  - ML Tracking: MLflow client.
- **API Endpoints** (REST):

```
POST   /api/v1/agents              Create agent
GET    /api/v1/agents              List agents
GET    /api/v1/agents/{id}         Get agent detail
PATCH  /api/v1/agents/{id}         Update agent
DELETE /api/v1/agents/{id}         Delete agent

POST   /api/v1/agents/{id}/run     Trigger agent execution
GET    /api/v1/runs/{runId}        Get run status
GET    /api/v1/runs/{runId}/logs   Stream run logs
GET    /api/v1/runs/{runId}/trace  Get OTEL trace

GET    /api/v1/metrics/agents      Per-agent stats
```

```
GET    /api/v1/metrics/system      System health
POST   /api/v1/webhooks/github     GitHub webhook (for PR agent)
```

- **Background Jobs** (Celery):
  - Execute agent graph.
  - Send notifications (Slack, email).
  - Sync MLflow metrics to Prometheus.
  - Cleanup old runs and logs (retention policy).

3.3 Agent Execution Runtime (LangGraph + Ollama)

- **Multi-Agent Graph**:
  - Each agent = LangGraph node with state schema.
  - Supervisor coordinates tool calls, decisions, retries.
  - Tools: Code execution (E2B sandbox), REST calls (httpx), DB queries (SQLAlchemy).
- **Model Selection**:
  - Local: Ollama with Llama 3.2, Qwen, Mistral (low latency, free).
  - Cloud: OpenAI, Anthropic, Mistral API via API keys.
  - Fallback: If local fails, retry with API key.
- **State Persistence**:
  - Redis for in-flight state (agent context, memory).
  - PostgreSQL for completed run history and artifacts.

3.4 Data Layer

- **PostgreSQL** (Primary):
  - Tables: agents, runs, traces, users, orgs, usage_meters.
  - Indexes: run_id, agent_id, created_at for fast queries.
- **Redis** (Cache + Queue):
  - Cache: Agent definitions, user settings, rate limits.
  - Queue: Celery task queue for agent execution.
  - Pub/Sub: Real-time updates to frontend WebSocket.
- **Object Storage** (S3/GCS):
  - Artifacts: Uploaded files, outputs, fine-tuned models.
  - MLflow backend: Experiment artifacts.
- **MLflow Backend**:
  - Tracking server for logging metrics, params, artifacts.
  - Model registry for versioning and promotion.

3.5 Infrastructure (Kubernetes + Terraform + GitOps)

- **Kubernetes Cluster**:
  - Node pool: 2–10 nodes (auto-scale based on workload).
  - Namespaces: `agentforge` (core), `agents` (per-tenant workloads).
- **Helm Chart**:

```
agentforge:
  frontend: 2 replicas (nginx reverse proxy)
  backend: 3 replicas (FastAPI)
  worker: 5 replicas (Celery agent execution)
  redis: StatefulSet
  postgres: StatefulSet (persistent volume)
```

- **ArgoCD** (GitOps):
  - App: `agentforge` syncs from `main` branch.
  - Push agent YAML → Git → ArgoCD detects → kubectl apply.
- **Terraform Modules**:
  - `aws-eks`: EKS cluster, security groups, IAM roles.
  - `gcp-gke`: GKE cluster, Workload Identity.
  - `core`: RDS, ElastiCache, S3 buckets, IAM.
  - Outputs: kubeconfig, endpoint URLs.

3.6 Observability Stack

- **Metrics (Prometheus)**:
  - Scrape FastAPI `/metrics` endpoint (standard Prometheus format).
  - Celery worker metrics (task queue depth, execution time).
  - Kubernetes metrics (kube-state-metrics).
  - Alert rules: High error rate, SLA breach, cost anomalies.
- **Traces (OpenTelemetry + Grafana Tempo)**:
  - Instrument FastAPI with OTel SDK.
  - Capture LLM calls via MLflow Tracing (OTel compatible).
  - Send to Tempo backend; queryable via Grafana.
- **Logs (Loki + Fluent-Bit)**:
  - Pod logs collected by Fluent-Bit, shipped to Loki.
  - Agent run logs indexed by agent_id, run_id, tenant_id.
- **Dashboards (Grafana)**:
  - Pre-built: Agent health, model performance, cost trends, SLA status.
  - Tenant-specific: Filter by org_id.

Page 4: Feature Details & Workflows

4.1 Agent Builder Workflow

User Journey: 1. Click "New Agent" → choose template or blank canvas. 2. Drag nodes (Input → LLM → Tool → Output) onto canvas. 3. Configure each node: model, temperature, tool permissions, error handlers. 4. Connect nodes with decision logic (if success → Tool, else → Retry). 5. Click "Save & Deploy" → Choose environment (dev/staging/prod) and cloud (AWS/GCP). 6. View

deployment progress in real-time. 7. Test with sample input; see trace, logs, cost.

Example Agent: GitHub PR Reviewer

```
Input Node: GitHub PR Webhook
  ↓
LLM Node: "Review code for security issues"
  Model: Mistral Medium
  Temperature: 0.2 (deterministic)
  ↓
Tool Node: Code Analysis
  - Static analysis (ESLint/mypy)
  - Security scan (Semgrep)
  ↓
Decision Node: Issues found?
    Yes: Tool Node → Create Review Comment (GitHub API)
    No: Output Node → "Approved"
  ↓
Output Node: Post GitHub comment, send Slack notification
```

4.2 Multi-Agent Orchestration Example

Real-World Use Case: Autonomous Research Team

```
Supervisor Agent (Coordinator)
  Researcher Agent (LLM + Web search tool)
   Task: "Fetch latest papers on AGI"
   Output: Markdown summary
  Analyst Agent (LLM + Analysis tool)
   Input: Research summary
   Task: "Extract key insights and trends"
   Output: Structured JSON
  Report Agent (LLM + Writer tool)
   Input: Analysis
   Task: "Write 1-page executive summary"
   Output: PDF report

Final Output: PDF → Send to Slack, save to S3
```

4.3 Deployment & Monitoring Workflow

Step 1: Infrastructure Setup (Terraform)

```
terraform init
terraform plan -var="environment=prod" -var="cloud_provider=aws"
terraform apply
# Output: EKS cluster, RDS endpoint, S3 bucket
```

Step 2: Deploy AgentForge (Helm + ArgoCD)

```
helm install agentforge ./charts/agentforge \
  --namespace agentforge \
  --values values.prod.yaml
kubectl apply -f argocd/agentforge-app.yaml
```

Step 3: User Pushes Agent (GitOps)

```
# In agents repo: agents/pr-reviewer/manifest.yaml
git push
# ArgoCD detects → Creates Kubernetes Deployment for this agent
```

Step 4: Monitor in Real-Time - Grafana dashboard shows: agent uptime, latency (p95 = 800ms), errors/min, cost per run ($0.02). - Click on run → Tempo trace shows: input processing (50ms) → LLM call (500ms) → Tool call (200ms) → Output (50ms). - Loki shows: "PR Review successful. Posted comment to GitHub."

Page 5: Technology Stack & Dependencies

Layer	Technology	Purpose	Cost
Frontend	Next.js 15, React 19, TanStack Query, React Flow, Tailwind CSS v4	Web UI, canvas editor, real-time updates	Free (Open Source)
Backend API	FastAPI, SQLAlchemy, Pydantic, Celery	REST API, business logic, job queue	Free (Open Source)
Agent Runtime	LangGraph, LangChain, Ollama, E2B SDK	Multi-agent orchestration, LLM calls, sandboxing	Free (Open Source) + API fees optional
LLM Models	Llama 3.2 (Ollama), Mistral, Claude, GPT-4	LLM inference (local or API)	Free local / $ API calls
Database	PostgreSQL, Redis	Primary DB, cache, queue	Free (self-hosted) / $50/mo cloud
Storage	S3 / GCS	Artifacts, models, logs	Free tier / Pay-as-you-go
ML Tracking	MLflow	Experiment tracking, model registry	Free (self-hosted)

Layer	Technology	Purpose	Cost
Observability	Prometheus, Grafana, Loki, Tempo, OpenTelemetry	Metrics, dashboards, traces, logs	Free (self-hosted) / $10/mo cloud
Infrastructure	Kubernetes (EKS/GKE/DOKS), Terraform, Helm, ArgoCD	Container orchestration, IaC, GitOps	Free (self-hosted) / $50–200/mo cloud
Auth	Clerk	User authentication, SSO	Free tier (up to 10k MAU) / $25/mo
Notifications	Slack API, SendGrid	Alerts, messages, emails	Free tier / $10–50/mo

---

Page 6: Project Folder Structure

```
agentforge/
  frontend/                        # Next.js 15 frontend
      app/
          layout.tsx
          dashboard/
              page.tsx
          agents/
              page.tsx             # List agents
              [id]/                # Agent detail + canvas
                  page.tsx
              new/
                  page.tsx         # Create agent
          runs/
              [runId]/
                  page.tsx         # Run detail + trace viewer
      components/
          AgentCanvas.tsx          # React Flow canvas
          DashboardMetrics.tsx     # Real-time charts
          TraceViewer.tsx          # Trace timeline
      lib/
          api.ts                   # API client (React Query hooks)
          auth.ts                  # Clerk integration
          types.ts                 # TypeScript types
      package.json
```

```
    next.config.js

backend/                          # FastAPI backend
    app/
        main.py                   # FastAPI app
        api/
            agents.py             # Agent CRUD endpoints
            runs.py               # Run execution endpoints
            metrics.py            # Metrics endpoints
        models/
            agent.py              # SQLAlchemy Agent model
            run.py                # Run model
            user.py               # User model
        services/
            agent_service.py      # Agent business logic
            langgraph_engine.py   # LangGraph orchestration
            mlflow_logger.py      # MLflow integration
            otel_tracer.py        # OpenTelemetry instrumentation
        workers/
            celery_app.py         # Celery config
            execute_agent.py      # Agent execution task
        middleware/
             auth.py              # Clerk middleware
             otel.py              # OpenTelemetry middleware
    requirements.txt
    Dockerfile
    alembic/                      # Database migrations
        versions/

agents/                           # Example agent templates
    pr-reviewer/
        agent.yaml               # Agent definition
        graph.py                 # LangGraph definition
        tools.py                 # Custom tools
    anomaly-detector/
        ...
    sentiment-analyzer/
         ...

k8s/                              # Kubernetes manifests
    namespace.yaml
    deployment.yaml              # Core deployment
    service.yaml
    configmap.yaml               # Config
    secret.yaml                  # Secrets (managed by Sealed Secrets)
    ingress.yaml                 # Nginx ingress
```

```
helm/                           # Helm chart
    agentforge/
        Chart.yaml
        values.yaml
        templates/
            deployment.yaml
            service.yaml
            configmap.yaml
        values.prod.yaml

terraform/                      # Infrastructure as Code
    main.tf
    aws_eks.tf                  # AWS EKS
    gcp_gke.tf                  # GCP GKE
    variables.tf
    outputs.tf

observability/                  # Monitoring configs
    prometheus/
        prometheus.yaml
        rules.yaml              # Alert rules
    grafana/
        datasources.yaml
        dashboards/
            agents-health.json
            cost-breakdown.json
            sla-tracking.json
    loki/
        loki-config.yaml
    otel-collector/
        config.yaml

docker-compose.yaml             # Local dev stack
docs/
    ARCHITECTURE.md
    API.md
    DEPLOYMENT.md
    CONTRIBUTING.md
    USER_GUIDE.md
README.md
LICENSE                         # MIT / Apache 2.0
.github/
    workflows/
        ci.yaml                 # Test on PR
        build.yaml              # Build images
```

```
deploy.yaml              # Deploy to staging/prod
```

Page 7: Development Roadmap & MVP Timeline

Phase 1: Core MVP (Weeks 1–3) — Single-Agent, Local Deployment

Week 1: Foundation - [ ] Next.js frontend scaffold with auth (Clerk integration). - [ ] FastAPI backend with basic agent CRUD endpoints. - [ ] Simple React Flow canvas (no advanced logic yet). - [ ] Local PostgreSQL + Redis via docker-compose.

Week 2: Agent Execution - [ ] Integrate LangGraph for single-agent orchestration. - [ ] Connect to Ollama (local Llama 3.2 model). - [ ] Build agent execution task (Celery). - [ ] Real-time WebSocket for run status updates.

Week 3: Observability & Testing - [ ] MLflow integration: log run metrics. - [ ] Basic Prometheus metrics from FastAPI. - [ ] Grafana dashboard: agent latency, success rate, token usage. - [ ] E2E tests for agent creation → execution → results.

MVP Deliverables: - User can create agent (canvas → LLM node → execute locally). - See results + metrics in dashboard. - GitHub repo with README, local setup guide. - Demo video: "Create sentiment analyzer in 2 minutes."

Phase 2: Multi-Agent & Cloud (Weeks 4–6)

Week 4: Multi-Agent Orchestration - [ ] LangGraph multi-agent patterns (supervisor, swarm). - [ ] Tool binding (code execution, API calls, DB queries). - [ ] Human-in-the-loop approval workflows.

Week 5: Cloud Deployment (AWS/GCP) - [ ] Terraform modules for EKS, GKE, networking, IAM. - [ ] Helm chart packaging for agentforge. - [ ] ArgoCD setup for GitOps sync.

Week 6: Testing & Documentation - [ ] Integration tests with K8s. - [ ] Architecture docs, deployment guide. - [ ] Public demo: "Deploy agent to EKS in 3 clicks."

Deliverables: - Multi-agent tutorial (supervisor + workers). - Terraform + Helm ready for prod deployment. - Example agents: PR reviewer, anomaly detector.

Phase 3: SaaS Features & Scale (Weeks 7–9)

Week 7: Multi-Tenant & Billing - [ ] Org management, role-based access (RBAC). - [ ] Usage metering (tokens, compute minutes, storage). - [ ] Free tier limits (10k tokens/month, 5 agents). - [ ] Stripe integration for paid tiers (beta).

Week 8: Advanced Observability - [ ] OpenTelemetry full instrumentation (backend, frontend, agents). - [ ] Loki log aggregation with structured logging. - [ ] Tempo trace backend for request tracing. - [ ] Grafana dashboards: per-tenant cost, SLA, model performance.

Week 9: Community & Polish - [ ] Agent template marketplace (submit via PR). - [ ] Public API rate limits, API key management. - [ ] Community docs, video tutorials, Discord. - [ ] Refinement based on early user feedback.

Deliverables: - Production-ready SaaS platform. - 10+ agent templates (PR reviewer, SQL debugger, content generator, etc.). - Community website + docs.

Phase 4: Advanced Features (Weeks 10+)

- **A/B Testing Agents**: Route % of traffic to new agent version; auto-promote on metrics.
- **RLHF Loop**: Thumbs-up/down feedback; compute reward model; re-train.
- **Fine-Tuning Pipeline**: Upload data → auto-fine-tune Llama/Mistral → deploy.
- **Marketplace**: List public agents, monetize (revenue share).
- **IDE Plugin**: VSCode extension to run agents from editor.

Page 8: Key Success Metrics & KPIs

Metric	Target	Owner
User Adoption	100 sign-ups by month 2	Product
Agent Deploys	1,000 agents deployed by month 3	Community
Uptime SLA	99.8%	DevOps/SRE
Avg Deploy Time	$<$ 2 min (UI → Live in K8s)	Engineering
Cost per Agent Run	$<$ \$0.05 (compute + model)	Finance/Ops
GitHub Stars	500+ by end of year 1	Community/Marketing
Documentation Completeness	95% code coverage in docs	Docs Team

Metric	Target	Owner
Community PRs	50+ contributions by month 4	Community
Latency (p95)	$<$ 1s for agent execution	Backend
Error Rate	$<$ 0.1% for deployed agents	QA/Monitoring

Page 9: Security, Compliance & Risk Mitigation

9.1 Security Measures

- **Auth**: Clerk with SSO (SAML 2.0 for enterprise).
- **Data**: Encryption at rest (AES-256), in transit (TLS 1.3).
- **API**: Rate limiting, API key rotation, audit logs.
- **Sandbox**: E2B for code execution (isolated, immutable).
- **Secrets**: Sealed Secrets in K8s, HashiCorp Vault for prod.

9.2 Compliance

- **GDPR**: User data export, right to deletion, privacy policy.
- **SOC 2 Type II**: Target for enterprise tier (year 2).
- **Audit Logs**: All agent executions, API calls, user actions logged.

9.3 Risk Mitigation

- **Model Jailbreaks**: Use tempering, input validation, guardrails (via LLM agent).
- **Cost Runaway**: Per-tenant quotas, spend alerts, automatic suspension.
- **Data Leakage**: Egress filtering, no logs to unauthorized storage.

Page 10: Monetization & Business Model

10.1 Revenue Streams

1. **Free Tier**: 10k tokens/month, 5 agents, community support.
2. **Pro Tier** ($29/mo): 1M tokens/month, 50 agents, email support, custom domain.
3. **Enterprise** (custom): Unlimited agents, dedicated support, SLA, on-prem option.
4. **Marketplace Commission** (5–10%): Revenue share on paid agent templates.

10.2 Positioning

- **For Startups**: "Deploy AI prototypes to production in hours, not months."
- **For Enterprises**: "Unified governance, audit, cost control for all your AI agents."
- **For Developers**: "Build, share, and monetize AI agents; showcase full-stack skills."

Page 11: Competition & Differentiation

Competitor	Strengths	Weaknesses	AgentForge Edge
LangChain + Supabase	Good LLM APIs	DIY infra, no UI	Integrated cloud + UI builder
Crew AI	Multi-agent focus	Python-only, no viz	Visual builder, K8s-ready
OpenDevin	Code agent	Single agent, limited tools	Multi-agent, full observability
Zapier + Make	Visual builder	Not ML-focused	ML-first, agent-native

AgentForge USP: - "Cloud-native from day 1" (Terraform + ArgoCD built-in). - "Full observability by default" (OpenTelemetry + Grafana integrated). - "Open-source + SaaS" (run locally or on cloud, no lock-in). - "Multi-agent native" (LangGraph from the ground up).

Page 12: Getting Started & Quick Reference

12.1 Local Development (docker-compose)

```
git clone https://github.com/AdityaShenvi/AgentForge.git
cd AgentForge
docker-compose up
# Frontend: http://localhost:3000
# Backend API: http://localhost:8000
# Grafana: http://localhost:3001
```

12.2 Deployment (One Command)

```
terraform init
terraform apply -var="environment=prod" -var="cloud_provider=aws"
helm install agentforge ./helm/agentforge --namespace agentforge
# EKS cluster + agentforge ready in ~10 minutes
```

12.3 Key Links

- **GitHub**: https://github.com/AdityaShenvi/AgentForge
- **Docs**: https://docs.agentforge.dev
- **Discord**: https://discord.gg/agentforge
- **Demo**: https://demo.agentforge.dev

12.4 Key Team Roles (Open Contributions)

- **Backend Engineer**: LangGraph, FastAPI, async tasks.
- **Frontend Engineer**: Next.js, React Flow, real-time updates.
- **DevOps Engineer**: Kubernetes, Terraform, GitOps, observability.
- **ML Engineer**: Fine-tuning, agent optimization, benchmarking.
- **Community Manager**: Docs, tutorials, templates, Discord.

Page 13: Conclusion & Vision Statement

AgentForge is positioned as the **"GitHub for AI Agents"**—democratizing AI deployment just as GitHub democratized code. By combining visual simplicity with production-grade infrastructure, AgentForge enables developers, startups, and enterprises to build, share, and scale intelligent agents without the complexity.

Vision (2026): - 10k+ deployed agents across AWS, GCP, Azure. - 500+ GitHub stars, 100+ community contributors. - Enterprise customers (Fortune 500) using AgentForge for mission-critical agents. - Marketplace with 1k+ agent templates, $5M+ in cumulative creator revenue.

2025–26 Milestones: - MVP (local agent builder + deploy to K8s): Dec 2025. - SaaS platform live with 100+ users: Q1 2026. - Enterprise tier + on-prem option: Q2 2026. - IPO-ready / Series A fundraising: Q3 2026.

Project Coordinator: Aditya Shenvi
Portfolio: https://www.adityacuz.dev
Contact: aditya@agentforge.dev
GitHub: https://github.com/AdityaShenvi

© 2025–26 Aditya Shenvi. All rights reserved. AgentForge is released under the MIT License.

Document Version: 1.0 | **Last Updated:** December 6, 2025 | **Status:** Production-Ready Specification

---