Extream.AI – Environment Setup & Directory Guide

1. 📁 Base Project Directory Structure

```
extream-ai/
├─ backend/              # FastAPI APIs + LLM adapters
│   ├─ requirements.txt
│   ├─ app/
│   └─ tests/
├─ frontend/             # React.js no-code builder console
│   ├─ package.json
│   └─ src/
├─ ingestion/            # Data ingestion services (HTTP/gRPC, Kafka)
│   ├─ configs/
│   └─ services/
├─ agents/               # Agent orchestration + MicroBOTs
│   ├─ state_machines/
│   └─ bots/
├─ governance/           # Policy packs + blockchain audit ledger
│   ├─ opa/
│   ├─ rules/
│   └─ dashboard/
├─ eap/                  # Extreme Automation Protocol (codecs, routing)
│   ├─ codecs/
│   └─ adapters/
├─ infra/                # Infra-as-code + CI/CD
│   ├─ docker/
│   ├─ k8s/
│   └─ terraform/
├─ monitoring/           # Prometheus, Grafana dashboards
├─ data/                 # Sample ingestion payloads
├─ scripts/              # Setup, run, test scripts
├─ docs/                 # Technical + mentor guides
└─ README.md
```

---

2. 🖥️ System Requirements

Windows: Windows 10/11 Pro (with WSL2 recommended)
macOS: macOS 12+ (Intel or Apple Silicon)

- RAM: 16GB+
- Disk: 50GB+ free

Tools (common):

- Git
- Python 3.10+

- Node.js 18+ & npm/yarn
- Docker Desktop
- Kubernetes (kubectl, Minikube or Kind)
- Terraform (latest stable)
- Helm
- PostgreSQL (via Docker)
- VSCode + extensions (Python, Docker, Kubernetes, YAML)

3. ⚙️ Installation Scripts

A) Windows (PowerShell / WSL2)

```
# Update system
winget upgrade --all

# Install Git
winget install --id Git.Git -e --source winget

# Install Python
winget install Python.Python.3.11

# Create venv
python -m venv venv
.\venv\Scripts\activate
pip install --upgrade pip

# Install Node.js + npm
winget install OpenJS.NodeJS.LTS
npm install -g yarn

# Install Docker Desktop
winget install Docker.DockerDesktop

# Enable WSL2 + Kubernetes
wsl --install -d Ubuntu
wsl --set-default-version 2

# Install kubectl
winget install Kubernetes.kubectl

# Install Minikube
winget install Kubernetes.minikube

# Install Terraform + Helm
winget install HashiCorp.Terraform
winget install Helm.Helm

# Verify installs
python --version
node -v
npm -v
```

```
docker --version
kubectl version --client
terraform -v
helm version
```

B) macOS (zsh/bash)

```
# Install Homebrew (if not installed)
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

# Update packages
brew update

# Install Git
brew install git

# Install Python
brew install python@3.11
python3 -m venv venv
source venv/bin/activate
pip install --upgrade pip

# Install Node.js + npm
brew install node
npm install -g yarn

# Install Docker Desktop
brew install --cask docker

# Install Kubernetes tools
brew install kubectl
brew install minikube

# Install Terraform + Helm
brew tap hashicorp/tap
brew install hashicorp/tap/terraform
brew install helm

# Verify installs
python3 --version
node -v
npm -v
docker --version
kubectl version --client
terraform -v
helm version
```

4. 📜 Environment Setup Scripts

Inside `scripts/` add:

scripts/setup_backend.sh

```
#!/bin/bash
# Setup Python backend
cd backend
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

scripts/setup_frontend.sh

```
#!/bin/bash
# Setup React frontend
cd frontend
yarn install
```

scripts/setup_ingestion.sh

```
#!/bin/bash
# Setup Ingestion Layer (Kafka, services)
cd ingestion
docker-compose up -d
```

scripts/setup_agents.sh

```
#!/bin/bash
# Deploy sample agents & MicroBOTs
cd agents
python deploy_agents.py
```

scripts/dev_env.sh

```
#!/bin/bash
# Run all core services via Docker Compose
docker-compose -f infra/docker/docker-compose.yml up --build
```

5. 🧪 Quick Smoke Test

Backend

```
curl http://localhost:8000/health
# Expected → { "status": "ok" }
```

Frontend

- Open `http://localhost:3000`
- Expected → React no-code builder splash screen

Ingestion

```
curl -X POST http://localhost:8080/ingest -d '{"msg":"hello"}'
# Expected → ACK response + record in Kafka
```

Agents

```
python agents/tests/run_agent.py --agent AppOps --event "scale up"
# Expected → Agent responds + logs action
```

Governance

```
curl http://localhost:9000/policies
# Expected → JSON list of SOC2-lite rules
```

Infra

```
kubectl get pods
# Expected → Cluster up, pods running
```

✅ With this, every new developer can:
Clone repo → run `scripts/dev_env.sh` → have backend, frontend, ingestion, agents, governance, and infra stack up in <30 min.