BRAUDE
College of Engineering, karmiel

Capstone Project – Phase A (61998)

Cryptopus

Reinforcement Learning-Based Meta-Strategy for Algorithmic Bitcoin Trading

Project Code: 26-1-D-29

By: Omer Lev & Matan Reuven Tal

Advisor: Dr. Reuven Cohen

January 2026

Algorithmic Trading in Financial Markets

Algorithmic trading uses computer programs and mathematical models to automatically analyze market data, apply predefined rules or learned policies and execute trades in financial markets.

Market Adoption & Scale

- Over the past two decades, algorithmic trading has become dominant due to advances in computing power, data availability and automation.
- The global algorithmic trading market is estimated at USD 18.7B (2025) and projected to reach **USD 28.4B by 2030.**

Key advantages

- Real-time processing of large data volumes
- High-speed and precise trade execution
- Emotion-free, consistent decision-making

Bitcoin and Cryptocurrency Markets

Bitcoin is the first and most widely adopted cryptocurrency. Key characteristics:

1

Continuous Trading

Operates 24/7 without market closures

2

High Liquidity

Active trading across major global exchanges

3

Market Benchmark

Reference asset for most crypto trading pairs

4

Data Accessibility

Open access to price, volume, and order data via APIs

5

Transparency

Public and data-rich market structure

Core Challenges in Algorithmic Trading

Despite the advantages of algorithmic trading, several fundamental challenges limit its effectiveness and adoption.

1. Strategy Sensitivity to Market Conditions
2. Conflicting Signals in Multi-Strategy Systems
3. Limited Access to Advanced Trading Infrastructure
4. High Knowledge Barriers

Strategy Sensitivity to Market Conditions

Condition-Dependent Performance

Strategy effectiveness depends on the current market regime

Performance Degradation

Strategy performance declines as regimes change

Market Regime Shifts

Market transitions between trending, sideways and volatile states

Conflicting Signals in Multi-Strategy Systems

Use of Multiple Strategies

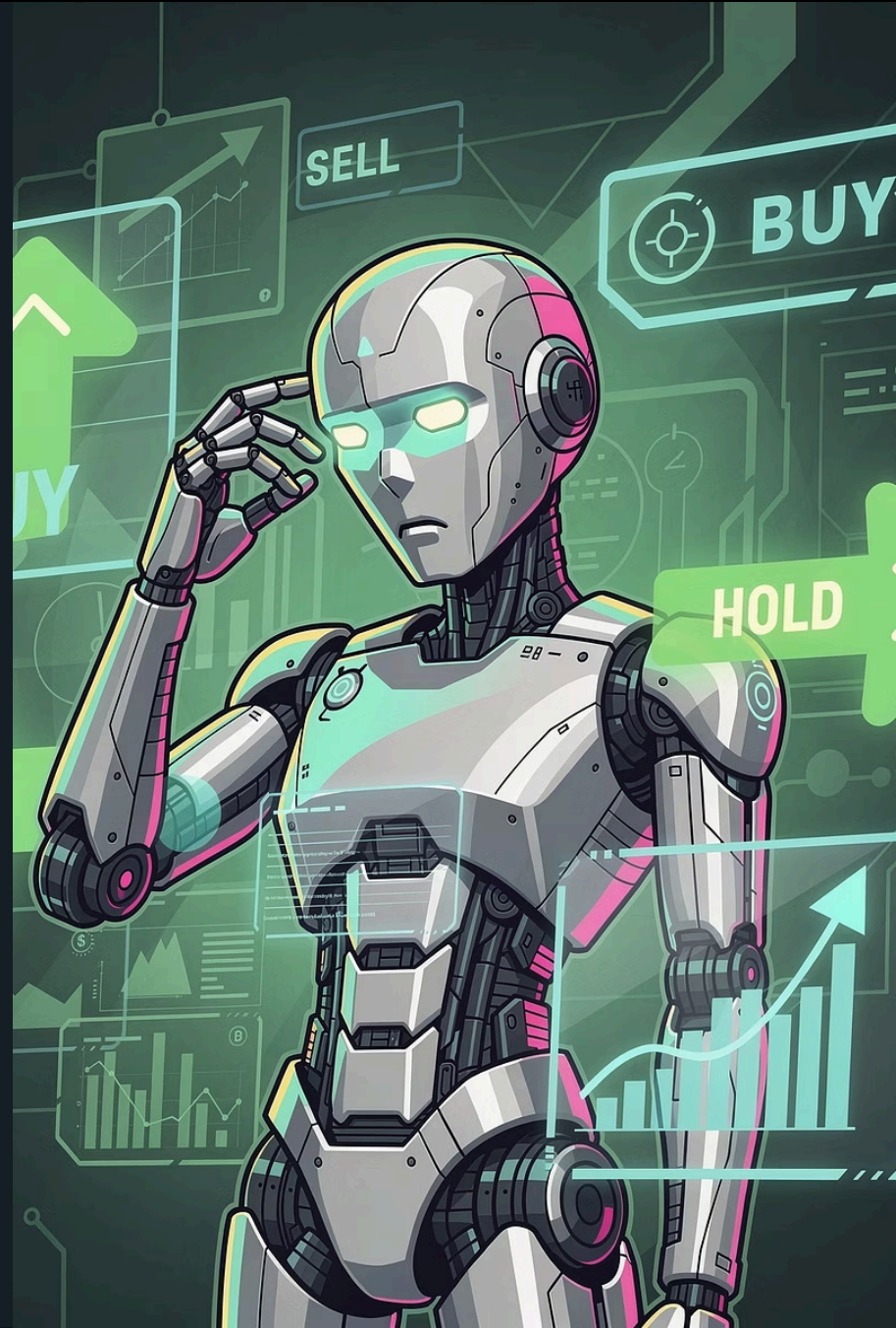
Traders combine several strategies to handle changing market conditions

Conflicting Signals

Different strategies may suggest buy, sell or hold simultaneously

Decision Complexity

Resolving conflicts without a clear rule becomes inconsistent



Limited Access to Advanced Trading Infrastructure

1

High Technical Requirements

Advanced systems require significant computing power and infrastructure

2

Institutional Advantage

Sophisticated tools are mainly available to hedge funds and institutions

High Knowledge Barriers

1

Technical Expertise Required

Programming, data analysis and software engineering skills

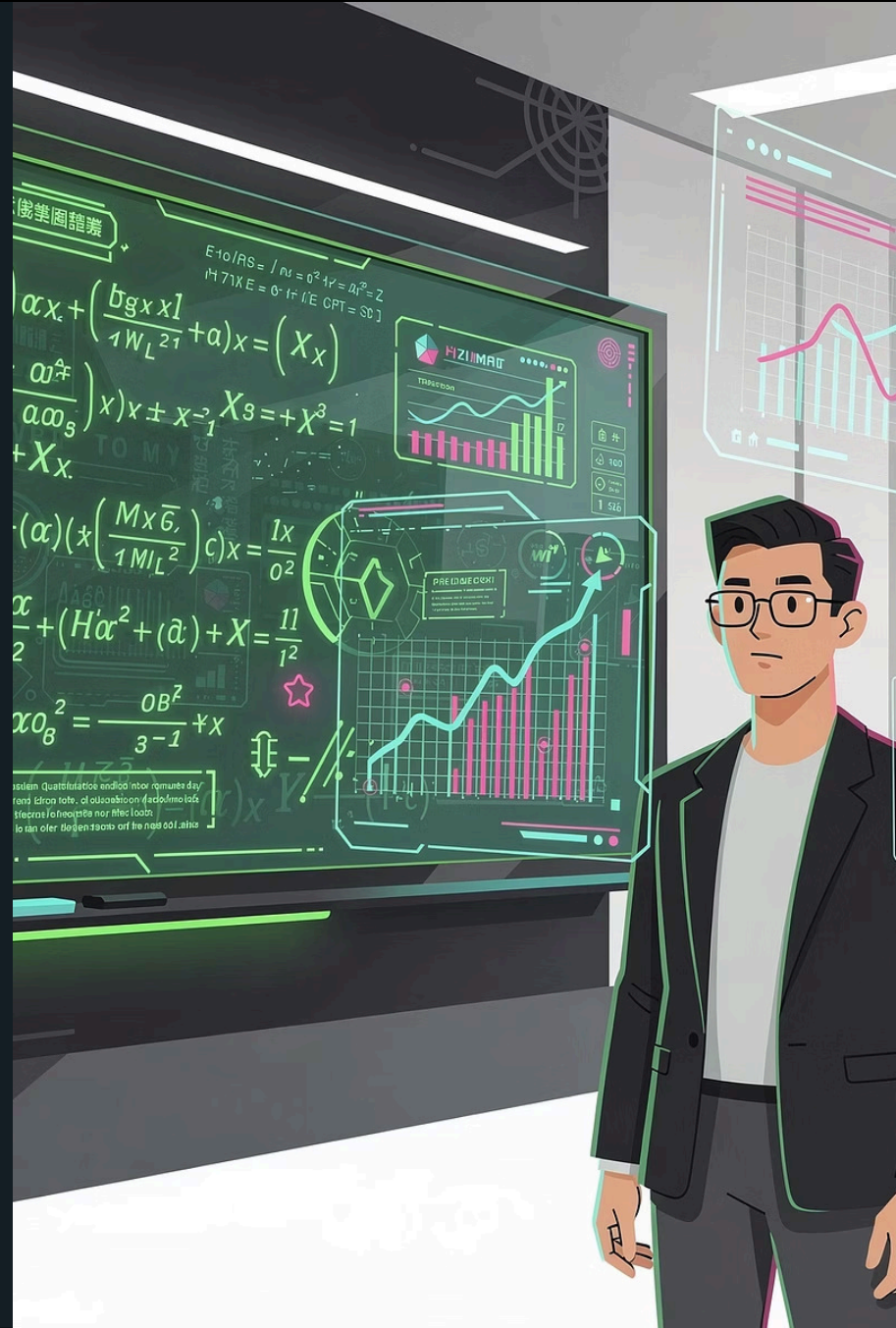
2

Financial Domain Knowledge

Understanding markets, risk management and trading mechanics

Combined Skill Requirement

Effective systems require both technical and financial expertise



Existing Solutions and Limitations

Several automated trading solutions are used in Bitcoin markets. However, none of them fully address the core challenges identified earlier.

Solution Type	Key Characteristics & Limitations	Examples
Exchange-Native Trading Bots	Easy to use and accessible, but based on fixed strategies and manual adjustments	Binance Trading Bots, Pionex
Third-Party Automated Platforms	Reduce operational overhead, yet rely on static rules and lack adaptive decision-making	3Commas, Coinrule, Stoic AI
Script-Based Trading Systems	Offer flexibility and control, but require significant technical expertise and manual maintenance	TradingView (Pine Script)

LITERATURE REVIEW

Reinforcement Learning for Adaptive Decision-Making

These challenges point to the need for a decision-making mechanism that can:

1. Operate under non-stationary conditions
2. Integrate multiple sources of information
3. Continuously adjust its behavior

Reinforcement Learning (RL)

Formulates trading as a sequential decision-making problem and enables adaptive policy learning in dynamic and uncertain environments.

Empirical Evidence for RL Performance

Comparison between DDPG (RL), Buy-and-Hold (DJIA) and Minimum-Variance

Source: Liu et al., "Practical Deep Reinforcement Learning Approach for Stock Trading", arXiv:1811.07522 (2018)

- RL-based strategy achieves **higher cumulative portfolio value**
- Outperformance is sustained over extended time horizon
- Demonstrates RL's ability to **adapt to changing market conditions**

LITERATURE REVIEW

Reinforcement Learning Overview

- Reinforcement Learning (RL) learns **how to act through interaction** with an environment
- An **agent** observes a state, selects an action, and receives a **reward**
- Decisions are **sequential and closed-loop**: actions affect future states
- The agent learns a **policy** that maximizes **long-term cumulative reward**
- Learning is driven by **trial and error**, not explicit instructions
- A key challenge is balancing **exploration vs. exploitation**
- Problems are commonly formulated as a **Markov Decision Process (MDP)**

Proposed Solution: Cryptopus

Cryptopus is a reinforcement learning-based meta-strategy for algorithmic Bitcoin trading. It dynamically combines and adapts multiple trading strategies according to prevailing market conditions. The solution is implemented as a multi-component system consisting of four main parts:

1

RL Training and Evaluation Project

Responsible for producing a trained trading agent using historical data

2

Cloud-Deployed Server-Side

Operates continuously, where the trained agent interacts with the exchange in real time

3

Strategy Conversion Component

Adapts successful open-source trading strategies into a standardized, system-compatible format

4

Client-Side Desktop Application

Enables monitoring, configuration, and control of the platform

PROPOSED SOLUTION

Sub-Project I: RL Training and Evaluation

This component is responsible for training, evaluating and backtesting reinforcement learning agents for Bitcoin trading.

- **Key Characteristics:** Separated from live trading, not user-exposed, and operated via a developer-oriented Command-Line Interface (CLI) for controlled execution.
- **Data Used:**
  - Historical Bitcoin OHLCV data
  - Technical indicators
  - Risk-related features
  - Outputs from multiple trading strategies
- **Simulation Environment:** A custom trading simulation models realistic portfolio dynamics.

PROPOSED SOLUTION

Sub-Project I: RL Training and Evaluation

Model training utilizes policy-gradient reinforcement learning algorithms within a configurable framework. A centralized configuration allows developers to adjust parameters that directly influence the learned policy.

Outputs:

- Trained reinforcement learning model
- Metadata files detailing the training configuration
- Comprehensive backtesting results including:
  - Step-level and trade-level logs
  - Quantitative performance metrics
  - Visual comparisons against baseline strategies (e.g., buy-and-hold)

PROPOSED SOLUTION

Sub-Project II: Client-Server Trading Execution System

- Executes the **trained RL agent** in a real-time trading environment
- Designed as a **client-server architecture**

Server Side

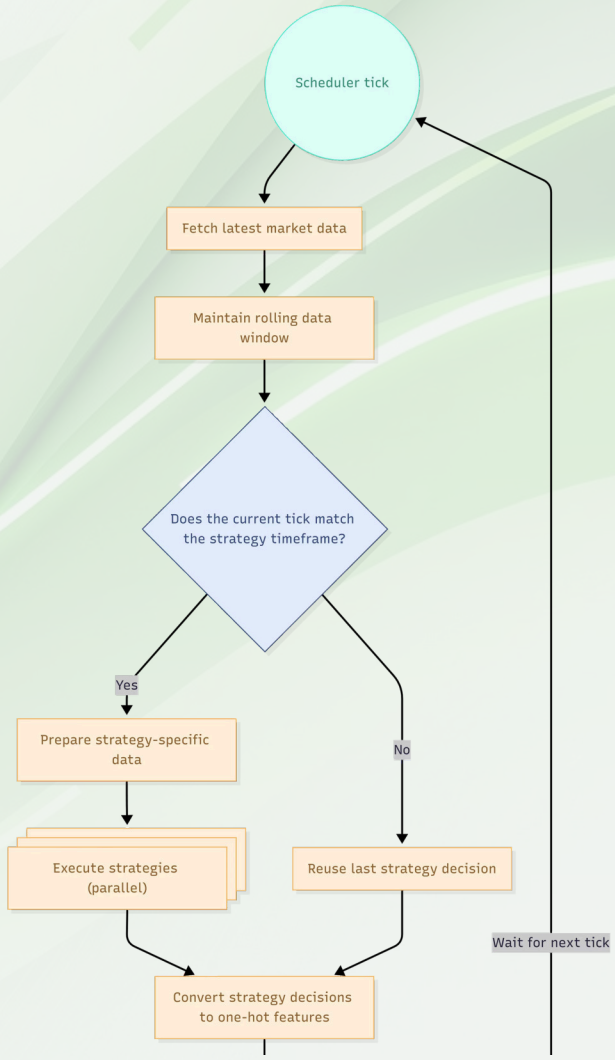
- Cloud-deployed (AWS) for 24/7 operation and scalability
- Executes the selected trained RL agent in real time
- Connects directly to the **Kraken exchange**
- Manages backend components: users, models, trades, market data and logs

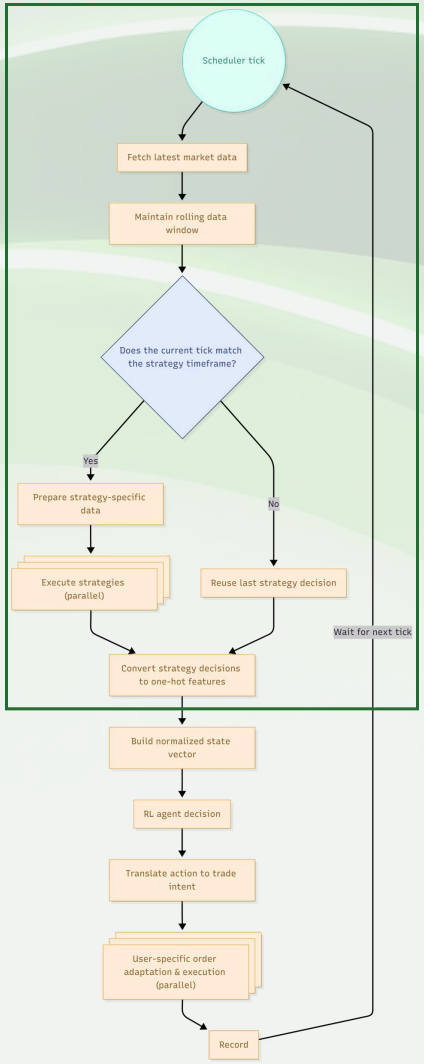
Client Side

- Desktop GUI for monitoring, management, and interaction
- Supports **end users** and **administrators**
- End users allocate funds and control trading activation only
- Administrators manage trained agents and system configuration

Continuous Execution Mechanism

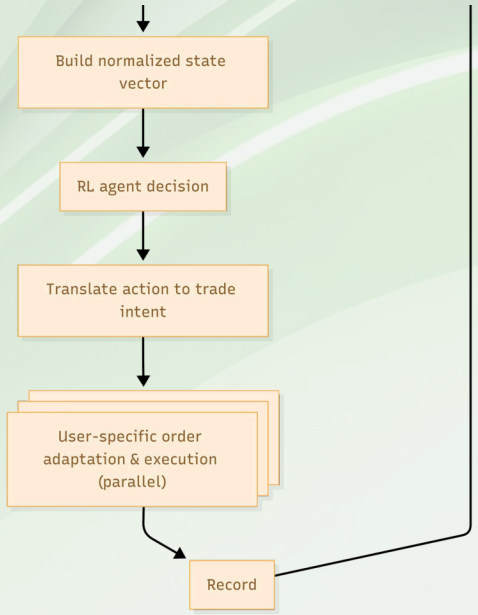
Fetch market data & execute strategies

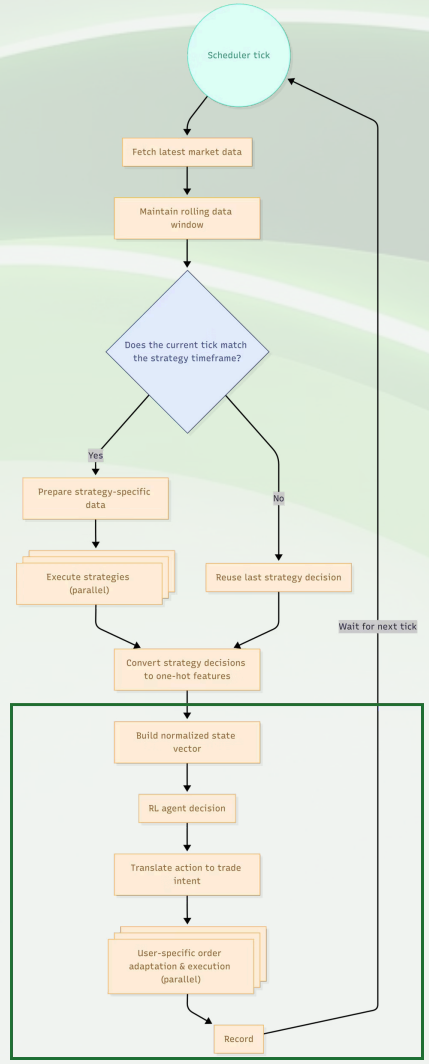




Continuous Execution Mechanism

RL-Based Decision & Trade Execution





PROPOSED SOLUTION

Sub-Project III: PineScript to Python Strategy Converter

- Addresses integration of **existing open-source trading strategies**
- Focuses on converting **TradingView Pine Script** strategies to Python
- Enables compatibility with the RL training and execution infrastructure
- Conversion performed via a **controlled AI agent–based workflow using Claude Code**
- Operates on a strategy-by-strategy basis with human approval
- Failed validation or testing halts the process for developer review
- Successfully converted strategies are **versioned and deployed** in a controlled manner

PROPOSED SOLUTION

Example Strategy: Moving Average Crossover

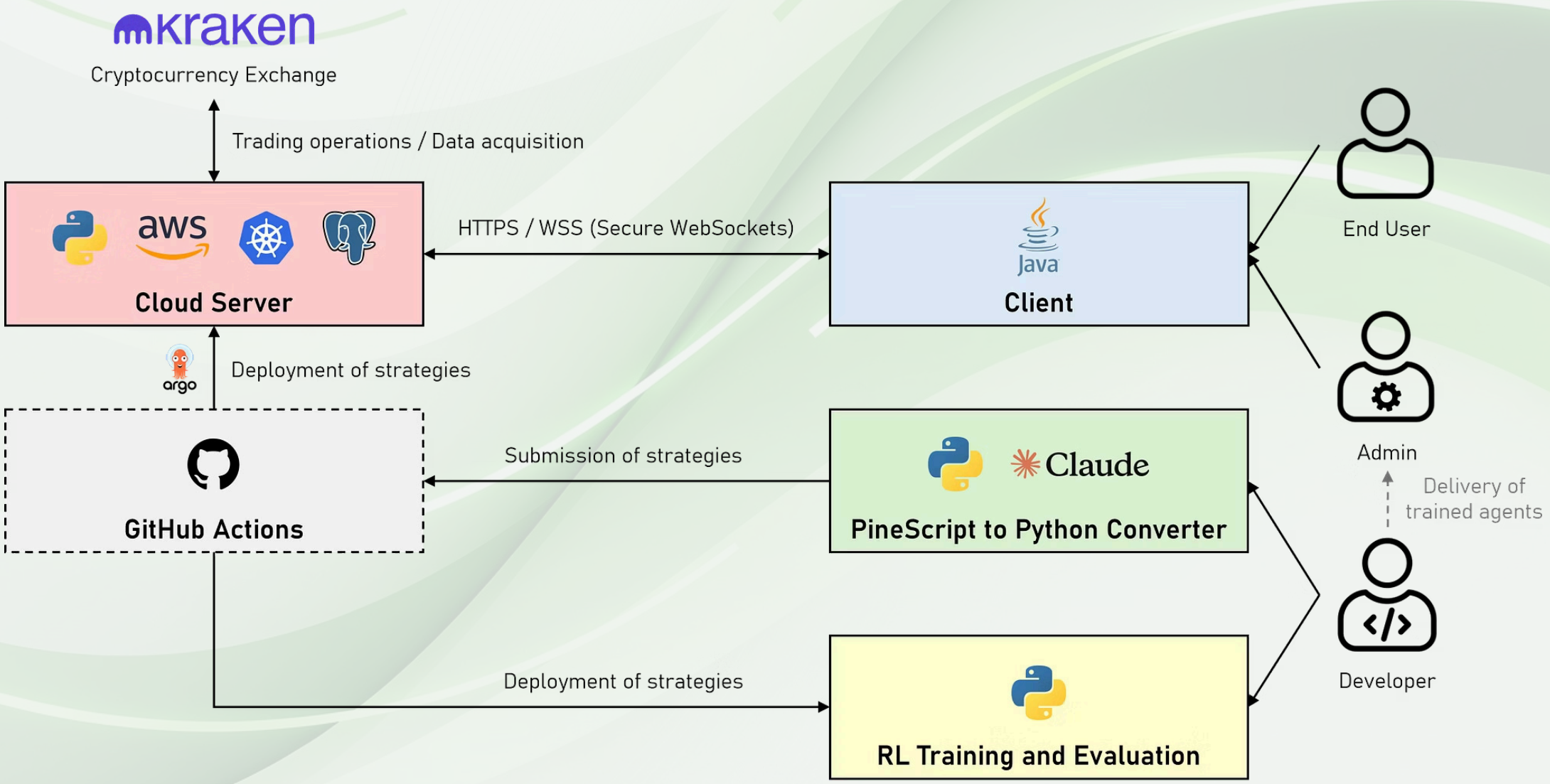
A simple trend-following strategy that detects market direction using two moving averages with different time horizons.

Moving Average- a smoothed price series representing the average of past prices over time.

- Short-Term Moving Average (e.g., MA 20)- **Blue**
- Long-Term Moving Average (e.g., MA 50)- **Green**

Rationale: The crossover reflects a shift in market momentum, indicating a potential trend change.

Integrated System Architecture


kraken
Cryptocurrency Exchange
Trading operations / Data acquisition
aws
Cloud Server
HTTPS / WSS (Secure WebSockets)
Java
Client
End User
argo
Deployment of strategies
Admin
Submission of strategies
Claude
PineScript to Python Converter
Delivery of trained agents
GitHub Actions
Deployment of strategies
RL Training and Evaluation
Developer

Testing Process and Validation

- Testing integrated throughout development following **Agile principles**
- Emphasis on automated unit and integration tests for core logic
- Manual acceptance testing applied to user-facing interfaces
- **Continuous Integration:** Automated verification on every commit (GitHub Actions)

Sub-Project-Specific Validation

RL Training

- Verification of deterministic components
- Validation via historical backtesting and comparison to baselines

Client-Server System

- Automated backend and database testing
- Manual UI acceptance testing
- Security verification for authentication and credential handling

Strategy Conversion

- Automated syntax and interface validation
- Logical validation via sanity checks and visual inspection

Expected Challenges

1

Budget Constraints

Mitigation:

Use student credits, open-source tools and local model training on personal machines to minimize cloud costs

2

Trained Model Quality

Mitigation:

Externalize all training configurations, enabling rapid iterative experimentation with different parameter settings until robust and stable model performance is achieved

3

Trading Strategy Discovery

Mitigation:

Leverage the extensive TradingView strategy repository and integrate selected strategies into the system using an automated PineScript-to-Python conversion

4

Latency and Execution Delays

Mitigation:

Utilize parallel execution and multi-core processing to run trading strategies concurrently, reducing decision latency

Thank you for listening!