### **Milestone 1: The Foundation – Data Pipeline and Core Technical Analysis**

The goal of this first phase is to build the absolute core of the application: the ability to connect to Coinbase, retrieve market data, and run a single, classic trading strategy on it. This milestone validates that your fundamental data connection and analysis engine are working correctly.

**What you will achieve:** A Python script that can fetch historical price data for a single cryptocurrency (e.g., BTC-USD) from Coinbase and print historical buy/sell signals to the console based on a Moving Average Crossover strategy.

**Steps to Complete:**

1. **Environment Setup:** Create an isolated Python environment (venv) for your project. Install the essential libraries: coinbase-advanced-py for the API connection, pandas for data manipulation, and pandas-ta for technical indicators.
2. **Data Connector Module:** Write a dedicated Python module to handle the connection to the Coinbase API. This module should have a function that takes a cryptocurrency pair (e.g., 'BTC-USD') and a timeframe (e.g., '1h') as input and returns a clean pandas DataFrame with historical Open, High, Low, Close, and Volume (OHLCV) data.
3. **Implement a Simple Strategy:** In your main script, use the data connector to fetch data. Then, calculate two Simple Moving Averages (SMAs), a short-term one (e.g., 50-period) and a long-term one (e.g., 200-period).
4. **Generate Historical Signals:** Write logic to identify where the short-term SMA crosses above the long-term SMA ("Golden Cross" - a buy signal) and where it crosses below ("Death Cross" - a sell signal).
5. **Console Output:** Loop through your historical data and print a simple message to the console for each crossover event you find, including the date it occurred.

### **Milestone 2: Expanding the Analytical Engine and Adding Risk Management**

With the data pipeline established, this milestone focuses on making the signals more robust by requiring confirmation from multiple indicators. Crucially, it also introduces a professional risk management framework, transforming simple signals into actionable trade plans.

**What you will achieve:** The bot will now generate higher-quality signals based on the confluence of three technical indicators (MA Crossover, RSI, and MACD). For each signal, it will also calculate and display a suggested stop-loss and position size.

**Steps to Complete:**

1. **Integrate More Indicators:** Add the Relative Strength Index (RSI) and the Moving Average Convergence Divergence (MACD) to your analysis script.
2. **Develop Confluence Logic:** Instead of acting on a single indicator, create a multi-condition rule. For example, a "Strong Buy" signal is only triggered when a Golden Cross occurs, the RSI is not in the "overbought" zone (e.g., below 70), and the MACD is bullish.
3. **Introduce Volatility Measurement:** Calculate the Average True Range (ATR), which is a key measure of market volatility.
4. **Build a Risk Module:** Create a new module for risk management. It should contain functions that:
   * Calculate a suggested stop-loss price based on the current price minus a multiple of the ATR (e.g., Entry Price - 2.5 \* ATR).
   * Calculate a suggested position size based on a fixed percentage of a hypothetical portfolio (e.g., risk 1% of total capital).
5. **Enhance Console Output:** Upgrade the notification to provide a full trade plan: STRONG BUY SIGNAL: BTC-USD | Entry: $65,000 | Suggested Stop-Loss: $62,500 (based on 2.5 \* ATR) | Position Size for 1% risk on $100,000 portfolio: $4,000.

### **Milestone 3: Going Live – Automation and Operational Readiness**

This phase transitions the bot from a script that analyzes historical data to an automated application that runs continuously on your VM, monitors the market in near real-time, and logs its activity.

**What you will achieve:** A 24/7 bot that runs on your VM, fetches the latest market data periodically, and logs any generated trade plan notifications to the console and a log file.

**Steps to Complete:**

1. **Create the Main Loop:** Structure your application with a main while True: loop that performs the analysis, prints notifications, and then sleeps for a set interval (e.g., time.sleep(300) for 5 minutes).
2. **Adapt Data Fetching:** Modify your data connector to fetch only the most recent data candles needed to keep your indicators updated, rather than downloading the full history on every loop.
3. **Implement Robust Logging:** Use Python's built-in logging module to create a log file. Record key events such as application start/stop, successful or failed data fetches, and the full details of any signal generated. This is essential for debugging and monitoring.
4. **External Configuration:** Move hardcoded variables like API keys, the list of coins to track, and indicator parameters (e.g., SMA periods) into a separate configuration file (config.py or config.ini) so you can change settings without editing the core code.
5. **Prepare for Deployment:** Finalize your requirements.txt file. Write a simple shell script (e.g., start\_bot.sh) to activate the virtual environment and run your main Python script, making it easy to start on your VM.

### **Milestone 4: Advanced Intelligence – Sentiment and On-Chain Analysis**

This final milestone elevates the bot by incorporating alternative data sources. This provides crucial market context that price and volume data alone cannot capture, leading to more nuanced and potentially predictive signals.

**What you will achieve:** The bot's signal-generation logic will be enhanced to include scores from news sentiment and on-chain data, providing the most comprehensive analysis before issuing a notification.

**Steps to Complete:**

1. **Sentiment Analysis Module:**
   * Integrate a news API (like Crypto News API or Finnhub) to pull recent headlines for your target cryptocurrencies.
   * Use a library like VADER or TextBlob to score the sentiment of these headlines (positive, neutral, negative).
   * Aggregate the scores to create a daily sentiment rating for each coin.
2. **On-Chain Data Module:**
   * Integrate an on-chain data provider's API (like Glassnode or Santiment).
   * Fetch key metrics that act as leading indicators, such as **Exchange Net Flow** (large outflows are bullish) or the growth in **Active Addresses**.
3. **Synthesize All Data:** Upgrade your signal logic from Milestone 2 into a more comprehensive scoring model. A "High Conviction Buy" signal might now require:
   * **Technical Confirmation:** (e.g., Golden Cross + Bullish MACD).
   * **Positive Sentiment:** (e.g., Net sentiment score > 0).
   * **On-Chain Confirmation:** (e.g., Net outflows from exchanges).

By following this phased plan, you will build your application incrementally, ensuring you have a solid, working foundation before adding more complex features.