**Prepare your environment**

1. **Set up your machine**
   * Install **Git** and **GitHub CLI** to manage repositories.
   * Install **Docker Desktop** (Windows/Mac) or Docker Engine (Linux) so you can run containerised scrapers and tools.
   * Install **Node.js** (latest LTS) and **Python 3.10+**; these will be used by some repos.
   * Install **n8n**:
     + On a workstation: run npm install -g n8n or follow the [n8n Docker installation instructions](https://docs.n8n.io/) to host it in a container.
   * Ensure you have a modern browser (Chrome/Edge) for logging into LinkedIn and other sites if required.
2. **Collect your API tokens**
   * **OpenAI API key** – for summarisation and message generation.
   * **LinkedIn tokens** – for any tools requiring LinkedIn authentication (e.g., linkedapi‑node).
   * **GitHub PAT (Personal Access Token)** – to push updates back to the repo.
   * **Any other credentials** (Slack webhook, email SMTP, etc.) for notifications.
   * Store these securely in a .env file or inside n8n’s credential manager; never commit them to Git.
3. **Clone the master repository**

bash

CopyEdit

git clone https://github.com/DirtyDiablo/primetime-agent-mode.git

cd primetime-agent-mode

* + Familiarise yourself with the structure: /data for program folders (aesd, jets, etc.), /docs for configuration files, /scripts for utility scripts.

1. **Create program directories**
   * For each program you’ll track (e.g., AESD, JETS, GBSD), create a subfolder under data/:

bash

CopyEdit

mkdir -p data/aesd data/jets data/gbsd

touch data/aesd/jobs.csv data/aesd/people\_targets.csv data/aesd/competitors.csv

# Repeat for jets and gbsd

1. **Install dependencies for each GitHub repo**
   * For Python tools (fpdsScraper, JobSpy, jobsparser), run pip install -r requirements.txt inside a virtual environment.
   * For Node tools (linkedapi-node, linkedin-message-generator), run npm install in their directories.
   * For any Dockerised tool, create a Dockerfile (or use existing ones) and build images.

**2. Set up core GitHub repositories**

Below is a table summarising the repos and how you’ll deploy each one. Use this as a checklist during setup.

| **Repo/Tool** | **Key purpose** | **Setup steps** |
| --- | --- | --- |
| **apify‑linkedin‑profile‑search‑by‑name** | Scrape LinkedIn profiles (basic or full; with email in “Full + email” mode)[GitHub](https://github.com/DirtyDiablo/apify-linkedin-profile-search-by-name/blob/main/README.md#L1-L48) | Fork the repo and create a Dockerfile. Configure API credentials via environment variables. Build and run the container: docker build -t apify-linkedin . and docker run --env APIFY\_TOKEN=... apify-linkedin. |
| **fpdsScraper** | Pull award data from FPDS: contract value, dates, NAICS, prime/sub[GitHub](https://github.com/DirtyDiablo/fpdsScraper/blob/main/README.md#L1-L120) | Fork and pip install dependencies in a virtualenv or Docker container. Confirm connectivity to FPDS (may require authentication or proxies). Write a wrapper script that takes a URL/award ID and outputs structured JSON. |
| **job-board-scraper** | Scrape jobs from Greenhouse, Lever and store data[GitHub](https://github.com/DirtyDiablo/job-board-scraper/blob/main/README.md#L9-L31) | Fork; update spiders to include DoD primes (SAIC, Leidos, GDIT). Use Docker to encapsulate the Python environment and schedule runs via n8n. |
| **JobSpy & jobsparser** | Concurrent scraping from multiple boards (LinkedIn, Indeed, Google, Glassdoor, ZipRecruiter)[GitHub](https://github.com/DirtyDiablo/JobSpy/blob/main/README.md#L3-L53)[GitHub](https://github.com/DirtyDiablo/jobsparser/blob/main/README.md#L1-L68) | Install via pip; set up proxies if needed. Create command‑line scripts that accept keywords, locations and return CSVs matching your schema. |
| **CrossLinked** | Enumerate employee names via search engine scraping[GitHub](https://github.com/DirtyDiablo/CrossLinked/blob/master/README.md#L11-L105) | Clone and build a Docker image. Set up proxies. Write a script to run CrossLinked with company names and output names to CSV. |
| **linkedapi-node** | Send LinkedIn connection requests and fetch profile/company data[GitHub](https://github.com/DirtyDiablo/linkedapi-node/blob/main/README.md#L19-L35)[GitHub](https://github.com/DirtyDiablo/linkedapi-node/blob/main/README.md#L73-L102) | Install Node dependencies. Obtain LinkedIn API tokens (requires LinkedIn developer account). Integrate into n8n for automated actions. |
| **linkedin-message-generator** | Use OpenAI and Google search to craft personalised messages[GitHub](https://github.com/DirtyDiablo/linkedin-message-generator/blob/main/README.md#L9-L22)[GitHub](https://github.com/DirtyDiablo/linkedin-message-generator/blob/main/README.md#L38-L45) | Install via Node, supply OpenAI and Google API keys. Use this to generate outreach messages after your engine produces call lists. |

Make sure to configure each repo’s environment variables via your .env or n8n credentials, and test them independently before chaining them together.

**3. Set up Docker containers**

1. **Write Dockerfiles** for each tool. For example, for a Python scraper:

Dockerfile

CopyEdit

FROM python:3.10-slim

WORKDIR /app

COPY . .

RUN pip install -r requirements.txt

CMD ["python", "run\_scraper.py"]

1. **Build images**:

bash

CopyEdit

docker build -t fpds-scraper ./fpdsScraper

docker build -t jobspy ./jobspy

# Repeat for others

1. **Run containers** either directly or orchestrate them with Docker Compose. E.g.:

yaml

CopyEdit

# docker-compose.yml

version: '3.8'

services:

jobspy:

image: jobspy

environment:

- PROXY=...

volumes:

- ./data:/app/data

linkedin\_search:

image: apify-linkedin

environment:

- APIFY\_TOKEN=...

1. **Persist scraped data** by mounting the /data folder into your containers, so outputs go straight into data/program/jobs.csv.

**4. Set up n8n workflow automation**

1. **Install n8n** using Docker:

bash

CopyEdit

docker run -it --name n8n -p 5678:5678 \

-v ~/.n8n:/home/node/.n8n \

n8nio/n8n

1. **Configure credentials** inside n8n:
   * GitHub (PAT)
   * LinkedIn (for linkedapi-node)
   * OpenAI API
   * Slack/Teams webhook
2. **Create workflows** for the major tasks:
   * **Job scrape workflow** (runs 3× daily):
     + Trigger (cron) → call JobSpy container → parse output → append to jobs.csv → commit to GitHub.
   * **Engine run workflow**:
     + Trigger (after job scrape) → run Python engine (script inside repository) → generate jobs\_scored.csv, jobs\_delta.csv, update org.json → commit to GitHub.
   * **Profile enrichment workflow**:
     + Trigger (daily or when people\_targets.csv changes) → loop through new names → call apify LinkedIn actor → append results to people\_targets.csv.
   * **Notification workflow**:
     + When jobs\_delta.csv contains new or changed reqs, send a Slack message summarising the top roles and pay shifts.
   * **Outreach workflow**:
     + When outreach\_drafts.csv updates, call linkedin-message-generator to create a personalised InMail; optionally use linkedapi-node to send connection requests.
3. **Test each workflow** individually; then chain them via Webhook or Execute Workflow nodes.

**5. Program & Contract Discovery (Step-by-Step)**

1. **Upload research PDFs** (Competitive Capture & Insertion Strategy, Past Performance lists, etc.) into the /research folder of the repo.
2. **Run PDF parsing scripts**:
   * Use Python with pdfplumber or the provided pdf reader to extract program names, primes, contract values, NAICS codes, and performance periods.
   * Summarise each program into a row in programs.json: e.g., {"program": "AESD", "value": "$757M", "prime": "SAIC", "subs": [], "performance": "Nov 2022–Nov 2032"}.
3. **Scrape FPDS**:
   * For each contract name or ID, call the fpdsScraper functions (scrape\_fpds\_data) to fetch official award details[GitHub](https://github.com/DirtyDiablo/fpdsScraper/blob/main/README.md#L1-L120).
   * Append or update this information in your program map.
4. **Cross‑reference open‑source news**:
   * Run JobSpy with search terms like “contract award” + prime names to find press releases; summarise them into your map.

**6. Job & Requisition Intelligence (Step-by-Step)**

1. **Define search parameters**:
   * For each program, set keywords (e.g., AESD: “service desk”, “ServiceNow”), sites (Adelphi, Aberdeen), clearances (Secret, TS/SCI) and pay range filters.
2. **Run scraping tools**:
   * Use JobSpy for bulk scraping across LinkedIn, Indeed, Glassdoor, etc.; use job-board-scraper for Greenhouse/Lever jobs[GitHub](https://github.com/DirtyDiablo/job-board-scraper/blob/main/README.md#L9-L31); use jobsparser when you need CLI‑level control[GitHub](https://github.com/DirtyDiablo/jobsparser/blob/main/README.md#L1-L68).
   * Pipe the raw results into a normalisation script that matches your jobs.csv schema.
3. **Append to jobs.csv**:
   * Each run should add new roles with last\_seen\_utc set to the current timestamp. If the role already exists (match on req\_id or title+location), update last\_seen\_utc but don’t duplicate.
4. **Commit the updated file** to GitHub via n8n or a GitHub Action.

**7. Delta & Salary Benchmarking (Step-by-Step)**

1. **Run the engine** after each scrape; it:
   * Reads jobs.csv and compares it to jobs\_snapshot.csv.
   * Flags each row as new, changed or removed; captures changed fields and computes pay\_change\_min and pay\_change\_max.
   * Writes out jobs\_delta.csv and updates jobs\_snapshot.csv.
2. **Ingest competitor salary data**:
   * Parse competitors.csv where each row includes company, title, location, pay range.
   * During scoring, compute salary advantage: (competitor\_avg - our pay); if the advantage is ≥10 %, add 0.5 to priority.
3. **Review jobs\_delta.csv**:
   * This file is your daily call‑blitz list—focus on roles with change\_type=new, pay jumps, or changed locations/clearances.
   * Prioritise roles with positive pay changes (urgent) and those flagged by repost\_flag.

**8. Candidate & Contact Enrichment (Step-by-Step)**

1. **Populate people\_targets.csv**:
   * Use CrossLinked to gather names of PMs, SDMs, Site Leads at primes; store them with placeholders for email, phone, LinkedIn.
2. **Automate LinkedIn scraping**:
   * Use your fork of apify‑linkedin‑profile‑search‑by‑name. Write a script or n8n workflow to pass each name with company/location filters. Capture linkedinUrl, job title and any available contact info[GitHub](https://github.com/DirtyDiablo/apify-linkedin-profile-search-by-name/blob/main/README.md#L1-L48).
3. **Fill in emails**:
   * If “Full + email” mode returns a business email, add it. Otherwise, cross‑reference with your internal CRM or use contact‑finding services (RocketReach, ContactOut) where permitted.
4. **Update clearance & certs**:
   * Keep track of candidate archetypes (e.g., TS/SCI + ServiceNow + Security+) and note them in the notes field for easy matching.
5. **Personalise outreach**:
   * Once enriched, feed the profile to linkedin-message-generator along with your target messaging points (program name, pay advantage, cleared bench). Append the resulting message to outreach\_drafts.csv.

**9. Competitive & Past‑Performance Analysis (Step-by-Step)**

1. **Study Past Performance docs**:
   * Extract prime contractor strengths, weaknesses and key past performance metrics from the uploaded PDFs (e.g., “Past Performance Made Easy”).
   * Add these insights to your program map to inform wedge strategies (e.g., emphasise your RMF expertise when the prime historically struggles with compliance).
2. **Analyse prime org structures**:
   * Use findings from documents (e.g., PMO structure, IPT leads) to fill in roles in org.json.
   * Identify missing or high‑turnover roles; plan your backfill approach accordingly.
3. **Track competitor postings**:
   * Use JobSpy to scrape competitor postings; store salary ranges and tool stacks.
   * Compute site‑specific wage averages and highlight roles where you have a clear cost advantage.

**10. BD Output & Execution (Step-by-Step)**

1. **Generate outreach drafts**:
   * The engine will create 10 high‑priority drafts. Review each for tone and accuracy. Use the Sales Prospecting Playbook’s BLUF approach: start with who you are and your value proposition, then state your proof points and a clear ask.
2. **Create call sheets**:
   * Merge jobs\_delta.csv with people\_targets.csv to build a call list: each entry should have the target’s name, role, pain signal (e.g., repost\_30d, pay\_jump), phone/LinkedIn link, and your key talking points.
   * Plan your outreach sequence: call → follow‑up email → LinkedIn message. Use n8n to schedule reminders.
3. **Daily execution plan**:
   * Dedicate time each morning to review new contracts, pay shifts, and competitor activities.
   * By midday, complete your top calls and send follow‑up emails. Use the afternoon to enrich contacts and prep for the next cycle.
   * Record outcomes in a CRM (e.g., Bullhorn) so you can track progress and set follow‑up tasks.
4. **Follow BD playbook**:
   * Always ask for referrals and intelligence (e.g., “Do you know anyone leaving this program?”) as recommendedA screenshot of a computer

     AI-generated content may be incorrect..
   * Keep notes on who is the hiring manager, what their pain points are, and how your cleared bench solves those issues.

**11. Stay compliant and iterate**

* **Respect clearance & privacy**: Do not scrape private phone numbers or emails without consent; rely on publicly available business info and your ATS.
* **Rotate tokens** regularly to avoid API throttling and maintain security.
* **Iterate scoring parameters**: Review how well your priority scores align with actual response rates; tweak weights for AESD likelihood, urgency and salary advantage.
* **Refine search terms** and location filters as you discover new prime sites or program expansions.

By following these steps carefully, you can start from an empty machine and build a fully operational, customised BD engine that scrapes opportunities, tracks requisitions, enriches contacts, benchmarks salaries and generates targeted outreach. The key is to automate as much as possible with Docker and n8n, while constantly feeding your engine new intelligence from PDFs, FPDS, job boards and competitor activ