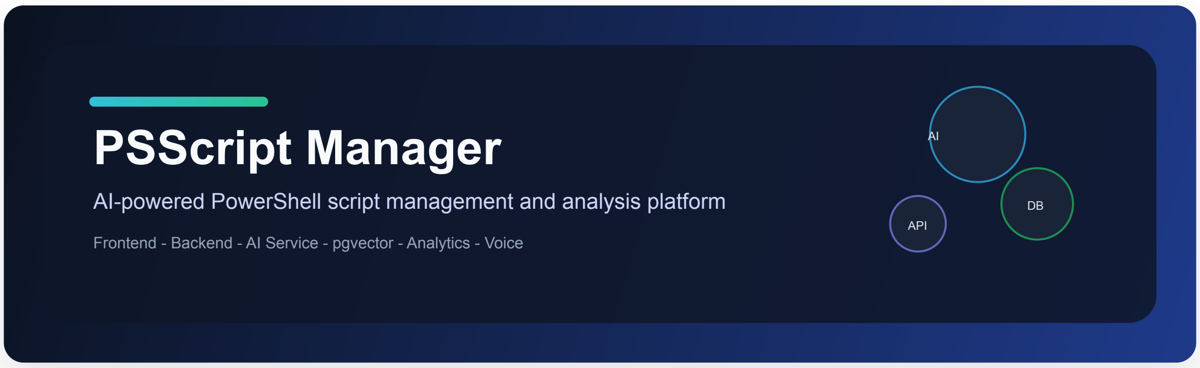
PSScript Manager

AI-powered PowerShell script management and analysis platform

Product README


PSScript Manager

AI-powered PowerShell script management and analysis platform

Frontend - Backend - AI Service - pgvector - Analytics - Voice

PSScript Manager is an AI-powered platform for managing, analyzing, and optimizing PowerShell scripts with an end-to-end workflow for teams.

status active

stack React | Node | Python

db Postgres + pgvector

ai LangGraph + OpenAI

Getting Started | Training Suite | Docker Quickstart | Support

Table of Contents

Executive overview

PSScript Manager is a centralized platform for PowerShell automation that balances governance, speed, and safety. It is structured so management can roll out in phases, engineers can integrate quickly, and new users can follow a guided path.

- Management: phased rollout plan, KPIs, and governance scorecards
- Engineering: AI analysis pipeline, vector search, and API-first workflows
- New users: structured training with screenshots, labs, and checklists

At-a-glance outcomes:

Outcome	**Description**	**Evidence in product**
Reduced risk	AI flags security issues and missing validation	Security scorecards + analysis tab
Faster discovery	Semantic search finds similar scripts	Search + embeddings
Better governance	Activity and review cadence tracked	Analytics + scorecards
Strong onboarding	Guided training with labs	Training suite + exports

Why PSScript Manager

- Centralizes PowerShell scripts with tagging, categories, and versioning
- Adds AI analysis for security, quality, and performance guidance
- Enables semantic search with pgvector embeddings
- Supports agentic workflows, AI chat, and documentation discovery
- Tracks activity, analytics, and governance signals

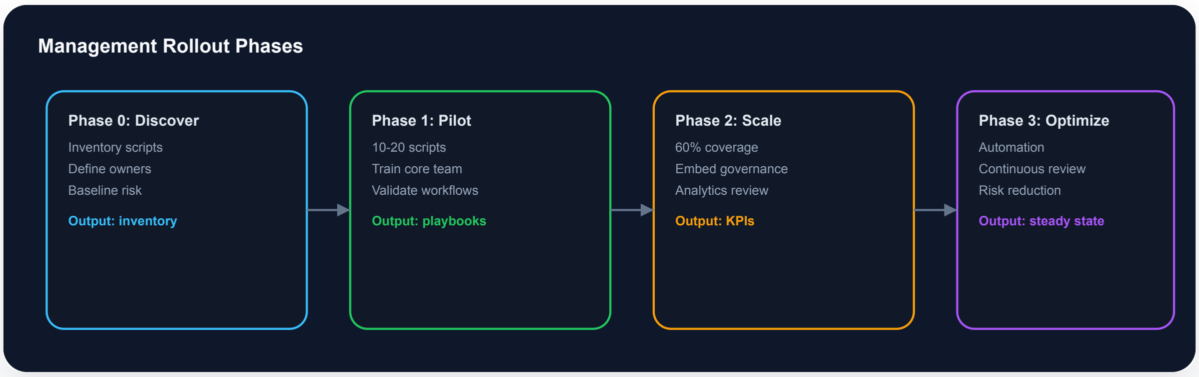
What it does

- Upload, edit, tag, and manage PowerShell scripts with metadata
- Run AI analysis for security risks, code quality, and recommendations
- Find similar scripts with vector and hybrid search
- Provide documentation explorer and AI chat
- Track analytics and usage trends
- Provide file integrity checks with hash-based deduplication
- Offer voice input and output in the AI service

Audience and outcomes

Audience	Primary goals	Key outcomes
Script authors	Upload, tag, and improve scripts	Clean library with metadata and analysis
Security reviewers	Identify and remediate risks	Scorecards and remediation notes
Platform admins	Operate services and reliability	Stable services and clean audit trail
Program managers	Rollout and governance	Phase milestones and KPI reporting

Management rollout plan

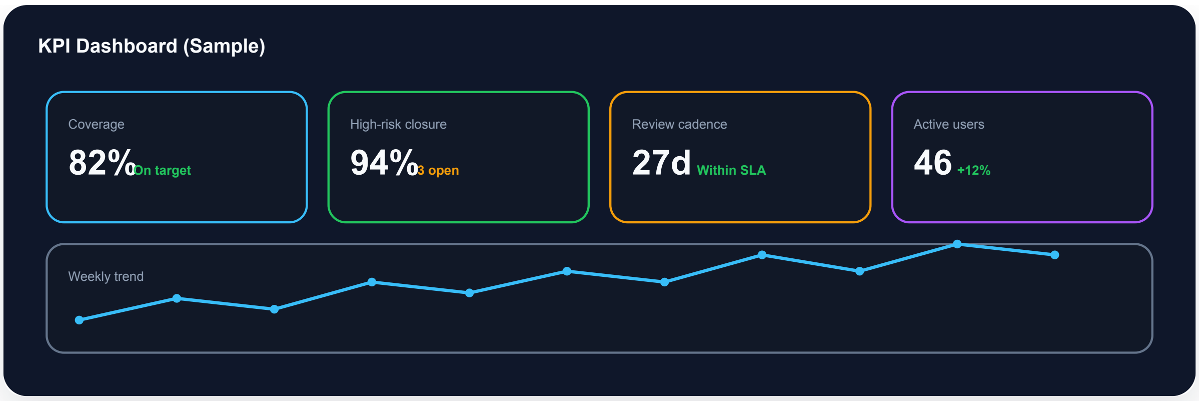


This README summarizes the phased approach. For full details, see `docs/MANAGEMENT-PLAYBOOK.md`.

Phase	Duration	Focus	Deliverables	Success signals
Phase 0: Discover	1-2 weeks	Inventory and ownership	Script inventory, owners, baseline risk	90% cataloged
Phase 1: Pilot	2-4 weeks	Validate workflows	Training completion, pilot scorecards	10-20 scripts analyzed
Phase 2: Scale	4-8 weeks	Expand coverage	Governance cadence, dashboards	60% coverage
Phase 3: Optimize	Ongoing	Continuous improvement	Automation backlog, SLA tracking	95% on-time reviews

Management KPI targets (sample):

KPI	Target
Coverage	>= 80% scripts analyzed
Review cadence	<= 30 days between reviews
High-risk closure	100% within 30 days
Owner tags	100% scripts assigned

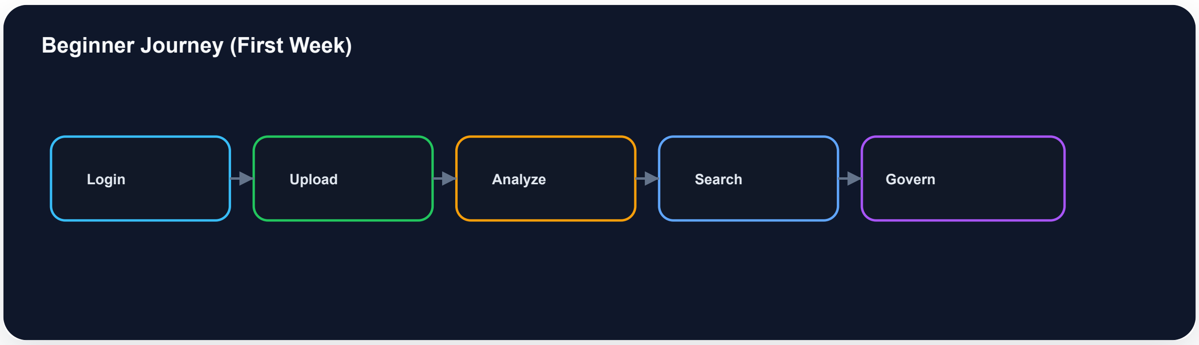


Monthly status template (management ready):

Section	What to report
Coverage	% scripts analyzed, top categories, backlog
Risk	High-risk findings, remediation status, SLA breaches
Adoption	Active users, searches per user, training completion
Operations	Uptime, incident count, release changes
Next steps	Decisions needed, upcoming training, owners

Beginner path (first week)

Use this path if you are brand new to the platform. It pairs screenshots with action steps and leaves deeper automation for later.



Day	Focus	Actions	Evidence
Day 0	Access + orientation	Login, open Dashboard, review Scripts list	Dashboard + Scripts view
Day 1	Upload + analysis	Upload one script, run analysis, review findings	Script detail + analysis tab
Day 3	Search + docs	Run keyword search, try vector search, read docs	Search results + docs panel
Day 7	Governance basics	Tag owners, capture remediation notes	Scorecard + audit notes

Recommended next step: complete `docs/training-suite/TRAINING-GUIDE.md`.

Day 0: Login and orient (15-30 minutes)

Step	Where	Action	Expected result	Screenshot
1	Login	Click "Use Default Login"	You land on the dashboard	docs/screenshots/login.png
2	Dashboard	Scan the top stats cards	You see total scripts and recent activity	docs/screenshots/dashboard.png
3	Scripts	Open the script list	You see scripts with tags and scores	docs/screenshots/scripts.png

Day 1: Upload and analyze (30-45 minutes)

Step	Where	Action	Expected result	Screenshot
1	Upload	Upload `test-script.ps1` and add tags	Script appears in the library	docs/screenshots/upload.png
2	Script detail	Open the script and view metadata	Tags, category, and version visible	docs/screenshots/script-detail.png
3	Analysis	Review findings and scores	Security and quality scores appear	docs/screenshots/analysis.png

Day 3: Search and documentation (20-30 minutes)

Step	Where	Action	Expected result	Screenshot
1	Scripts	Search for a cmdlet (example: Get-ADUser)	Keyword results are returned	docs/screenshots/scripts.png
2	Documentation	Search for a command	Documentation details appear	docs/screenshots/documentation.png

Day 7: Governance basics (20 minutes)

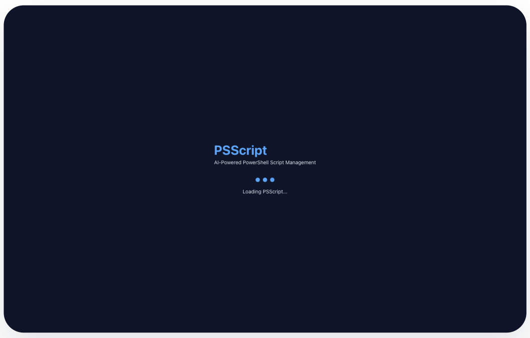
Step	Where	Action	Expected result	Screenshot
1	Script detail	Add an owner tag (example: owner:team-identity)	Ownership visible in metadata	docs/screenshots/script-detail.png
2	Analytics	Review usage trends	Activity metrics displayed	docs/screenshots/analytics.png
3	Scorecard	Record a remediation note	Findings tracked for follow up	docs/graphics/security-scorecard.svg

Beginner checklist:

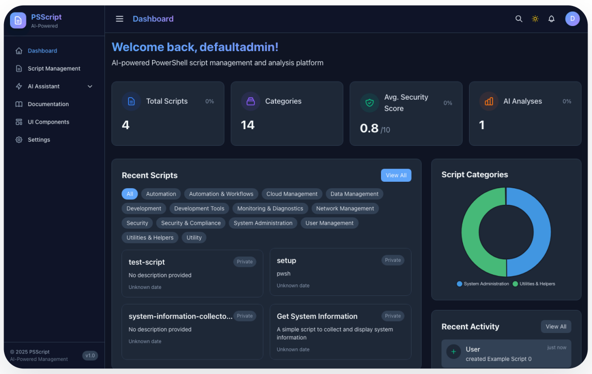
- [ ] Logged in with default credentials
- [ ] Uploaded a script and added tags
- [ ] Reviewed AI analysis scores
- [ ] Tried keyword or vector search
- [ ] Added an owner tag and review note

Screenshots

Login


PSScript
AI-Powered PowerShell Script Management
Loading PSScript...

Dashboard



Welcome back, defaultadmin!
AI-powered PowerShell script management and analysis platform

Total Scripts 0% 4
Categories 14
Avg. Security Score 0% 0.8 /10
AI Analyses 0% 1

Recent Scripts

test-script Private
No description provided
Unknown date

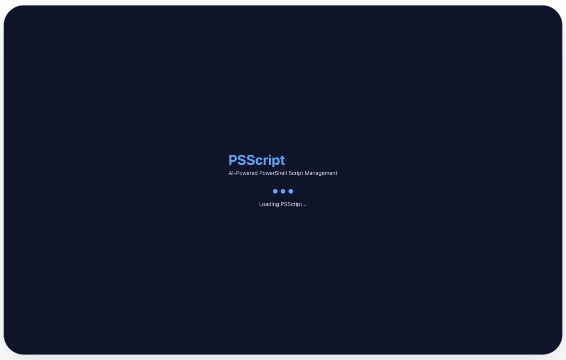
setup Private
pwsh
Unknown date

system-information-collecto... Private
No description provided
Unknown date

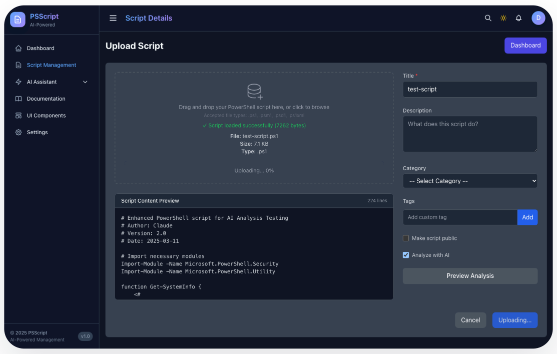
Get System Information Private
A simple script to collect and display system information
Unknown date

Recent Activity
User just now
created Example Script 0

Script Library


PSScript
AI-Powered PowerShell Script Management
Loading PSScript...

Script Detail



Upload Script

Drag and drop your PowerShell script here, or click to browse
Accepted file types: .ps1, .psm1, .psd1, .ps1xml
✓ Script loaded successfully (7262 bytes)
File: test-script.ps1
Size: 7.1 KB
Type: .ps1

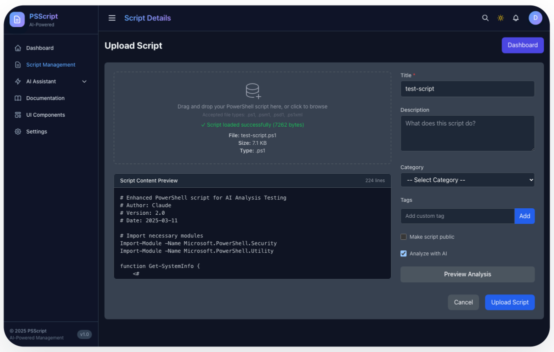
Uploading... 0%

Script Content Preview 224 lines

```
# Enhanced PowerShell script for AI Analysis Testing
# Author: Claude
# Version: 2.0
# Date: 2025-03-11

# Import necessary modules
Import-Module -Name Microsoft.PowerShell.Security
Import-Module -Name Microsoft.PowerShell.Utility

function Get-SystemInfo {
    <#
```

Title * test-script
Description: What does this script do?
Category: -- Select Category --
Tags: Add custom tag
☐ Make script public
☑ Analyze with AI

Script Upload



Upload Script

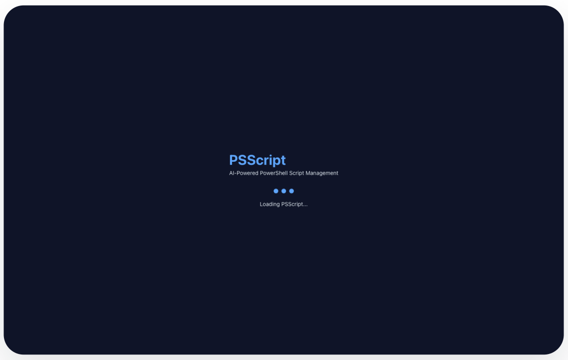
Drag and drop your PowerShell script here, or click to browse
Accepted file types: .ps1, .psm1, .psd1, .ps1xml
✓ Script loaded successfully (7262 bytes)
File: test-script.ps1
Size: 7.1 KB
Type: .ps1

Script Content Preview 224 lines

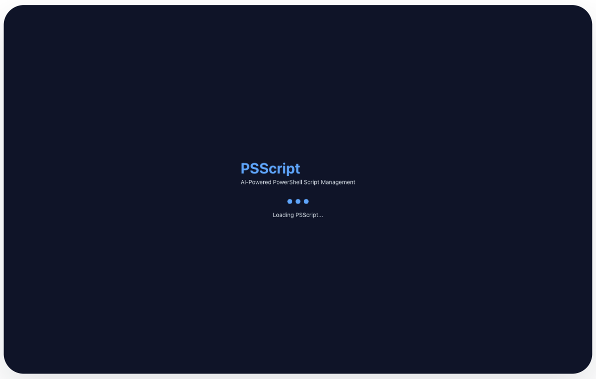
```
# Enhanced PowerShell script for AI Analysis Testing
# Author: Claude
# Version: 2.0
# Date: 2025-03-11

# Import necessary modules
Import-Module -Name Microsoft.PowerShell.Security
Import-Module -Name Microsoft.PowerShell.Utility

function Get-SystemInfo {
    <#
```

Title * test-script
Description: What does this script do?
Category: -- Select Category --
Tags: Add custom tag
☐ Make script public
☑ Analyze with AI

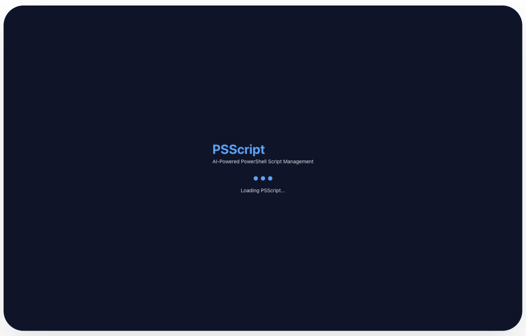
Script Analysis


PSScript
AI-Powered PowerShell Script Management
Loading PSScript...

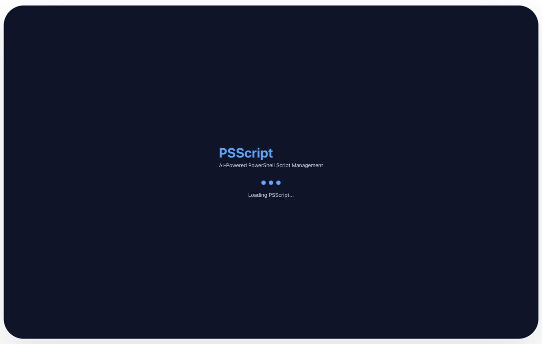
Documentation Explorer



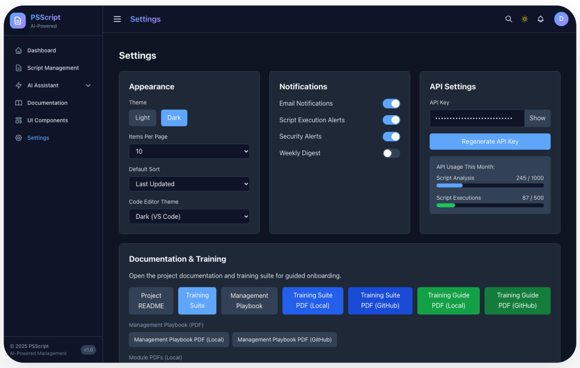
Analytics



AI Chat



Settings



Settings

Appearance

Theme: Light [✓ Dark]

Items Per Page: 10

Default Sort: Last Updated

Code Editor Theme: Dark (VS Code)

Notifications

Email Notifications: on

Script Execution Alerts: on

Security Alerts: on

Weekly Digest: off

API Settings

API Key: ••••••••••••••••••••••••••• Show

Regenerate API Key

API Usage This Month:

Script Analysis 245 / 1000

Script Executions 87 / 500

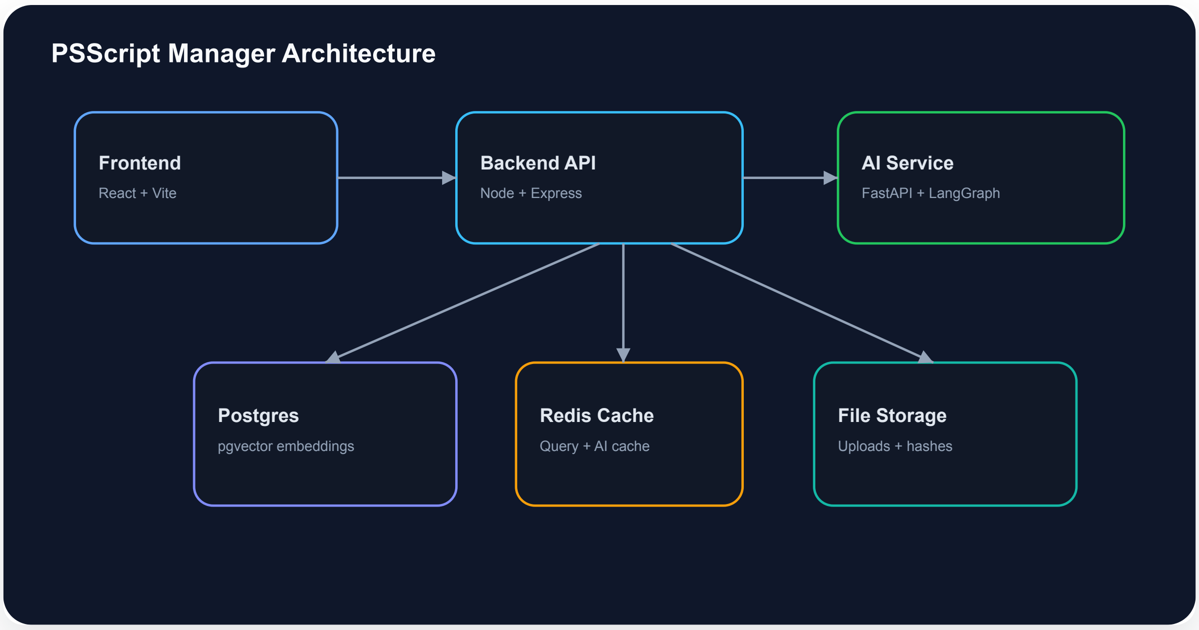
Documentation & Training

Open the project documentation and training suite for guided onboarding.

Project README
Training Suite
Management Playbook
Training Suite PDF (Local)
Training Suite PDF (GitHub)
Training Guide PDF (Local)
Training Guide PDF (GitHub)
Management Playbook (PDF)
Management Playbook PDF (Local)
Management Playbook PDF (GitHub)
Module PDFs (Local)

How it works

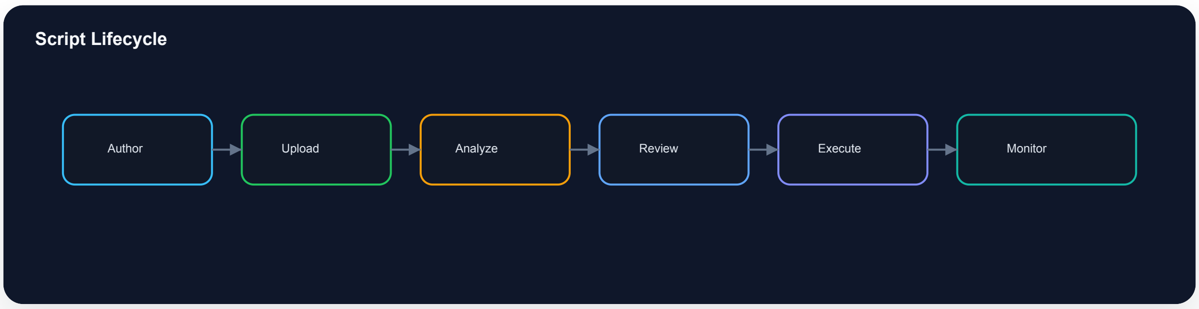
Platform architecture



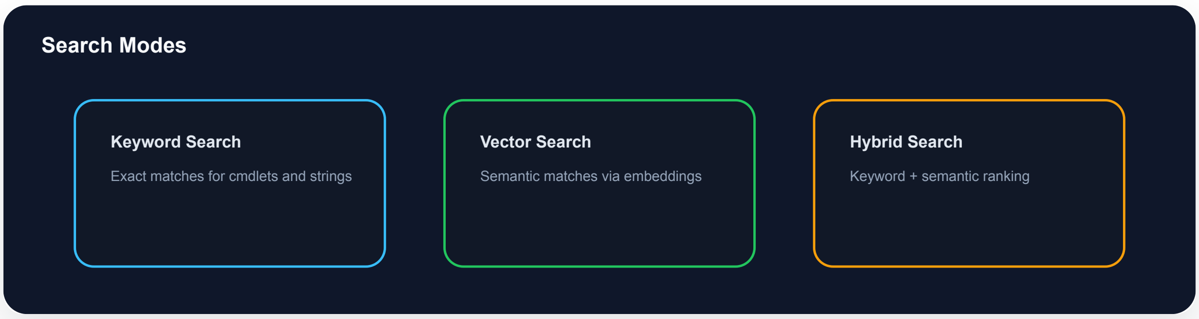
Analysis pipeline

Analysis Pipeline

Script lifecycle



Search modes



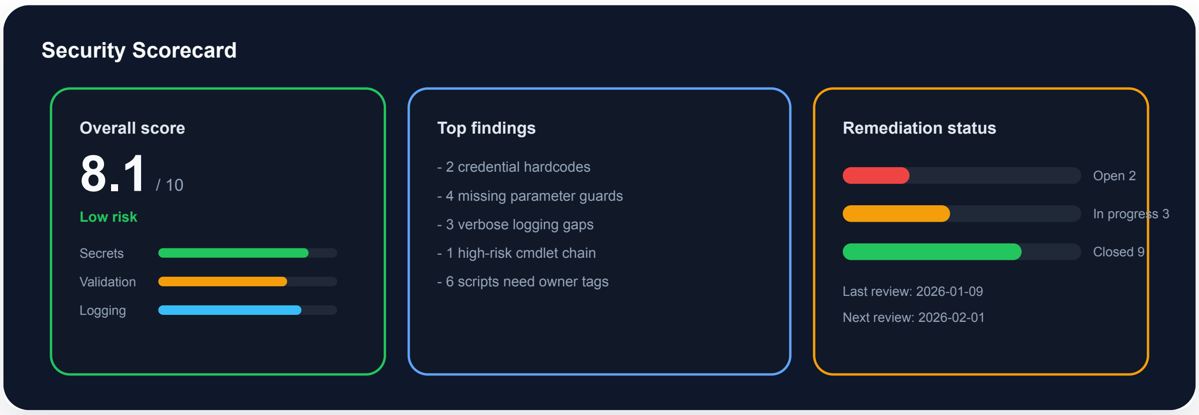
Search Modes

Keyword Search
Exact matches for cmdlets and strings

Vector Search
Semantic matches via embeddings

Hybrid Search
Keyword + semantic ranking

Security scorecard



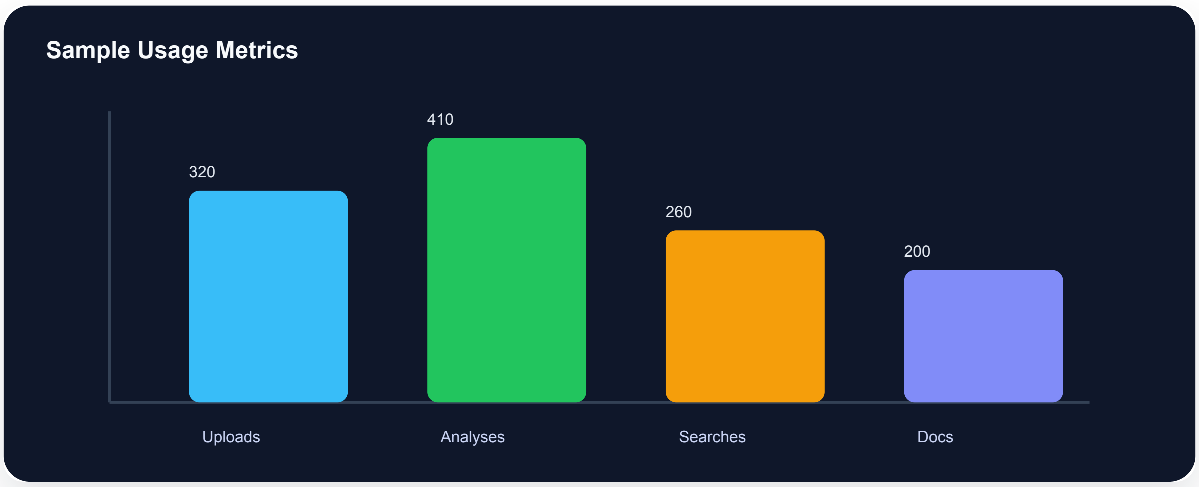
Security Scorecard

Overall score
8.1 / 10
Low risk
Secrets
Validation
Logging

Top findings
- 2 credential hardcodes
- 4 missing parameter guards
- 3 verbose logging gaps
- 1 high-risk cmdlet chain
- 6 scripts need owner tags

Remediation status
Open 2
In progress 3
Closed 9
Last review: 2026-01-09
Next review: 2026-02-01

Usage and governance signals



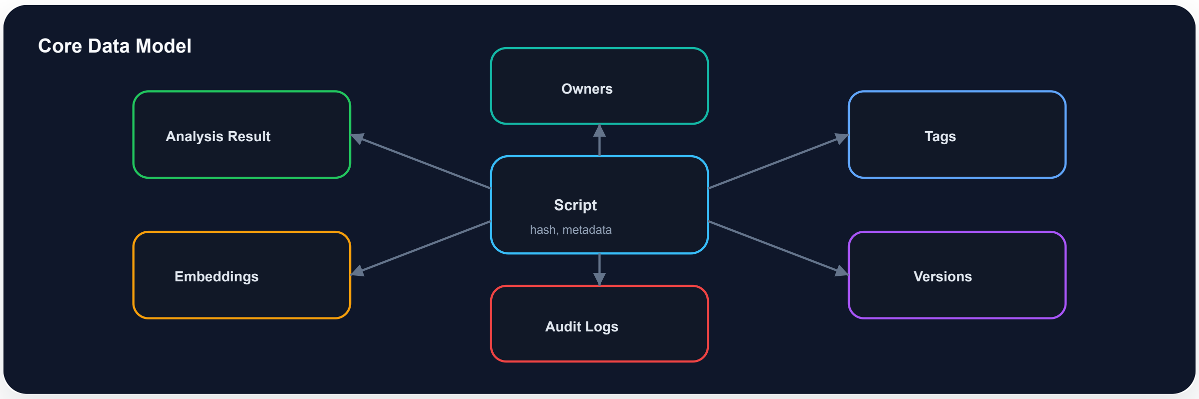
Script analysis output example

```
{
  "security_score": 7.8,
  "quality_score": 8.3,
  "issues": [
    "Hardcoded credential found on line 42",
    "Missing input validation for parameters"
  ],
  "recommendations": [
    "Use Get-Credential or SecretStore",
    "Add ValidateSet or ValidatePattern"
  ]
}
```

Architecture and services

Component	Role	Tech	Key paths
Frontend	UI and workflows	React, Vite, TypeScript	src/frontend
Backend	API, auth, scripts	Node, Express, TypeScript	src/backend
AI Service	Analysis, chat, voice	Python, FastAPI, LangGraph	src/ai
Database	Metadata and embeddings	Postgres, pgvector	src/db
Cache	Performance	Redis	docker/

Core data model



Entity	Purpose	Notes
Script	Source content and metadata	Stored with hash for dedup
ScriptAnalysis	AI output and scorecards	Linked to Script
Embeddings	Vector representations	Used for similarity search
Tags	Metadata and ownership	Enables filtering and governance
ScriptVersion	Change history	Supports comparison and rollback
ExecutionLog	Run history and outcomes	Used for audit and analytics

Feature matrix

Area	Capabilities	Notes
Script management	Upload, edit, tag, categorize, version	UI + API endpoints
AI analysis	Security, quality, performance, best practices	Mock mode supported
Vector search	Similarity and hybrid search	pgvector embeddings
Documentation	Crawl and explore PowerShell docs	Search + stats
Analytics	Usage, token tracking, trends	Analytics dashboard
Voice	TTS and STT endpoints	Multi-provider support
Integrity	Hash deduplication	Prevents duplicates

Engineer deep dive

API surface (selected)

Service	Endpoint	Method	Purpose
Backend	/api/scripts	GET	List scripts with filters
Backend	/api/scripts/upload	POST	Upload script file + metadata
Backend	/api/scripts/:id/analyze	POST	Run AI analysis and persist
Backend	/api/scripts/:id/analysis-stream	GET	SSE analysis progress
Backend	/api/scripts/search	GET	Keyword search + filters
Backend	/api/scripts/upload/async	POST	Async upload queue
Backend	/api/scripts/upload/status/:uploadId	GET	Async status tracking
AI service	/analyze	POST	Analyze script content
AI service	/chat	POST	AI chat workflows
AI service	/embedding	POST	Create embeddings
AI service	/similar	POST	Semantic similarity search

Workflow notes

- Upload flow writes script metadata, hashes, and tags before analysis runs.
- Analysis flow stores scorecards and findings as linked analysis records.
- Search uses keyword filters plus vector similarity when embeddings exist.
- Governance uses analytics and scorecards to drive review cadence.

Analysis pipeline detail

Analysis Pipeline

Stage	Input	Output	Notes
Ingest	Script file + metadata	Stored script + hash	Hash prevents duplicates
Analyze	Script content	Scores + findings	AI service computes scorecards
Persist	Analysis payload	ScriptAnalysis record	Linked to Script
Index	Script text	Embeddings	Enables semantic search
Report	Analytics events	Dashboards	Supports governance

Example requests (curl)

Upload a script file:

```
curl -X POST http://localhost:4000/api/scripts/upload \
  -H "Authorization: Bearer <token>" \
  -F "title=Reset-UserPassword" \
  -F "description=Reset AD user password" \
  -F "tags=security,active-directory" \
  -F "script_file=@test-script.ps1"
```

Analyze a script by ID:

```
curl -X POST http://localhost:4000/api/scripts/1/analyze \
  -H "Authorization: Bearer <token>"
```

Vector search by script ID:

```
curl http://localhost:4000/api/scripts/1/similar
```

Vector search by content (AI service):

```
curl -X POST http://localhost:8000/similar \
  -H "Content-Type: application/json" \
  -d '{"content": "active directory onboarding", "limit": 5}'
```

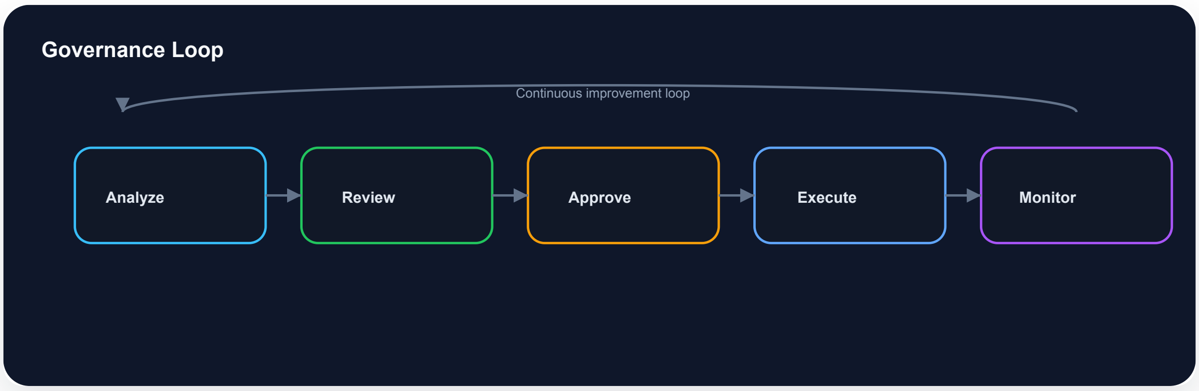
AI service analysis (direct):

```
curl -X POST http://localhost:8000/analyze \
  -H "Content-Type: application/json" \
  -d '{"content": "Get-ADUser -Filter *", "type": "security"}'
```

Local validation checklist

Check	Command	Expected result
Backend health	`curl http://localhost:4000/health`	status: ok
API health	`curl http://localhost:4000/api/health`	status: ok
AI health	`curl http://localhost:8000/health`	status: ok

Governance and security



Control	Cadence	Owner	Evidence
Security scorecard review	Weekly	Security lead	Scorecard + findings
Owner tagging	Continuous	Script owners	Metadata tags
Review cadence	Monthly	Program manager	Analytics + audit logs
High-risk remediation	Within 30 days	Security lead	Remediation notes

Scorecard thresholds (sample):

Score band	Meaning	Action
9.0 - 10	Low risk	Approve and monitor
7.0 - 8.9	Moderate risk	Fix and re-run analysis
< 7.0	High risk	Remediate before execution

Quickstart

Docker (recommended)

```
cp .env.example .env
mkdir -p backups/postgres backups/redis backups/logs
./docker-manage.sh start
```

See `DOCKER-QUICKSTART.md` for full setup and ports.

Mock mode

```
./start-all-mock.sh
```

Mock mode lets you run the UI without external dependencies or API keys.

Local development

```
 npm run install:all
npm run dev
```

Default ports: - Frontend: http://localhost:3002 - Backend API: http://localhost:4000/api - AI service: http://localhost:8000

Configuration

Key environment variables from `.env.example`:

Variable	Purpose	Example
OPENAI_API_KEY	AI analysis key	sk-...
AI_SERVICE_URL	AI service base URL	http://localhost:8000
DB_HOST	Postgres host	localhost
DB_PASSWORD	Postgres password	postgres
JWT_SECRET	Auth signing secret	your_secret
PGVECTOR_ENABLED	Enable embeddings	true
ENABLE_AI_ANALYSIS	Toggle AI analysis	true
VITE_DOCS_URL	Frontend docs export base URL	http://localhost:4000

Docs export routing: - Backend serves exports at `/docs/exports` - Optional frontend override: `VITE_DOCS_URL=http://localhost:4000`

Training suite

The training suite is a structured onboarding program with modules, labs, checklists, and screenshots.

- `docs/training-suite/README.md`
- `docs/training-suite/TRAINING-GUIDE.md`

Playwright screenshot automation

The screenshot set used in this README is generated with Playwright.

```
scripts/capture-readme-screenshots.sh
```

This script starts the mock services, captures all key UI screens, and stores them in `docs/screenshots/`.

Document exports (HTML + PDF + DOCX)

Generate polished exports of key docs (README, training guide, support):

```
scripts/export-docs.sh
```

Outputs land in `docs/exports/` with `html/`, `pdf/`, and `docx/` subfolders.

Key exports:

Document	PDF	DOCX
README	docs/exports/pdf/README.pdf	docs/exports/docx/README.docx
Training guide	docs/exports/pdf/Training-Guide.pdf	docs/exports/docx/Training-Guide.docx
Training suite	docs/exports/pdf/Training-Suite.pdf	docs/exports/docx/Training-Suite.docx
Management playbook	docs/exports/pdf/Management-Playbook.pdf	docs/exports/docx/Management-Playbook.docx
Support	docs/exports/pdf/Support.pdf	docs/exports/docx/Support.docx

With the backend running, exports are also served at: - http://localhost:4000/docs/exports/

Documentation

- `docs/GETTING-STARTED.md`
- `DOCKER-QUICKSTART.md`
- `docs/README-VECTOR-SEARCH.md`
- `docs/README-VOICE-API.md`
- `docs/LOGIN-CREDENTIALS.md`
- `docs/MANAGEMENT-PLAYBOOK.md`
- `docs/training-suite/README.md`
- `docs/SUPPORT.md`

Support

See `docs/SUPPORT.md` for issue reporting, triage, and operational checklists.

Documentation patterns referenced

Structure and layout cues inspired by: - https://github.com/supabase/supabase - https://github.com/appwrite/appwrite - https://github.com/posthog/posthog - https://github.com/pocketbase/pocketbase - https://github.com/prisma/prisma - https://github.com/istio/istio

Full reference list: `docs/REFERENCE-SOURCES.md`

License

MIT. See `LICENSE`.

Generated 2026-01-13 06:26 UTC