USER MANUAL

Harness-Chandelier

Find what the user keeps coming back to

Recurring-Intent Detection for Long AI Conversations — MCP Server, Bedrock and GPU Editions

Version 1.0.0 · Container product for AWS Marketplace

Klastrovanie Co., Ltd.

Results describe conversation patterns for AI agents and their operators; they are not judgements about people.

Contents

1. **What This Product Is**
2. **How It Works**
3. **Editions**
4. **Architecture and Delivery**
5. **Deployment on AWS**
6. **Configuration**
7. **Using the MCP Tools**
8. **Reading the Results**
9. **Languages**
10. **Token Usage, Performance and Sizing**
11. **Licensing and Entitlement**
12. **Health and Monitoring**
13. **Troubleshooting**
14. **Responsible Use and Data Protection**
15. **Disclaimer of Warranty and Limitation of Liability**
16. **Support**

1. What This Product Is

In long conversations with an AI agent, users drift between topics — and keep coming back to what they actually want. **Harness-Chandelier** finds that topic: the one the user returns to after other topics and after time away, even when another topic had more messages in one burst.

It runs as an **MCP server** (Model Context Protocol) in your own AWS account. An agent, or the application around it, sends the user's messages in conversation order and receives the **main topic** — its name, keywords, a one-sentence description of what the user keeps coming back for, and the messages that belong to it — together with every other topic, ranked.

Typical uses: keeping a long-running agent focused on the user's persistent intent, routing a conversation to the right specialist, and summarizing what a user really wanted for support and product teams.

Every model the product uses runs on **Amazon Bedrock** under your IAM role. Nothing is bundled in the image, no third-party AI API is called, and no data is sent to Klastrovanie.

Harness-Chandelier is the commercial MCP edition of the open-source Klastrovanie/Harness-Chandelier project (reverse topic-drift PageRank).

2. How It Works

The method is **reverse topic-drift PageRank**. The messages are grouped into topics; each pair of consecutive messages is a transition from one topic to the next. The transitions form a directed graph whose edges are weighted by

- **return gap** (`delta_time`) — how long the user stayed away before making the same transition again: coming back after a long time is a strong signal of real intent;
- **transition count** — how often the transition occurs.

PageRank over this graph ranks the topics. The topic at the top is the “chandelier”: the fixed point the conversation keeps circling back to. Short, context-free messages (“thanks”, “ok”) become outliers and do not influence the ranking.

> **EXAMPLE**
>
> A user spends one session asking 14 questions about a kitchen renovation and never returns to it, while asking about marathon training in short visits over three weeks. A message count would pick the kitchen; Harness-Chandelier picks marathon training.

3. Editions

One AWS Marketplace product with two delivery options. Both expose the same MCP tools and the same response shape, so a client can switch editions without changes.

	Bedrock edition	GPU edition
How topics are found	The messages, as written, go to an LLM on Amazon Bedrock, which groups them, names the topics and judges which one the user keeps returning to.	Each message's language is detected and the message is translated into English; English embeddings are clustered on the GPU; the topics are ranked with reverse topic-drift PageRank; names and the description are written in the user's language.
Models (Amazon Bedrock)	Anthropic Claude Sonnet 4.6	Amazon Nova Lite (language detection, translation, description); Cohere Embed English v3 (embeddings)
LLM tokens per call	The whole conversation, every call	Only new messages are translated (cached); one short description call
Hardware	CPU (no GPU)	NVIDIA GPU (RAPIDS cuML and cuGraph)
Messages per call	Default 300, up to 600	Default 2,000, up to 5,000
Best for	Short to medium conversations, simplest setup	Long or frequently re-analyzed conversations; ranking by the PageRank method; cost at scale

The two editions can name a different main topic for the same conversation when several topics recur: the Bedrock edition reflects the model's judgement, the GPU edition the PageRank ranking.

4. Architecture and Delivery

- **Bedrock edition image**: Debian-based Python 3.12 runtime; MCP server on port 8000, path `/mcp`.
- **GPU edition image**: NVIDIA RAPIDS base (CUDA 12, Python 3.11) with cuML (UMAP, HDBSCAN) and cuGraph (PageRank); same server, port and path.
- The application ships as one compiled module. The images contain no application source code, no shell tools such as curl or wget, and no package manager in the Bedrock edition.
- The server is **stateless** (streamable HTTP). Send the whole conversation each time; the GPU edition keeps a per-container in-memory cache of translations and embeddings, so re-sending a growing conversation only processes the new messages.
- The images run as non-root users.

5. Deployment on AWS

Where it runs

Run the container on **Amazon ECS** or **Amazon EKS**. The Bedrock edition runs on any x86_64 CPU instance or AWS Fargate. The GPU edition needs a GPU EC2 instance.

GPU edition host requirements

- An NVIDIA GPU, for example `g4dn`, `g5`, `g6` or `g6e` instances; the engine uses one GPU.
- An NVIDIA driver that supports CUDA 12 and the NVIDIA container toolkit; current ECS GPU-optimized AMIs and AWS Deep Learning AMIs provide both.
- Pass the GPU to the container: `--gpus all` (Docker), a GPU `resourceRequirements` entry (ECS), or `nvidia.com/gpu: 1` (EKS). Without a GPU the GPU edition reports an error on each call and in its log.

Amazon Bedrock model access

The models run in your account, so they must be available to it in the Region you use. In the Amazon Bedrock console, confirm access to:

Edition	Models
Bedrock	Anthropic Claude Sonnet 4.6 (for Anthropic models, submit the one-time use-case details if the account has not used them before)
GPU	Amazon Nova Lite; Cohere Embed English v3

By default the LLMs are called through the cross-Region inference profile of your geography (`us.`, `eu.`, `apac.` ...), derived from `AWS_REGION`; see Section 6 to change the models or the geography.

NETWORK REQUIREMENT (OUTBOUND TO AWS)

The container needs outbound access to Amazon Bedrock Runtime in your Region and to `license-manager.us-east-1.amazonaws.com`, through a NAT gateway or VPC endpoints, and AWS credentials from its task, pod or instance role. When you start it with `docker run` on EC2 with IMDSv2, use `--network host` or an instance metadata hop limit of 2 so the container can read the instance role credentials.

Required IAM permissions

Grant the task role, pod role or instance profile access to Amazon Bedrock and to the AWS Marketplace entitlement. You can restrict the Bedrock resources to the inference profiles and foundation models listed above.

```
{
  "Version": "2012-10-17",
  "Statement": [
    { "Effect": "Allow",
      "Action": "bedrock:InvokeModel",
      "Resource": "*" },
    { "Effect": "Allow",
      "Action": [
        "license-manager:CheckoutLicense",
        "license-manager:ExtendLicenseConsumption",
        "license-manager:CheckInLicense",
        "license-manager:ListReceivedLicenses"
      ],
      "Resource": "*" }
  ]
}
```

On first use of AWS License Manager in an account, create its service-linked role once:

```
aws iam create-service-linked-role --aws-service-name license-manager.amazonaws.com
```

Example: ECS task definition (GPU edition, excerpt)

```
{
  "containerDefinitions": [{
    "name": "harness-chandelier",
    "image": "<marketplace-image-uri>:1.0.0",
    "portMappings": [{ "containerPort": 8000 }],
    "resourceRequirements": [{ "type": "GPU", "value": "1" }],
    "environment": [{ "name": "AWS_REGION", "value": "us-east-1" }],
    "healthCheck": {
      "command": ["CMD", "python", "-c",
        "import urllib.request,sys; sys.exit(0 if
urllib.request.urlopen('http://localhost:8000/health').status==200 else 1)"],
      "interval": 30, "timeout": 5, "retries": 3, "startPeriod": 60 }
  }],
  "cpu": "4096", "memory": "16384"
}
```

For the Bedrock edition, remove `resourceRequirements`; 1 vCPU and 2 GiB of memory are enough.

6. Configuration

No configuration is required to start. Optional environment variables:

Variable	Meaning (default)
`AWS_REGION`	Region for Amazon Bedrock calls (set automatically on ECS and EKS; default `us-east-1`).
`HC_BEDROCK_GEO`	Inference profile geography: `us`, `eu`, `jp`, `au` or `global` (derived from the Region).
`HC_BEDROCK_MODEL`	Bedrock edition LLM (`<geo>.anthropic.claude-sonnet-4-6`).
`HC_LIGHT_MODEL`	GPU edition model for language detection, translation and the description (`<geo>.amazon.nova-lite-v1:0`).
`HC_EMBED_MODEL`	GPU edition embedding model (`cohere.embed-english-v3`).
`HC_MAX_MESSAGES`	Default number of most recent messages analyzed per call (Bedrock 300, GPU 2,000).
`HC_MAX_MESSAGE_CHARS`	Longer messages are shortened to this many characters (4,000).
`HC_EMBED_CACHE_SIZE`	Cached translations and embeddings per container (20,000; `0` disables caching).
`HC_EMBED_CONCURRENCY`	Parallel Bedrock requests per call (8).
`PORT`, `LOG_LEVEL`	Server port (8000) and log level (INFO).

Use any model other than the defaults only after testing it on your own conversations.

7. Using the MCP Tools

Connect an MCP client to `http://HOST:8000/mcp` (streamable HTTP). Two tools are available.

find_main_topic

Argument	Meaning
`messages` (required)	The user's messages in conversation order. Each item is a string or an object `{"text": "...", "timestamp": "2026-01-10T09:00:05Z"}`; a timestamp may also be epoch seconds. Give a timestamp for every message or for none; without timestamps, messages count as one minute apart. Timestamps must be in conversation order.
`max_messages`	How many of the most recent messages to analyze (5 to the edition's limit).
`delta_time_weight`	GPU edition, −1 to 1 (default 0.2). Positive favours topics the user returns to after time away (intent); negative favours rapid back-and-forth switching (noise and blockers).

Send the user's own messages, not the agent's replies. Call the tool whenever the agent needs to re-center, for example every few turns; send the whole conversation each time.

```
{"messages": [
  {"text": "I signed up for a marathon in October. How should I start training?",
   "timestamp": "2026-03-02T09:00:00"},
  {"text": "Different thing: we are renovating our kitchen next month.",
   "timestamp": "2026-03-02T10:00:00"},
  ...
]}
```

engine_info

Returns the edition, version, Region, Bedrock model IDs, message limits, whether a GPU is visible (GPU edition) and the license status. Use it to verify a deployment.

8. Reading the Results

A real response of the GPU edition (shortened) for the marathon-and-kitchen conversation of Section 2:


```
{
  "edition": "gpu",
  "method": "translate-embed-graph-pagerank",
  "language": "en",
  "messages_analyzed": 29,
  "main_topic": {
    "topic_id": 0,
    "label": "marathon training",
    "keywords": ["marathon", "run", "km", "training", "long"],
    "keywords_en": ["marathon", "run", "km", "training", "long"],
    "summary": "The user keeps coming back to marathon training, asking about
                planning, running distances, and nutrition.",
    "score": 0.376223,
    "message_count": 11,
    "share_of_messages": 0.3793,
    "message_indices": [0, 1, 16, 17, 18, 21, 22, 23, 26, 27, 28],
    "representative_messages": [{"index": 21,
      "text": "Marathon again: how do I plan my long runs for the next 8 weeks?"}]
  },
  "topics": [
    {"topic_id": 0, "label": "marathon training",  "score": 0.376223, "message_count":
11},
    {"topic_id": 1, "label": "kitchen renovation", "score": 0.331662, "message_count":
11},
    {"topic_id": 3, "label": "meal preparation",   "score": 0.168232, "message_count":
4},
    {"topic_id": 2, "label": "contractor hiring",  "score": 0.123883, "message_count":
3}
  ],
  "outlier_indices": [],
  "usage": {"llm_input_tokens": 1366, "llm_output_tokens": 1100,
            "translated_messages": 29, "translation_cache_hits": 0,
            "embedded_messages": 29, "embedding_cache_hits": 0},
  "notes": []
}
```

Field	Meaning
`main_topic`	The topic the user keeps returning to; `null` when the conversation is too short or too varied (see `notes`).
`label`, `keywords`, `summary`	Name, keywords and what the user keeps coming back for, in the user's language. The GPU edition also returns `keywords_en`.
`score`	GPU edition: the topic's PageRank (higher = more central to the conversation). Bedrock edition: the model's ranking scaled to 0–1 (1 = main topic).
`message_indices`	Positions of the topic's messages in the list you sent.
`representative_messages`	Up to three messages that best represent the topic.
`topics`	All topics, most persistent first.
`outlier_indices`	Messages that belong to no topic (greetings, very short replies).
`language`	GPU edition: the language most messages are written in (ISO 639-1).
`usage`	Bedrock tokens and the number of messages translated and embedded in this call (cache hits are free).
`notes`	Information such as truncated messages, a limited window, or missing timestamps.

The main topic can change as a conversation grows: it reflects the conversation so far. With few messages (fewer than five for the GPU edition, three for the Bedrock edition) no main topic is returned.

9. Languages

- **Input**: messages may be written in any language, and languages may be mixed within a conversation.
- **Bedrock edition**: the LLM reads the messages as written and answers in the language the user mostly writes in.
- **GPU edition**: the language of each message is detected and the message is translated into English before it is embedded, so topics are formed in one language regardless of the input. The names, keywords and description are then written in the language most messages are written in.
- Coverage depends on the Bedrock models in use; test with conversations in your own languages.

10. Token Usage, Performance and Sizing

Bedrock usage is billed to your AWS account by AWS. Measured on test conversations:

Situation	Bedrock edition	GPU edition
55 messages, first call	about 2,000–3,100 input and 450 output tokens	55 messages translated and embedded; about 2,400–4,500 input and 2,100–2,500 output tokens (Nova Lite)
Same 55 messages again	the same as the first call	nothing translated or embedded; about 1,500 input and 300 output tokens for the description
15 new messages added	the whole conversation again	only the 15 new messages translated and embedded

- Latency is dominated by the Bedrock calls: seconds for tens of messages on the first call; repeated calls of the GPU edition are faster because of the cache.
- The GPU edition processes one analysis at a time per container; scale out with more tasks or pods for many concurrent conversations. Each running container needs one unit of your contract (Section 11).
- Sizing: Bedrock edition 1 vCPU, 2 GiB; GPU edition 4 vCPU, 16 GiB and one GPU.

11. Licensing and Entitlement

The product is sold under AWS Marketplace contract pricing and validates its entitlement against **AWS License Manager**. There are no license keys: the container reads its entitlement from the AWS account it runs in and renews it every 40 minutes.

There are two plans. The plan decides which edition you may run; both plans include unlimited calls, and the message limits of Section 3 apply in each edition.

Plan (contract dimension)	Bedrock edition	GPU edition
Bedrock License `bedrock_license`	Included	Not included
GPU License `gpu_license`	Included	Included

- **One unit = one running container.** Each unit of a plan covers one container (ECS task, Kubernetes pod or Docker container) running at a time. To run N containers at once, for example to scale out, buy N units of the plan. A container that starts while all units are in use answers 402 and retries every minute; it starts serving as soon as a unit is free. A container that stops normally returns its unit at once; one that ends abruptly holds its unit for up to one hour.
- Rolling deployments: if the number of containers equals the number of units, let the old container stop before the new one starts (ECS `minimumHealthyPercent` below 100, for example 0 for a single task; Kubernetes `maxSurge: 0`), or buy one extra unit for the overlap.

- The GPU edition starts serving only with a GPU License. With a Bedrock License it answers 402, and the `license` object in `/health` names the plan that is needed. To move to the GPU edition, upgrade the contract to the GPU License; no redeploy of the Bedrock edition is needed.
- At startup the server begins serving once the entitlement has been confirmed; until then `/health` returns 503, normally for a few seconds.
- While the contract is active, the server works normally and `GET /health` returns 200. If the contract expires or the entitlement is revoked, the MCP endpoint returns **HTTP 402** and `/health` returns **HTTP 503**. After the entitlement has been confirmed, temporary failures to reach License Manager are tolerated for a grace period of up to 72 hours. Renewing restores service without a redeploy.
- Without the license-manager permissions of Section 5 the entitlement cannot be confirmed and the MCP endpoint returns 402. The `license` object in the `/health` response shows the state and the last error reported by License Manager.

12. Health and Monitoring

Use `GET /health` on port 8000 as the health check. It returns HTTP 200 with `{"status":"ok", ...}` while entitled, and HTTP 503 with `{"status":"license_expired", ...}` otherwise; the response also shows the edition, version and license state.

The images do not include curl. For ECS or Kubernetes health checks use Python's standard library, as in the task definition in Section 5. If you do not want the task or pod restarted when the entitlement lapses, use `/health` as a readiness probe rather than a liveness probe.

The container logs to standard output: edition, Region and models at start, license state, warnings (for example a missing GPU) and errors. `engine_info` shows the same information on demand.

13. Troubleshooting

Message or symptom	Cause and fix
HTTP 402 from `/mcp`, 503 from `/health`	No active subscription in this account, or the role lacks the license-manager permissions. Check the `license` object in `/health`.
402 from the GPU edition; `/health` says the license does not include the GPU edition	The account holds the Bedrock License. Use the Bedrock edition, or upgrade the contract to the GPU License.
402 from a new or additional container while others run; `/health` shows `NoEntitlementsAllowedException`	All units of the contract are in use (one unit per running container). Buy more units, or stop a container; the new one retries every minute and starts serving when a unit is free.
AccessDenied ... Service role not found	Create the License Manager service-linked role (Section 5).
AccessDeniedException calling InvokeModel or Converse	Add the Bedrock permissions of Section 5 to the role and confirm model access in the Bedrock console for the Region.
ValidationException ... model identifier is invalid / not available in this Region	The model or inference profile is not offered in your Region. Set `HC_BEDROCK_GEO` or the model variables (Section 6).
ThrottlingException	Bedrock quota reached; requests are retried automatically. Request a higher quota, or lower `HC_EMBED_CONCURRENCY`.
No NVIDIA GPU is visible to the container	GPU edition without a GPU. Start it with `--gpus all`, a GPU resource requirement (ECS) or `nvidia.com/gpu: 1` (EKS).
`main_topic` is null	Too few messages, or no topic recurs yet. See `notes`; call again as the conversation grows.
give a timestamp for every message or for none of them	Some messages have timestamps and others do not.
timestamps must be in conversation order	A message is earlier than the one before it. Send messages in conversation order.
Topic names in English for a non-English conversation (GPU edition)	The description step was unavailable; the ranking stands and English keywords are shown. Check `notes` and the light model's access.

14. Responsible Use and Data Protection

The product runs entirely in your AWS account. The messages you send are processed by Amazon Bedrock under your account and IAM role; nothing is sent to Klastrovanie. The only other outbound calls are AWS License Manager entitlement checks, which carry no customer data.

Topic	Position
Conversation content	Messages may contain personal data. You are responsible for a lawful basis for processing, for informing users where required, and for not sending data you may not process with Amazon Bedrock.
Cross-Region inference	The default inference profiles keep processing within one geography (for example `eu.` for EU Regions). Choose the geography that matches your data-residency requirements (Section 6).
Storage	The server writes nothing to disk. The GPU edition keeps translations and embeddings in memory only, per container, until it restarts; set `HC_EMBED_CACHE_SIZE=0` to disable.
Access	Who can call the MCP server is controlled by your network configuration (private subnets, security groups) or by authentication on your MCP gateway.
Interpretation	Results describe what a conversation keeps returning to. They are not assessments of a person.

⚠ **NOT FOR DECISIONS ABOUT PEOPLE**

Do not use results to profile, evaluate or make decisions producing legal or similarly significant effects on a person.

15. Disclaimer of Warranty and Limitation of Liability

This section is a plain-language summary for information only. It is not legal advice and does not create new contractual terms. The binding terms are those of the **Harness-Chandelier MCP End User License Agreement** under which you obtained the software through AWS Marketplace. Where this summary and the agreement differ, the agreement controls.

No warranty. To the maximum extent permitted by applicable law, the software and all outputs are provided “as is” and “as available,” without warranties of any kind, including merchantability, fitness for a particular purpose, accuracy or non-infringement. Klastrovanie does not warrant that any output is correct, complete or suitable for any particular purpose.

Customer responsibility. You are responsible for the messages you send, the configuration of your AWS environment and Amazon Bedrock models, the review of outputs, and any decision made or action taken on the basis of an output.

Limitation of liability. To the maximum extent permitted by applicable law, Klastrovanie Co., Ltd. is not liable for any decision made or action taken in reliance on the software or its outputs, for the cost of AWS infrastructure or Amazon Bedrock usage, or for indirect, incidental, special, consequential, exemplary or punitive damages, or loss of profits, revenue, data or goodwill.

16. Support

Email **contact@klastrovanie.com**. Support is provided by Klastrovanie Co., Ltd. during Korean business hours (KST). Please include the image version and edition, the output of `engine_info` or the container log, and the `notes` of the response. Do not send conversation content or other personal data.
