Hestia: The IO-SEA HSM API

K. O'Connor, J. Grogan, B. Benek Gursoy

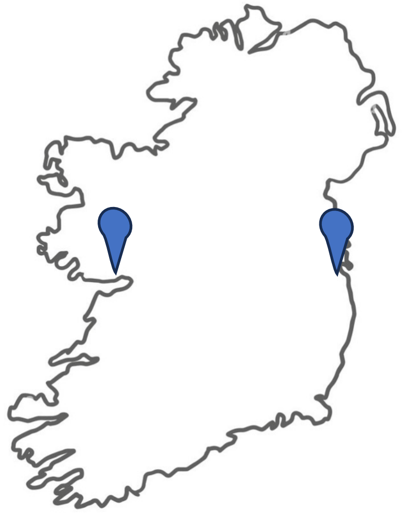
Irish Centre for High End Computing

katie.oconnor@ichec.ie

Irish Centre for High End Computing

- Ireland's national supercomputing centre
- Upcoming System: CaspIr (EuroHPC JU System)

- Activities:
  - Quantum Computing
  - Data Science
  - Earth Observation
  - Performance Engineering
  - National Service
  - Education and Training



ichec.ie

Ollscoil na Gaillimhe
University of Galway

ICHEC
Irish Centre for High-end Computing

SEA Project Consortium

IO-SEA

Storage I/O and Data Management for Exascale Architectures

DEEP-SEA

Developing a programming environment for European Exascale systems

RED-SEA

Network Interconnect for Exascale Systems

IO-SEA project partners

Motivation behind IO-SEA

- Data storage is becoming a bottleneck in the move to exascale systems
- Concerns about scaling of current storage solutions to exascale
- IO-SEA is investigating new storage paradigms for exascale
- In particular, using a HSM object store

The approach

- IO-SEA follows a co-design approach with several ”data hungry” use cases chosen to represent the sort of workflows that will be run on future exascale systems

- These use cases were chosen because they present significant challenges with IO due to the amount of data produced and consumed during their lifecycle

- The IO-SEA usecases are:
  - Simulation of astrophysical plasma flows (RAMSES)
  - Lattice QCD
  - Analysis of electron microscopy images
  - ECMWF weather forecasting workflow
  - Terrestrial Systems Modelling Platform

Data workflow

- In traditional HPC workflows, data is computed, stored in the global filesystem and then has to be retrieved again for the next stage of the compute

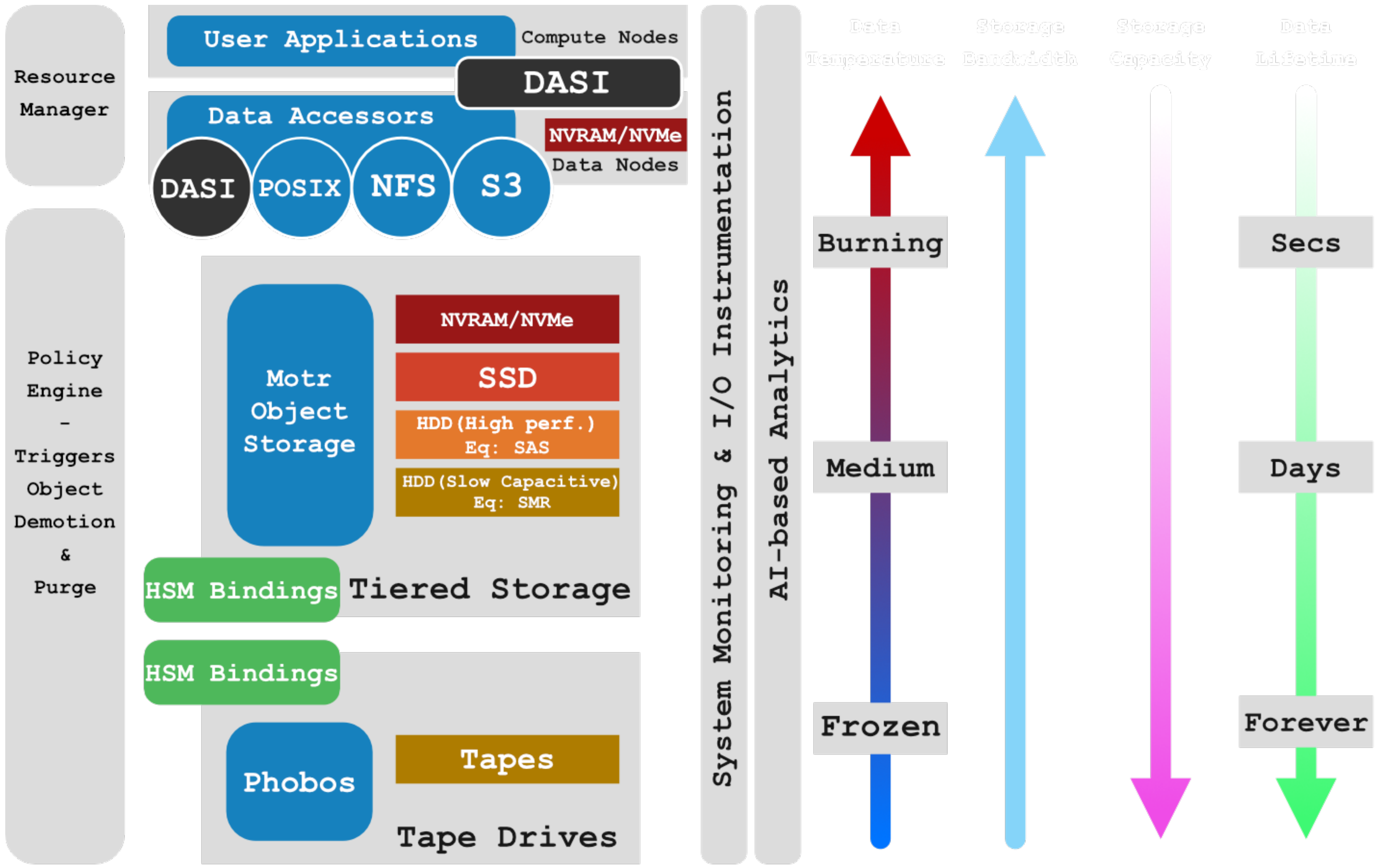
- This wastes time and energy, however traditional batch schedulers do not handle data, and cannot discern whether it will be used in the next stage or not, leaving no option other than this storing and reloading

- IO-SEA will allow users and applications to tag data to add information about potential future use and about the usage of potential results

Hierarchical Storage Management

- Hierarchical Storage Management (HSM) is a paradigm where data is stored on multiple “tiers” of storage ranging from high performance and high cost-per-byte NVMes to slower, high-capacity HDDs or even to tape store
- Data is moved according to its ”temperature”, which denotes the frequency of access, i.e. data that is accessed more often is hotter
- Data that is “frozen” and just needs to be kept for backups can be moved to a tape store to save space on the more expensive flash and disk tiers
- Data can be stored in the cheaper and less performant tiers when not in use and promoted to faster tiers when it needs to be accessed frequently, eg a simulation dataset

IO-SEA Project Architecture



iosea-project.eu

https://github.com/io-sea

Hestia

Hierarchical Storage Tiers Interface for Applications

A HSM wrapper for object stores, developed within the context of IO-SEA

OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

Hestia Implementation

ICHEC
Irish Centre for High-end Computing

Modern C++ Library – CMake Build

Standard Dependencies and Formats

- Curl, LibS3
- Interchangeable WebServer
- Yaml/JSON
- Spdlog/CLI11

CI with containerized builds

Open Source – MIT License

https://git.ichec.ie/io-sea-internal/hestia

Key Concepts

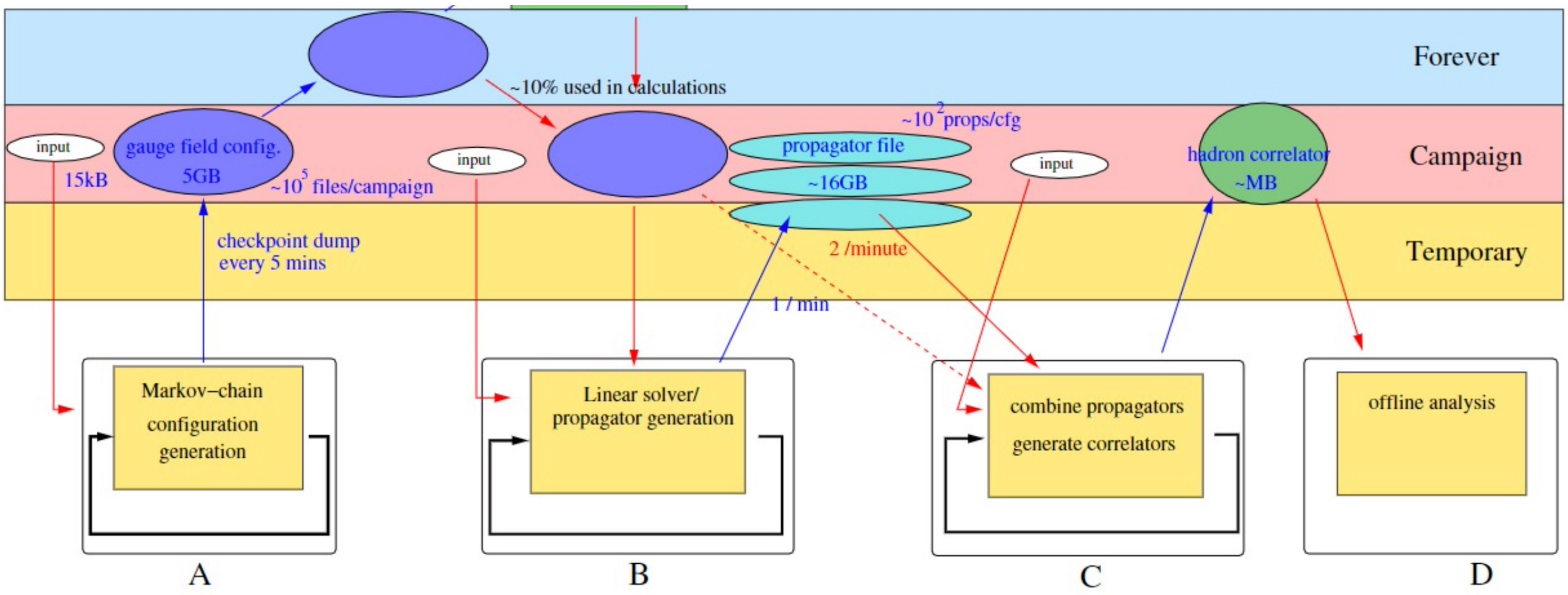
- Provides Object Store and Key-Value Store abstractions
  - Data blobs are kept in an Object Store, addressed with a unique identifier, eg 550e8400-e29b-41d4-a716-446655440000
  - Metadata is maintained in a Key-Value store
- Hestia provides these abstractions via a middleware functionality in a distributed storage setting, supporting:
  - Key-Value Store integration
  - Object Store integration
- Common API for tiered storage
- Allows for a distributed storage system
- CopyTool which allows the copying of data between different backends

Hestia hints

Hestia allows users to “tag” their data with hints as to how it will be used

These hints allow the policy engine to choose the correct location for the data

Small example of hints- LQCD workflow



Multi Backend Object Store

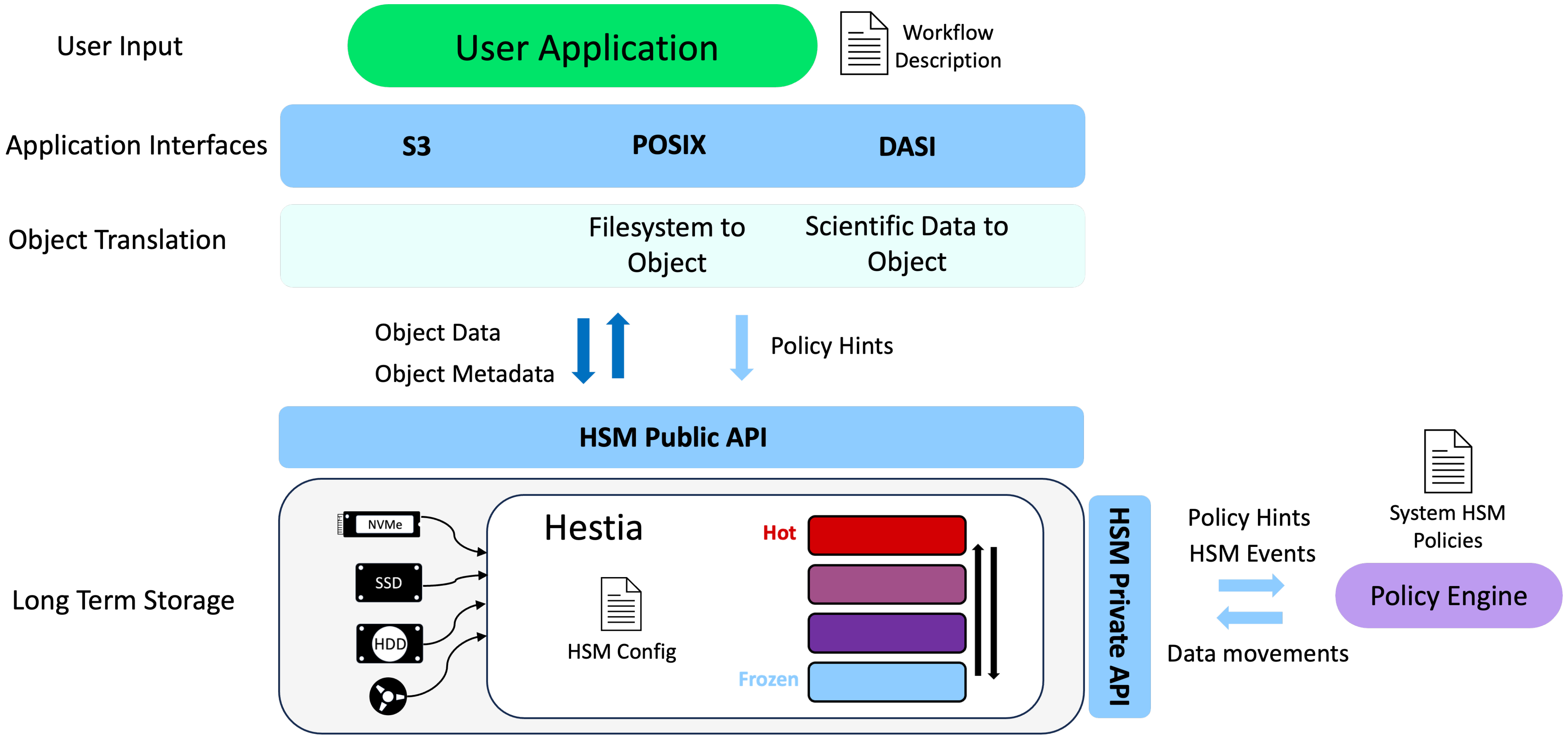
- Once the relationship between tiers and backends has been specified in the Hestia config file, Hestia can handle interfacing between multiple object store backends without further input from the user

- The following backends are currently supported:
  - FileObjectStoreClient
  - S3Client : A client for S3 endpoints - implemented using libs3
  - PhobosClient: A client for the Phobos library for tape-backed Object Stores
  - FileHsmObjectStoreClient : A HSM enabled version of the FileObjectStoreClient
  - MotrClient : A client for the HSM enabled Motr Object Store

- Hestia serves as a "glue" code in IO-SEA, putting a Hestia server in front of distributed storage systems allows them to communicate with each other via the Hestia S3 interface

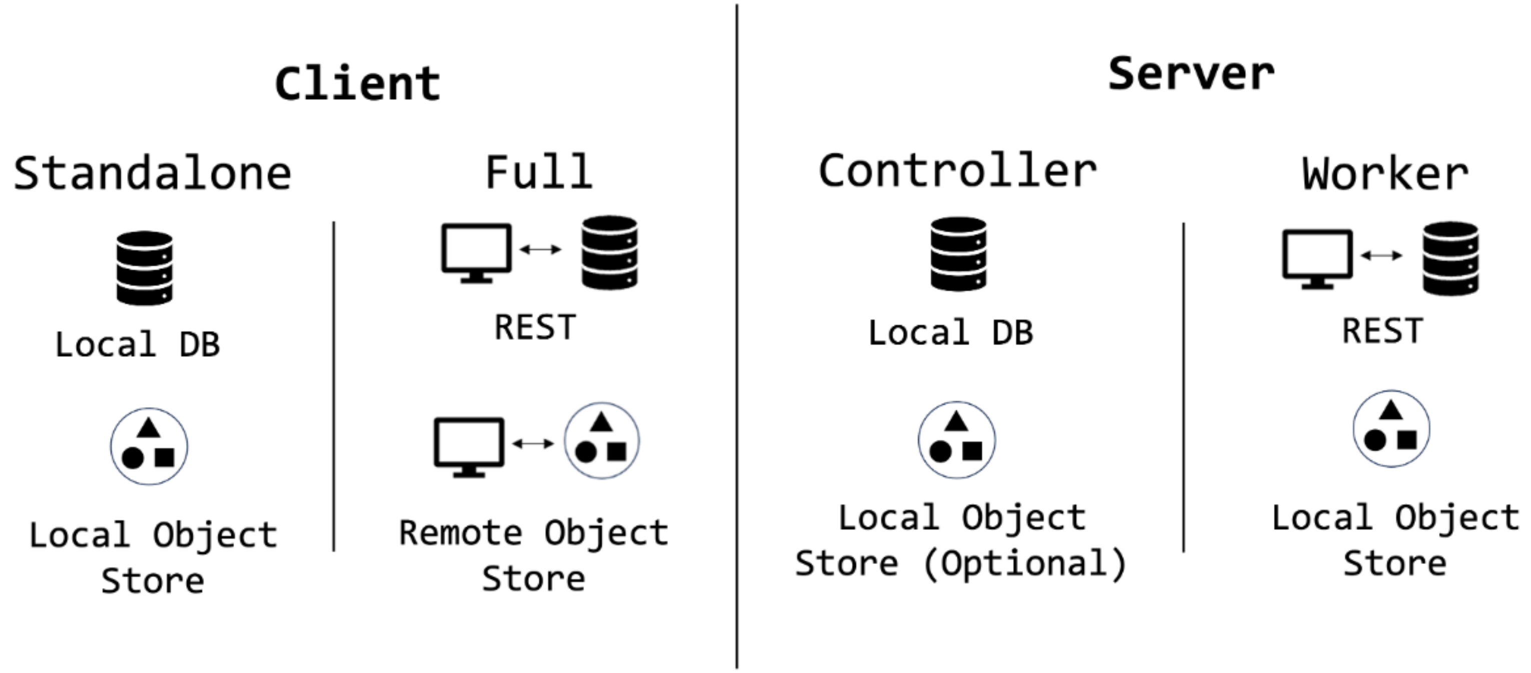
- Hestia currently implements a simple built in KVS and a Redis client

Hestia Architecture



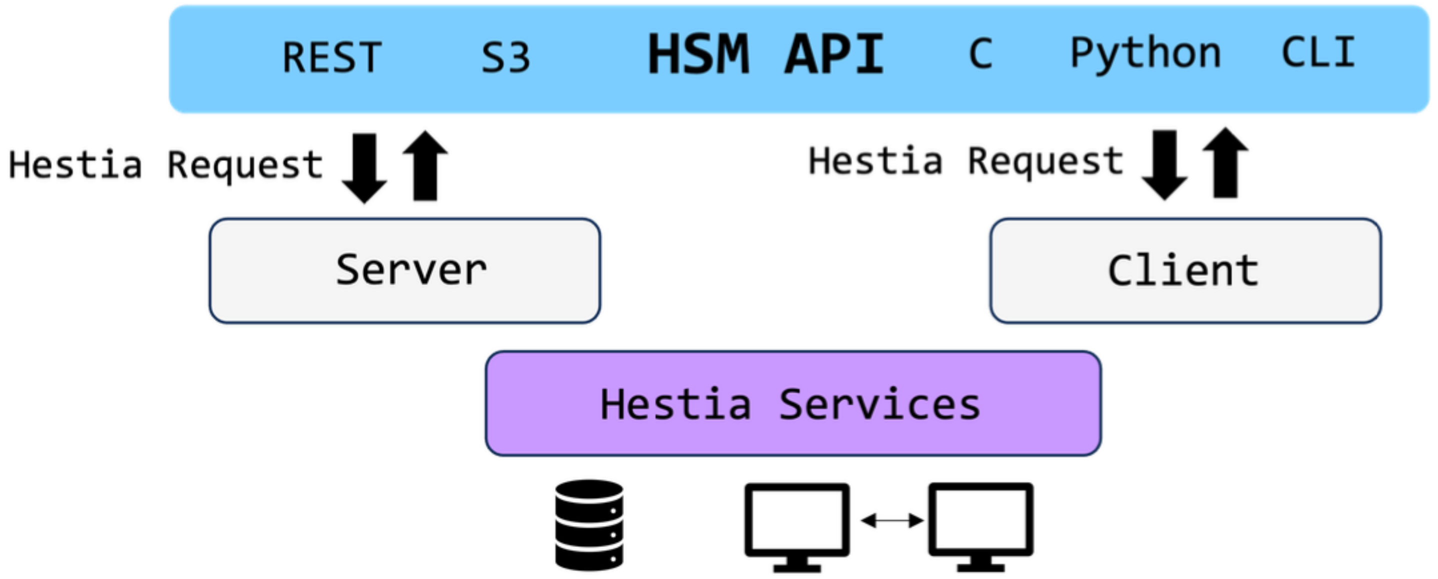
Hestia Runtime modes

- In all modes the application can read from a user provided config file on the system on launch and maintain a local cache.



Hestia Requests

API calls are translated into Hestia requests for a client or server to fulfil



Hestia API

Public HSM API

```
put: oid [data_source] [tier] [extent]
get: oid data_sink [tier] [extent]
set-attrs: oid attributes
get-attrs: oid > attributes
remove: oid
```

Private HSM API

```
copy/move: oid src_tier dst_tier [extent] ┐  HSM
release: oid tier [extent]                ┘ Action
locate: oid > tier
list-tiers: > tiers
list: tier > oids
```

```
oid | Object Identifier
```

Hestia Configuration

- Hestia takes a config file (yaml format) either as a standalone client or to start a server

- Using the config file it is possible to specify the server host and port, toggle the log level and event feed, and specify information about the tiers and backends

- Several example config files are given in the Hestia repo

- Launch the Hestia server using

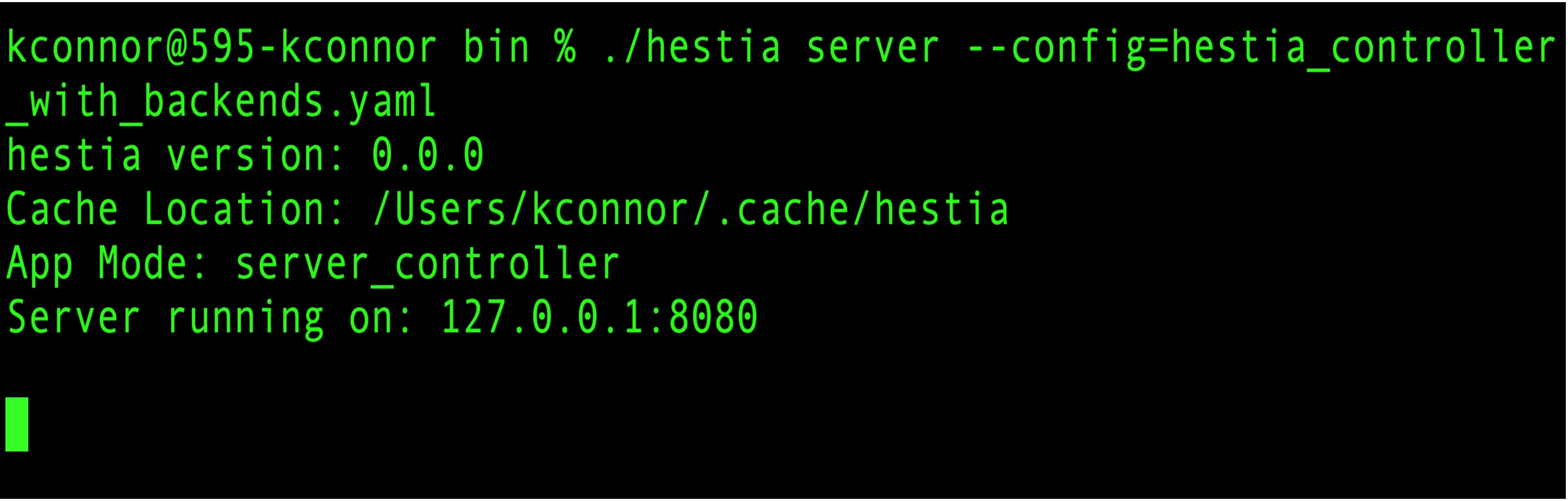
```
hestia server --config=example_config.yaml
```

```yaml
server:
  host: 127.0.0.1
  port: 8080
  tag: controller0

tiers:
  - id: "00000000-0000-0000-0000-000000000001"
    priority: 0
  - id: "00000000-0000-0000-0000-000000000002"
    priority: 1
  - id: "00000000-0000-0000-0000-000000000003"
    priority: 2
  - id: "00000000-0000-0000-0000-000000000004"
    priority: 3
  - id: "00000000-0000-0000-0000-000000000005"
    priority: 4

object_store_backends: # Specify which backends this node supports
  - backend_type: file_hsm # This is a POSIX filesystem store associated with 5 tiers
    tiers:
      ids: ["00000000-0000-0000-0000-000000000001",
            "00000000-0000-0000-0000-000000000002",
            "00000000-0000-0000-0000-000000000003",
            "00000000-0000-0000-0000-000000000004",
            "00000000-0000-0000-0000-000000000005"]
    config: # The root is the path to the store's cache, if relative it is relative to Hestia cache
      root: hsm_object_store_controller0

event_feed:
  active: true

logger:
  level: warn
```

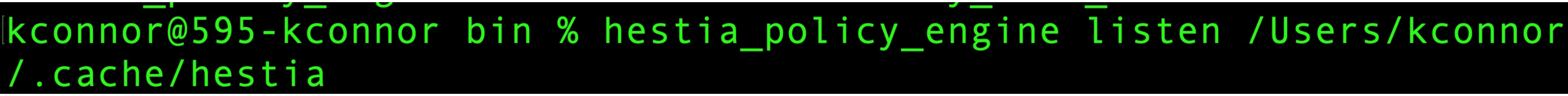
Hestia Trigger Migration hint

A hello world example

Hestia Server creation

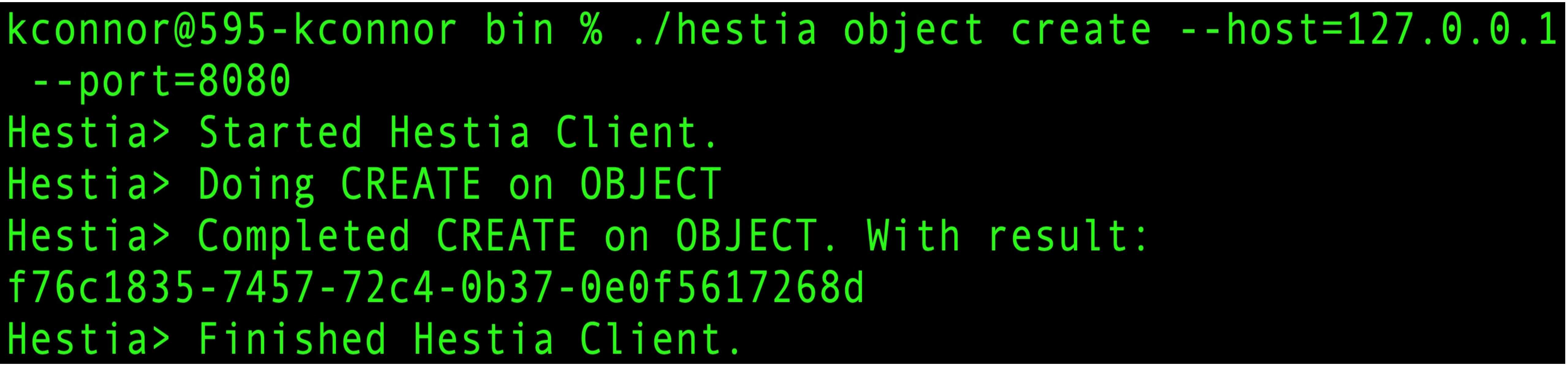


```
kconnor@595-kconnor bin % ./hestia server --config=hestia_controller
_with_backends.yaml
hestia version: 0.0.0
Cache Location: /Users/kconnor/.cache/hestia
App Mode: server_controller
Server running on: 127.0.0.1:8080
```



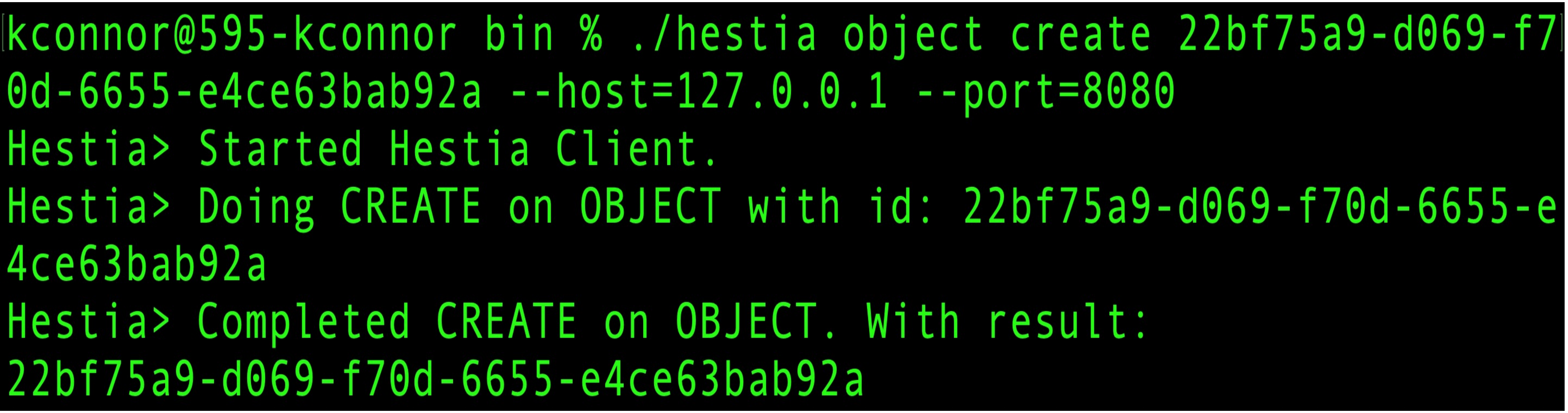
```
kconnor@595-kconnor bin % hestia_policy_engine listen /Users/kconnor
/.cache/hestia
```

Hestia object create



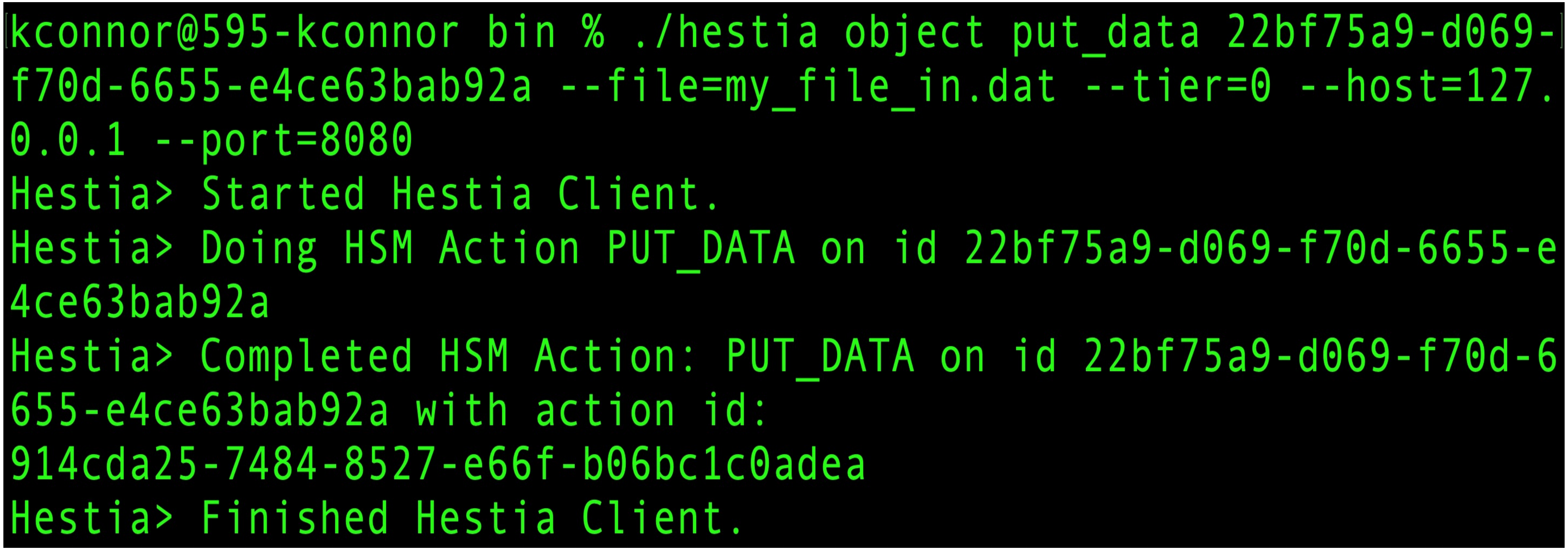
```
kconnor@595-kconnor bin % ./hestia object create --host=127.0.0.1
 --port=8080
Hestia> Started Hestia Client.
Hestia> Doing CREATE on OBJECT
Hestia> Completed CREATE on OBJECT. With result:
f76c1835-7457-72c4-0b37-0e0f5617268d
Hestia> Finished Hestia Client.
```

Hestia object create



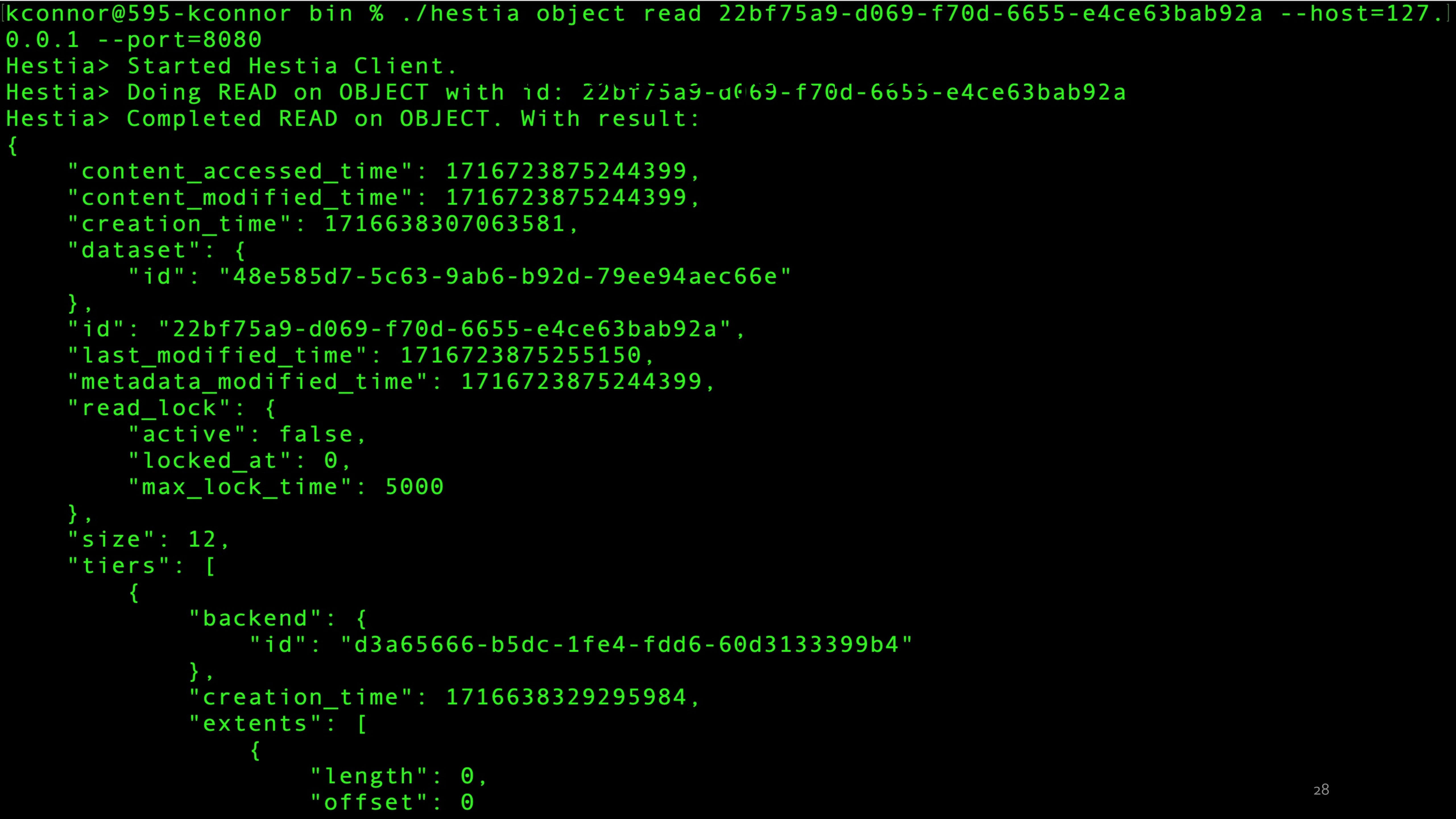
```
kconnor@595-kconnor bin % ./hestia object create 22bf75a9-d069-f70d-6655-e4ce63bab92a --host=127.0.0.1 --port=8080
Hestia> Started Hestia Client.
Hestia> Doing CREATE on OBJECT with id: 22bf75a9-d069-f70d-6655-e4ce63bab92a
Hestia> Completed CREATE on OBJECT. With result:
22bf75a9-d069-f70d-6655-e4ce63bab92a
```

Hestia object put_data



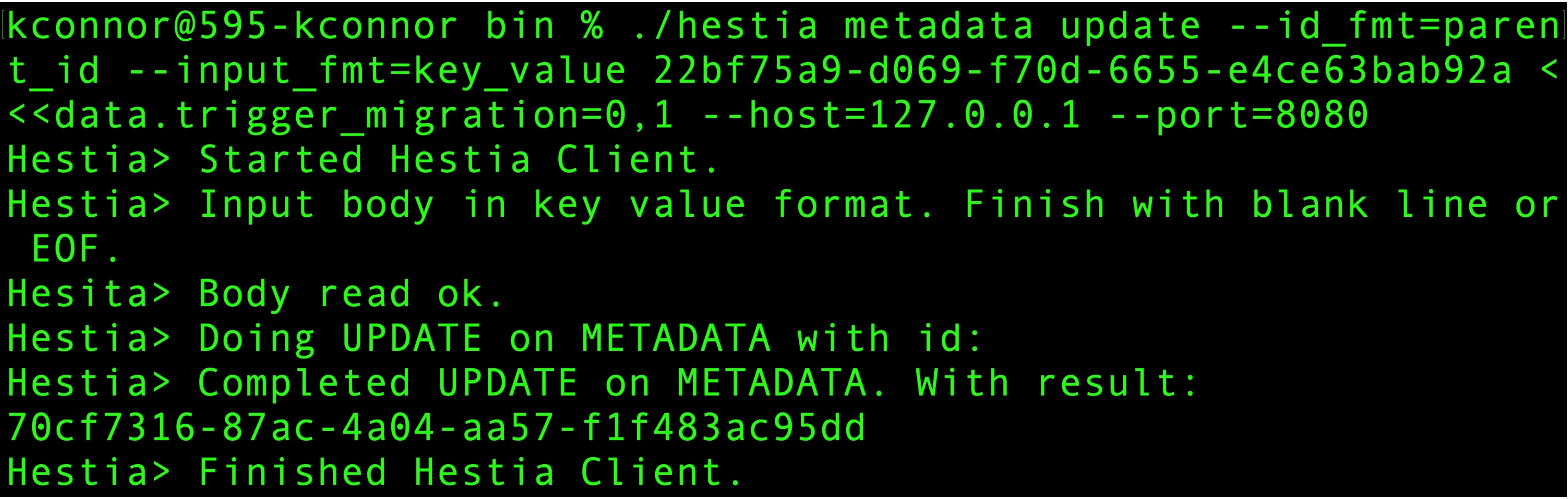
```
kconnor@595-kconnor bin % ./hestia object put_data 22bf75a9-d069-
f70d-6655-e4ce63bab92a --file=my_file_in.dat --tier=0 --host=127.
0.0.1 --port=8080
Hestia> Started Hestia Client.
Hestia> Doing HSM Action PUT_DATA on id 22bf75a9-d069-f70d-6655-e
4ce63bab92a
Hestia> Completed HSM Action: PUT_DATA on id 22bf75a9-d069-f70d-6
655-e4ce63bab92a with action id:
914cda25-7484-8527-e66f-b06bc1c0adea
Hestia> Finished Hestia Client.
```

```
kconnor@595-kconnor bin % ./hestia action read 914cda25-7484-8527-e66f-b06bc1c0adea --host=127.
0.0.1 --port=8080
Hestia> Started Hestia Client.
Hestia> Doing READ on ACTION with id: 914cda25-7484-8527-e66f-b06bc1c0adea
Hestia> Completed READ on ACTION. With result:
{
    "acl": {
        "group": [],
        "user": []
    },
    "action": "put_data",
    "created_by": {
        "id": "339c7602-16d7-3ac4-2029-d1d45f8e3a84"
    },
    "creation_time": 1716638329284369,
    "id": "914cda25-7484-8527-e66f-b06bc1c0adea",
    "is_request": false,
    "last_modified_time": 1716638329284369,
    "offset": 0,
    "progess_interval": 0,
    "source_tier": 0,
    "status": "finished_ok",
    "subject": "object",
    "subject_key": "22bf75a9-d069-f70d-6655-e4ce63bab92a",
    "target_tier": 0,
    "to_transfer": 12,
    "transferred": 12,
    "type": "action"
}
Hestia> Finished Hestia Client.
```



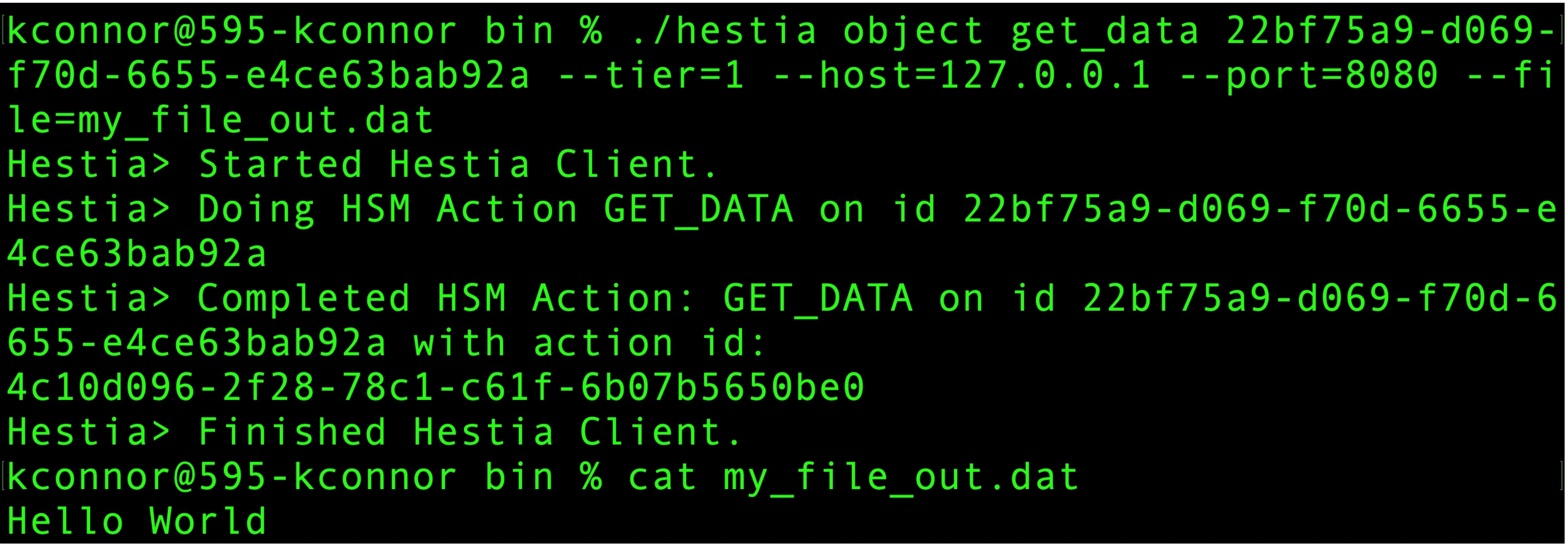
```
kconnor@595-kconnor bin % ./hestia object read 22bf75a9-d069-f70d-6655-e4ce63bab92a --host=127.
0.0.1 --port=8080
Hestia> Started Hestia Client.
Hestia> Doing READ on OBJECT with id: 22bf75a9-d069-f70d-6655-e4ce63bab92a
Hestia> Completed READ on OBJECT. With result:
{
    "content_accessed_time": 1716723875244399,
    "content_modified_time": 1716723875244399,
    "creation_time": 1716638307063581,
    "dataset": {
        "id": "48e585d7-5c63-9ab6-b92d-79ee94aec66e"
    },
    "id": "22bf75a9-d069-f70d-6655-e4ce63bab92a",
    "last_modified_time": 1716723875255150,
    "metadata_modified_time": 1716723875244399,
    "read_lock": {
        "active": false,
        "locked_at": 0,
        "max_lock_time": 5000
    },
    "size": 12,
    "tiers": [
        {
            "backend": {
                "id": "d3a65666-b5dc-1fe4-fdd6-60d3133399b4"
            },
            "creation_time": 1716638329295984,
            "extents": [
                {
                    "length": 0,
                    "offset": 0
```

Hestia metadata update- adding hints



```
kconnor@595-kconnor bin % ./hestia metadata update --id_fmt=paren
t_id --input_fmt=key_value 22bf75a9-d069-f70d-6655-e4ce63bab92a <
<<data.trigger_migration=0,1 --host=127.0.0.1 --port=8080
Hestia> Started Hestia Client.
Hestia> Input body in key value format. Finish with blank line or
 EOF.
Hesita> Body read ok.
Hestia> Doing UPDATE on METADATA with id:
Hestia> Completed UPDATE on METADATA. With result:
70cf7316-87ac-4a04-aa57-f1f483ac95dd
Hestia> Finished Hestia Client.
```

Hestia get_data

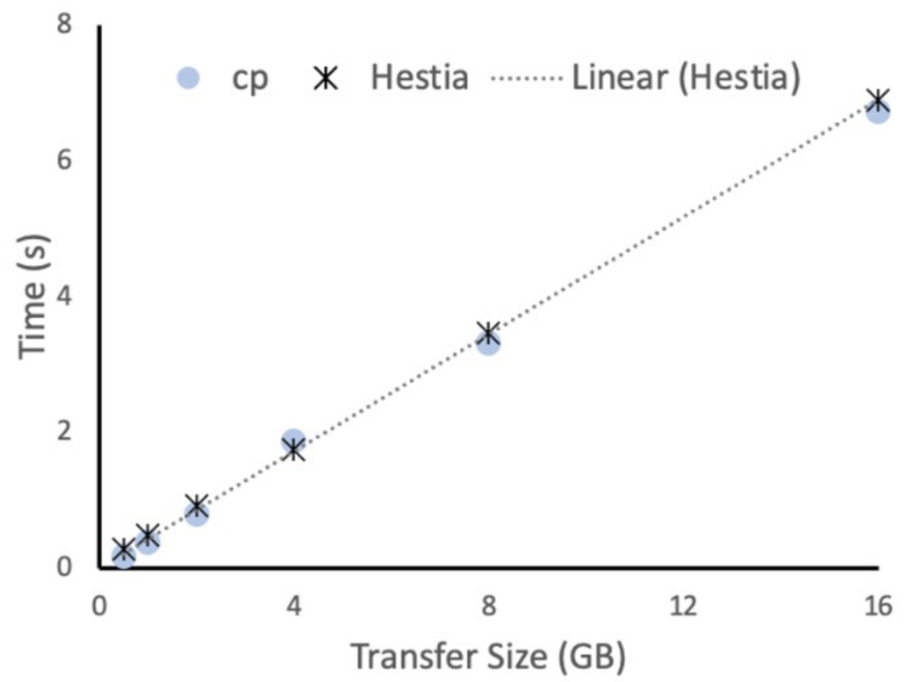


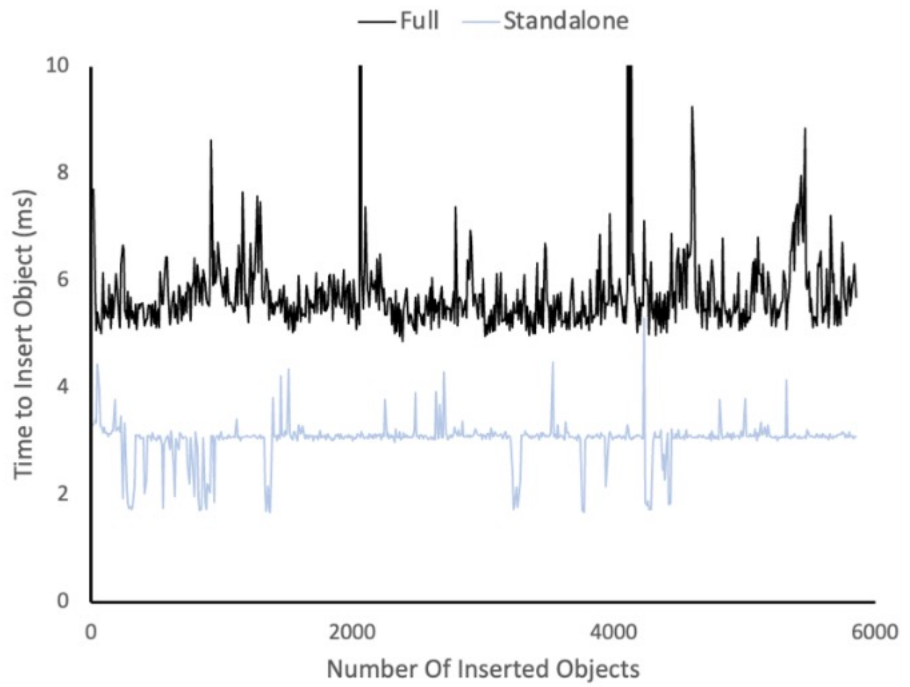
```
kconnor@595-kconnor bin % ./hestia object get_data 22bf75a9-d069-
f70d-6655-e4ce63bab92a --tier=1 --host=127.0.0.1 --port=8080 --fi
le=my_file_out.dat
Hestia> Started Hestia Client.
Hestia> Doing HSM Action GET_DATA on id 22bf75a9-d069-f70d-6655-e
4ce63bab92a
Hestia> Completed HSM Action: GET_DATA on id 22bf75a9-d069-f70d-6
655-e4ce63bab92a with action id:
4c10d096-2f28-78c1-c61f-6b07b5650be0
Hestia> Finished Hestia Client.
kconnor@595-kconnor bin % cat my_file_out.dat
Hello World
```

Ollscoil na Gaillimhe
University of Galway

Hestia performance

ICHEC
Irish Centre for High-end Computing





All performance tests performed on Mac with SSD

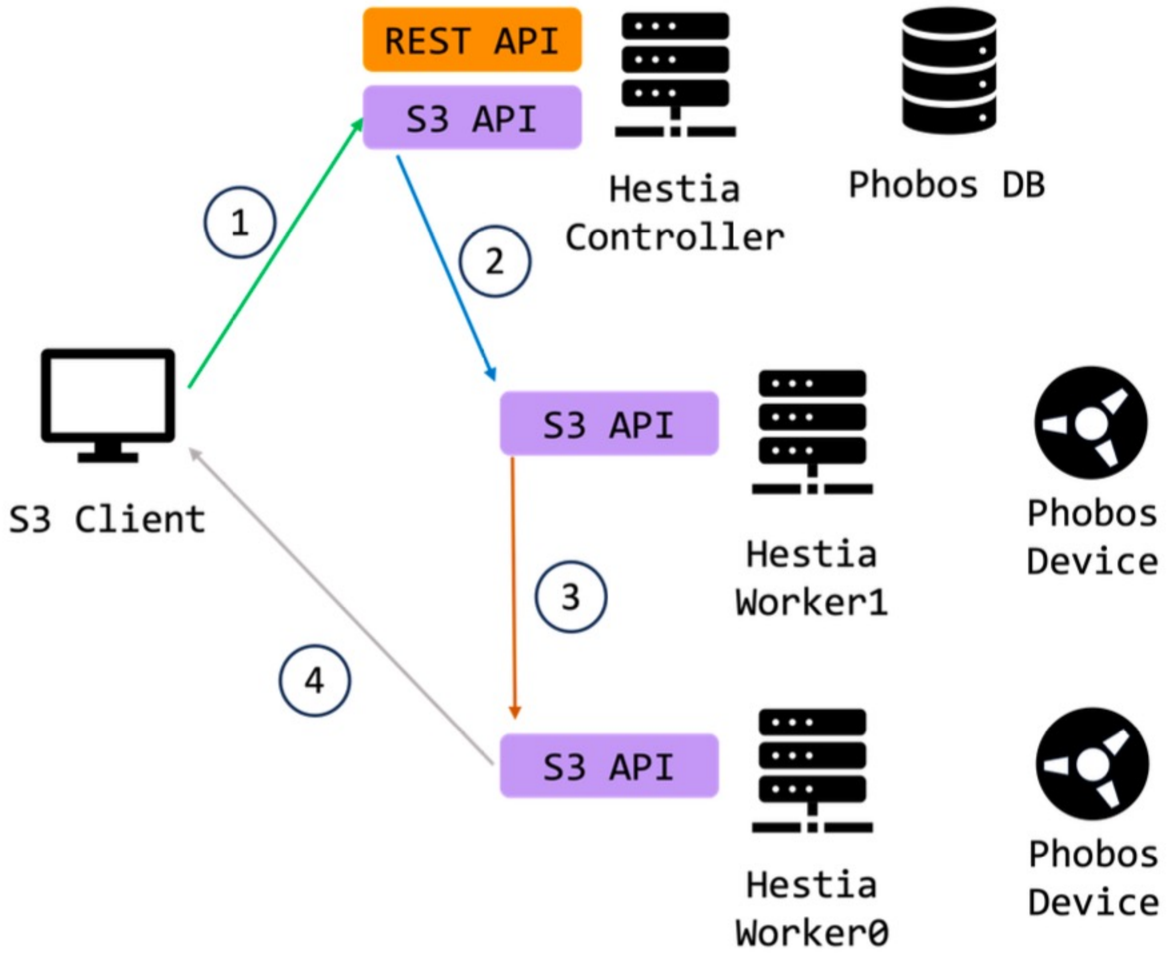
Integration with other IO-SEA software

- Hestia has been deployed on the DEEP prototype system in conjunction with other IO-SEA software components

- Hestia interfaces with other pieces of IO-SEA software developed by other partners, such as the Ganesha Request Handler, the RobinHood policy engine, and DASI

- Hestia is compatible with Phobos from the EUPEX (European Pilot for Exascale) software bundle

Hestia with Phobos



Potential future work

- Hestia is still in beta- some aspects such as object locking and more comprehensive error checking have yet to be fully implemented
- Complete and cleaner RPM packaging
- Further S3 support- current S3 interface is basic and allows for the creation and deletion of objects and the creation of buckets
- Parts of objects- random access read and write
- More testing, performance testing, documentation and demos

References and Links

- *IO-SEA website: https://iosea-project.eu/*
- *IO-SEA GitHub: https://github.com/io-sea*
- *Hestia: https://git.ichec.ie/io-sea-internal/hestia*
- *Phobos: https://github.com/cea-hpc/phobos*
- *Motr: https://github.com/Seagate/cortx-motr*