TECHPARK SÜDTIROL / ALTO ADIGE

SFSCON

Optimizing Cloud Compute Resources with Spare Cores

Gergely Daróczi
Spare Cores Team

Slides: sparecores.com/talks

<spare_cores>

>>> from sparecores import why

>>> from sparecores import why

Data Science / Machine Learning batch jobs:

>>> from sparecores import why

Data Science / Machine Learning batch jobs:

- run SQL

>>> from sparecores import why

Data Science / Machine Learning batch jobs:

- run SQL
- run R or Python script

>>> from sparecores import why

Data Science / Machine Learning batch jobs:

- run SQL
- run R or Python script
  - train a simple model, reporting, API integrations etc.

>>> from sparecores import why

Data Science / Machine Learning batch jobs:

- run SQL
- run R or Python script
  - train a simple model, reporting, API integrations etc.
  - train hierarchical models/GBMs/neural nets etc.

>>> from sparecores import why

Data Science / Machine Learning batch jobs:

- run SQL
- run R or Python script
  - train a simple model, reporting, API integrations etc.
  - train hierarchical models/GBMs/neural nets etc.

Scaling (DS) infrastructure.

<spare_cores>

>>> from sparecores import why

Databases | Shares | Data Marketplace | Warehouses | Worksheets | History | Partner Connect | Help | Snowsight | Gergely Daroczi DATASCIENCE_TEAM

Find database objects

Starting with...

Run | All Queries | Saved 25 seconds ago

DATASCIENCE_T... | S1_DS_ADHOC (L) | Select Database | Select Schema

```
1662 SELECT *
1663 FROM
1664 WHERE job_name =                                   AND build_number =
1665 ORDER BY "timestamp";
```

Results | Data Preview | Open History

Query ID | SQL | 233ms | 89 rows

Filter result... | Copy | Columns

Row	JOB_NAME	CPU_PERC	MEM	timestamp	BUILD_NUMBER
1		107	56303616	2021-11-02 00:11:30.000	scheduled__2021-11-01T22:07:00+00:00
2		21	114376704	2021-11-02 00:11:32.000	scheduled__2021-11-01T22:07:00+00:00
3		95	146558976	2021-11-02 00:11:34.000	scheduled__2021-11-01T22:07:00+00:00
4		0	147136512	2021-11-02 00:11:36.000	scheduled__2021-11-01T22:07:00+00:00
5		0	147136512	2021-11-02 00:11:38.000	scheduled__2021-11-01T22:07:00+00:00
6		119	153939968	2021-11-02 00:11:40.000	scheduled__2021-11-01T22:07:00+00:00
7		92	183046144	2021-11-02 00:11:42.000	scheduled__2021-11-01T22:07:00+00:00
8		100	189616128	2021-11-02 00:11:44.000	scheduled__2021-11-01T22:07:00+00:00
9		3	196734976	2021-11-02 00:11:46.000	scheduled__2021-11-01T22:07:00+00:00
10		111	198520832	2021-11-02 00:11:48.000	scheduled__2021-11-01T22:07:00+00:00
11		106	419045376	2021-11-02 00:11:50.000	scheduled__2021-11-01T22:07:00+00:00
12		83	620335104	2021-11-02 00:11:52.000	scheduled__2021-11-01T22:07:00+00:00
13		98	839749632	2021-11-02 00:11:54.000	scheduled__2021-11-01T22:07:00+00:00
14		99	1043267584	2021-11-02 00:11:56.000	scheduled__2021-11-01T22:07:00+00:00
15		100	1292713984	2021-11-02 00:11:58.000	scheduled__2021-11-01T22:07:00+00:00
16		100	1424506880	2021-11-02 00:12:00.000	scheduled__2021-11-01T22:07:00+00:00
17		100	1599643648	2021-11-02 00:12:02.000	scheduled__2021-11-01T22:07:00+00:00
18		101	1783463936	2021-11-02 00:12:04.000	scheduled__2021-11-01T22:07:00+00:00
19		99	1776459776	2021-11-02 00:12:06.000	scheduled__2021-11-01T22:07:00+00:00
20		87	1382346752	2021-11-02 00:12:08.000	scheduled__2021-11-01T22:07:00+00:00

Preview Data

489,873,031 rows 5.9 GB

Cluster by

-

Columns	Data Type
JOB_NAME	VARCHAR(16777216)
CPU_PERC	NUMBER(38,0)
MEM	NUMBER(38,0)
timestamp	TIMESTAMP_NTZ(9)
BUILD_NUMBER	VARCHAR(16777216)

>>> from sparecores import why

Databases | Shares | Data Marketplace | Warehouses | Worksheets | History | Partner Connect | Help | Snowsight | Gergely Daroczi DATASCIENCE_TEAM

Find database objects

Starting with...

DATA_SCIENCE

Run | All Queries | Saved 50 seconds ago | DATASCIENCE_T... | S1_DS_ADHOC (L) | Select Database | Select Schema

```
SELECT *
FROM
WHERE "start"::DATE = CURRENT_DATE - 1;
```

Results | Data Preview | Open History

Query ID | SQL | 345ms | 3,854 rows

Filter result... | Copy | Columns

Row	JOB_NAME	start	end	DURATION	EXIT_CODE	REMOTE	COST	WRITE_LATENC	WRITE_IOPS	VCPU	PHYS_MEM	INSTANCE_TYPI	BUILD_NUMI
1		2021-11-02 00:00:03.000	2021-11-02 00:02:02.000	119	0	FALSE	NULL	NULL	NULL	16	NULL	m5.4xlarge	281118
2		2021-11-02 00:02:33.000	2021-11-02 00:04:18.000	105	0	FALSE	NULL	NULL	NULL	16	NULL	m5.4xlarge	20065
3		2021-11-02 00:02:58.000	2021-11-02 00:03:27.000	29	0	FALSE	NULL	NULL	NULL	16	NULL	m5.4xlarge	23014
4		2021-11-02 00:00:02.000	2021-11-02 00:00:54.000	52	0	FALSE	NULL	NULL	NULL	16	NULL	m5.4xlarge	345244
5		2021-11-02 00:00:03.000	2021-11-02 00:01:45.000	102	0	FALSE	NULL	NULL	NULL	16	NULL	m5.4xlarge	1193
6		2021-11-02 00:00:03.000	2021-11-02 00:00:40.000	37	0	FALSE	NULL	NULL	NULL	16	NULL	m5.4xlarge	795
7		2021-11-02 00:01:02.000	2021-11-02 00:01:15.000	13	0	FALSE	NULL	NULL	NULL	16	NULL	m5.4xlarge	422230
8		2021-11-02 00:00:03.000	2021-11-02 00:00:28.000	25	0	FALSE	NULL	NULL	NULL	16	NULL	m5.4xlarge	13153
9		2021-11-02 00:01:02.000	2021-11-02 00:01:26.000	24	0	FALSE	NULL	NULL	NULL	16	NULL	m5.4xlarge	29138
10		2021-11-02 00:05:02.000	2021-11-02 00:05:25.000	23	0	FALSE	NULL	NULL	NULL	16	NULL	m5.4xlarge	13154
11		2021-11-02 00:05:02.000	2021-11-02 00:05:51.000	49	0	FALSE	NULL	NULL	NULL	16	NULL	m5.4xlarge	345245
12		2021-11-02 00:04:02.000	2021-11-02 00:05:15.000	73	0	FALSE	NULL	NULL	NULL	16	NULL	m5.4xlarge	33560
13		2021-11-02 00:05:03.000	2021-11-02 00:05:38.000	35	0	FALSE	NULL	NULL	NULL	16	NULL	m5.4xlarge	95623
14		2021-11-02 00:06:02.000	2021-11-02 00:06:14.000	12	0	FALSE	NULL	NULL	NULL	16	NULL	m5.4xlarge	422231
15		2021-11-02 00:05:03.000	2021-11-02 00:06:52.000	109	0	FALSE	NULL	NULL	NULL	16	NULL	m5.4xlarge	281119
16		2021-11-02 00:06:45.000	2021-11-02 00:08:04.000	79	0	FALSE	NULL	NULL	NULL	16	NULL	m5.4xlarge	23478
17		2021-11-02 00:11:04.000	2021-11-02 00:14:29.000	205	0	TRUE	0.00207601...	37.152780476	26.8924656...	2	NULL	m2.xlarge	scheduled
18		2021-11-02 00:14:02.000	2021-11-02 00:14:33.000	31	0	FALSE	NULL	NULL	NULL	16	NULL	m5.4xlarge	82441
19		2021-11-02 00:08:02.000	2021-11-02 00:08:38.000	36	0	FALSE	NULL	NULL	NULL	16	NULL	m5.4xlarge	2021
20		2021-11-02 00:11:02.000	2021-11-02 00:14:39.000	217	0	FALSE	NULL	NULL	NULL	16	NULL	m5.4xlarge	16508

Preview Data

8,329,043 rows 286.2 MB

Cluster by

-

Columns	Data Type
JOB_NAME	VARCHAR(16777216)
start	TIMESTAMP_NTZ(9)
end	TIMESTAMP_NTZ(9)
DURATION	NUMBER(38,0)
EXIT_CODE	NUMBER(38,0)
REMOTE	BOOLEAN
COST	FLOAT
WRITE_LATENCY	FLOAT
WRITE_IOPS	FLOAT
VCPU	NUMBER(38,0)

>>> from sparecores import why

>>> from sparecores import why

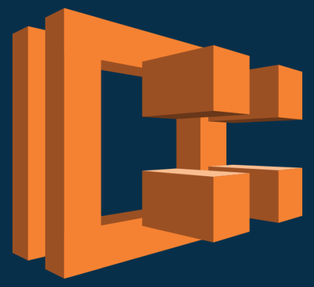
AWS ECS

>>> from sparecores import why

AWS ECS

AWS Batch

>>> from sparecores import why

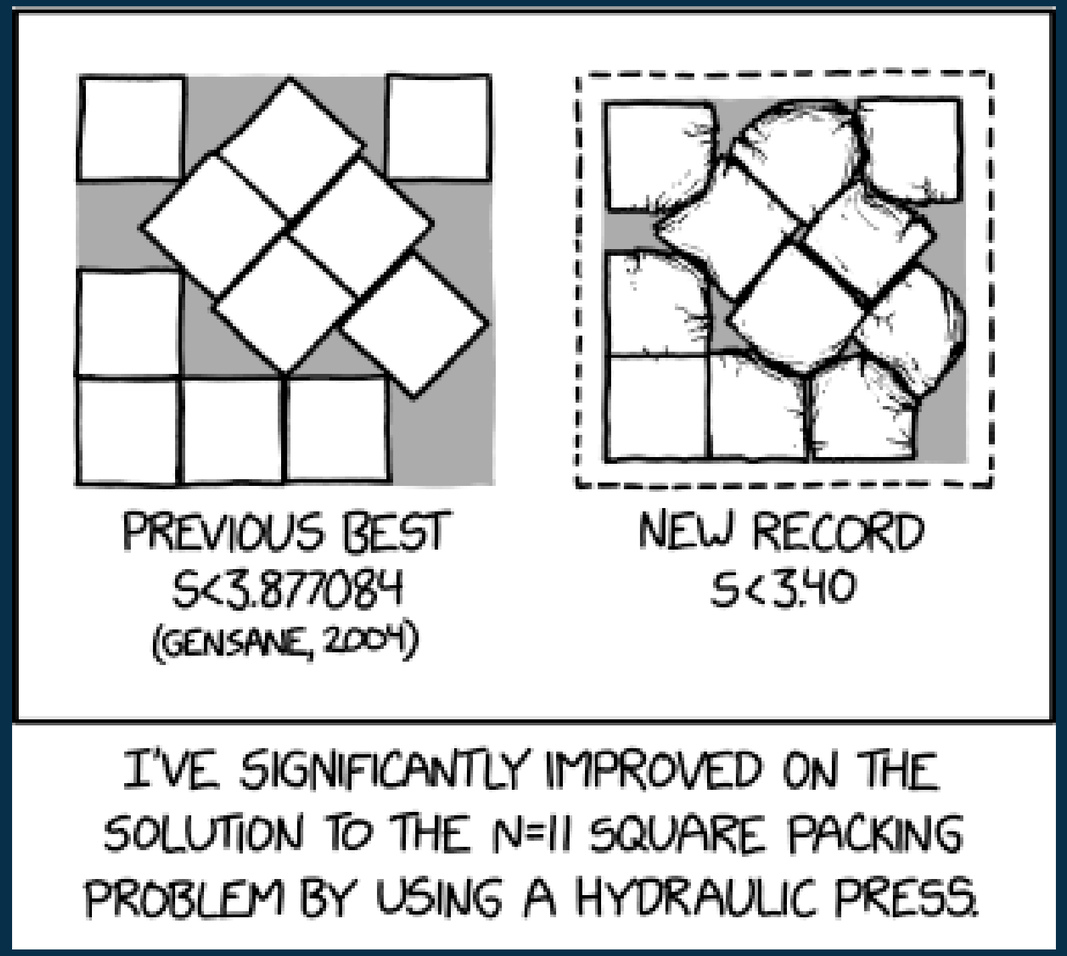


AWS ECS

AWS Batch

Kubernetes

>>> from sparecores import why



Source: xkcd

>>> from sparecores import why

```
Started by timer
Running as SYSTEM
[EnvInject] - Loading node environment variables.
Building remotely on Remote jobs (jobs Remote) in workspace /var/lib/jenkins-remote-jobs/workspace/[redacted]
No emails were triggered.
[[redacted]] $ /bin/sh -xe /tmp/jenkins3336101352682459 98.sh
+ docker-run.py --timeout 90m --remote --remote-filter vcpu>=32 --remote-filter memory>=200 Rscript [redacted] --env prod
INFO:root:Loaded maximum memory usage (325.79 GiBs)
INFO:root:Loaded maximum CPU usage (9600.00%/96 vcpu)
=> Run-time statistics for max CPU usage of 96
  instanceType  benchmark_score  benchmark_cpus  duration_max  max_cpu
2  m4.16xlarge        6907.9500              64          2314     6400
1     m5.metal       12147.0400              96          2136     9600
0    m5d.metal       12142.0700              96          2066     9600
=> Best perf/price on-demand instance: m5.metal vcpu:96, mem:384.0, price(spotmax):4.608
=> Minimum score needed for runtime target of 4320s is: 6006.03644444
=> Starting m5.metal spot:True price:1.0818 perf/price:11228.5450176 score:12147.04 vcpu:96 mem:384.0 AZ:us-west-2c AMI:ami-026eda26a9745c6c7
=> Insufficient capacity in us-west-2c
=> Starting m5.metal spot:True price:1.0818 perf/price:11228.5450176 score:12147.04 vcpu:96 mem:384.0 AZ:us-west-2b AMI:ami-026eda26a9745c6c7
=> Insufficient capacity in us-west-2b
=> Starting m5d.metal spot:True price:1.0818 perf/price:11223.9508227 score:12142.07 vcpu:96 mem:384.0 AZ:us-west-2c AMI:ami-026eda26a9745c6c7
=> Insufficient capacity in us-west-2c
=> Starting m5d.metal spot:True price:1.0818 perf/price:11223.9508227 score:12142.07 vcpu:96 mem:384.0 AZ:us-west-2a AMI:ami-026eda26a9745c6c7
=> Insufficient capacity in us-west-2a
=> Starting m5d.metal spot:True price:1.0818 perf/price:11223.9508227 score:12142.07 vcpu:96 mem:384.0 AZ:us-west-2b AMI:ami-026eda26a9745c6c7
=> Insufficient capacity in us-west-2b
=> Starting r5.metal spot:True price:1.1333 perf/price:10715.4769258 score:12143.85 vcpu:96 mem:768.0 AZ:us-west-2c AMI:ami-026eda26a9745c6c7
=> Insufficient capacity in us-west-2c
=> Starting r5.metal spot:True price:1.1333 perf/price:10715.4769258 score:12143.85 vcpu:96 mem:768.0 AZ:us-west-2b AMI:ami-026eda26a9745c6c7
=> Insufficient capacity in us-west-2b
=> Starting r5.metal spot:True price:1.1333 perf/price:10715.4769258 score:12143.85 vcpu:96 mem:768.0 AZ:us-west-2a AMI:ami-026eda26a9745c6c7
=> Insufficient capacity in us-west-2a
=> Starting m5.metal spot:True price:1.143 perf/price:10627.3315836 score:12147.04 vcpu:96 mem:384.0 AZ:us-west-2a AMI:ami-026eda26a9745c6c7
=> Insufficient capacity in us-west-2a
=> Starting m4.16xlarge spot:True price:0.8608 perf/price:8025.0348513 score:6907.95 vcpu:64 mem:256.0 AZ:us-west-2a AMI:ami-026eda26a9745c6c7
=> Insufficient capacity in us-west-2a
=> Starting m4.16xlarge spot:True price:0.8608 perf/price:8025.0348513 score:6907.95 vcpu:64 mem:256.0 AZ:us-west-2b AMI:ami-026eda26a9745c6c7
=> Waiting for instance i-073b52fdfb5cff452 to be ready ([redacted])
```

>>> from sparecores import why

```
DEBUG [2019-11-07 09:26:35] s3://[illegible]
Exit code: 0
Maximum memory usage: 206.1GiB
Average memory usage: 117.2GiB
Average CPU usage: 3325%
Max CPU usage: 6489%
Estimated EC2 cost: $0.56
Shutdown scheduled for Thu 2019-11-07 09:28:01 UTC, use 'shutdown -c' to cancel.
Remote exit code: 0
[Slack Notifications] found #4482 as previous completed, non-aborted build
No emails were triggered.
Finished: SUCCESS
```

>>> from sparecores import why

```
DEBUG [2019-11-07 09:26:35] s3://[redacted]
Exit code: 0
Maximum memory usage: 206.1GiB
Average memory usage: 117.2GiB
Average CPU usage: 3325%
Max CPU usage: 6489%
Estimated EC2 cost: $0.56
Shutdown scheduled for Thu 2019-11-07 09:28:01 UTC, use 'shutdown -c' to cancel.
Remote exit code: 0
[Slack Notifications] found #4482 as previous completed, non-aborted build
No emails were triggered.
Finished: SUCCESS
```

```
Error in mcfork(detached) :
  unable to fork, possible reason: Cannot allocate memory
Calls: mclapply_timeout -> mcparallel -> mcfork
Execution halted
Exit code: 1
Maximum memory usage: 341.1GiB
Average memory usage: 96.6GiB
Average CPU usage: 2251%
Max CPU usage: 9713%
Estimated EC2 cost: $0.56
Shutdown scheduled for Sun 2019-11-10 21:18:02 UTC, use 'shutdown -c' to cancel.
Remote exit code: 1
Build step 'Execute shell' marked build as failure
[Slack Notifications] found #4503 as previous completed, non-aborted build
Email was triggered for: Failure - Any
Sending email for trigger: Failure - Any
Sending email to: [redacted]
Finished: FAILURE
```

<spare_cores>

>>> from sparecores import why

Other use-cases:

- stats/ML/AI model training,
- ETL pipelines,
- traditional CI/CD workflows for compiling and testing software,
- building Docker images,
- rendering images and videos,
- etc.

>>> from sparecores import why

MIKATA

Open-source Python and R Tools for Data Science in Production

logger R
Maintained by daroczig

A lightweight, modern and flexible, log4j and futile.logger inspired logging utility for R

1629 KB **219** STARS **31** FORKS **27** ISSUES

botor R
Maintained by daroczig

Reticulate wrapper on 'boto3' with convenient helper functions -- aka "boto fo(u)r R"

414 KB **28** STARS **5** FORKS **3** ISSUES

FilterNet Python
Maintained by Mikata-Project

A PyTorch ensemble neural network model used for time series analysis.

16 KB **55** STARS **24** FORKS **0** ISSUES

cloudperf Python
Maintained by bra-fsn

Measuring the relative performance of cloud resources

473 KB **7** STARS **3** FORKS **3** ISSUES

fbRads R
Maintained by daroczig

Analyze and manage Facebook ads from R using this client library to access their Marketing APIs

407 KB **141** STARS **61** FORKS **16** ISSUES

grpc C++
Maintained by nfultz

gRPC clients and servers in R

158 KB **62** STARS **21** FORKS **11** ISSUES

<spare_cores>

>>> from sparecores import intro

- Open-source tools, database schemas and documentation to inspect and inventory cloud vendors and their compute resource offerings.

>>> from sparecores import intro

- Open-source tools, database schemas and documentation to inspect and inventory cloud vendors and their compute resource offerings.
- Managed infrastructure, databases, APIs, SDKs, and web applications to make these data sources publicly accessible.

>>> from sparecores import intro

- Open-source tools, database schemas and documentation to inspect and inventory cloud vendors and their compute resource offerings.
- Managed infrastructure, databases, APIs, SDKs, and web applications to make these data sources publicly accessible.
- Helpers to start and manage instances in your own environment.

>>> from sparecores import intro

- Open-source tools, database schemas and documentation to inspect and inventory cloud vendors and their compute resource offerings.
- Managed infrastructure, databases, APIs, SDKs, and web applications to make these data sources publicly accessible.
- Helpers to start and manage instances in your own environment.
- SaaS to run containers in a managed environment without direct vendor engagement.

<spare_cores>

>>> from sparecores import intro

```
pgsql=> SELECT
  MIN(updates), AVG(updates), MAX(updates),
  PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY updates)
    AS median
FROM (
  SELECT
    to_char(observed_at, 'YYYY-MM-DD HH24'),
    COUNT(*) AS updates
  FROM server_price_scd
  GROUP BY 1)

 min  |          avg          | median | max
------+-----------------------+--------+------
 4925 | 5124.0425531914893617 | 5119.5 | 5345
(1 row)
```

- **4** (out of 9 planned) vendors
- **334** availability zones
- **2014** server types
- **366,488** benchmark scores
- **270,228** live price records
- **~5k** records updated hourly
- **~20M** historical records

Tracked every 5 mins.

Source: sparecores.com

<spare_cores>

>>> from sparecores import intro

Search Prompt

Basics

Processor

Processor number

8 vCPUs

Processor architecture

- arm64
- arm64_mac
- i386
- x86_64
- x86_64_mac

GPU

Memory

Memory amount

64 GB

Storage

Vendor

Columns

NAME & PROVIDER	PROCESSOR	MEMORY	STORAGE	GPUs	GPU MIN MEMORY
g3.4xlarge aws	8x x86_64 8 cores at 2.7 Ghz	122.0 GB	-	1 M60	8.0 GB
g4ad.4xlarge aws	8x x86_64 8 cores at 3 Ghz	64.0 GB	600 GB nvme ssd	1 Radeon Pro V520	8.0 GB
g4dn.4xlarge aws	8x x86_64 8 cores at 2.5 Ghz	64.0 GB	225 GB nvme ssd	1 T4	16.0 GB
g5.4xlarge aws	8x x86_64 8 cores at 3.3 Ghz	64.0 GB	600 GB nvme ssd	1 A10G	24.0 GB
g6.4xlarge aws	8x x86_64 8 cores at 3.4 Ghz	64.0 GB	600 GB nvme ssd	1 L4	22.4 GB
gr6.4xlarge aws	8x x86_64 8 cores at 3.4 Ghz	128.0 GB	600 GB nvme ssd	1 L4	22.4 GB
g3.8xlarge aws	16x x86_64 16 cores at 2.7 Ghz	244.0 GB	-	2 M60	8.0 GB
g4ad.8xlarge aws	16x x86_64 16 cores at 3 Ghz	128.0 GB	1.2 TB nvme ssd	2 Radeon Pro V520	8.0 GB
g4dn.8xlarge aws	16x x86_64 16 cores at 2.5 Ghz	128.0 GB	900 GB nvme ssd	1 T4	16.0 GB
g5.8xlarge aws	16x x86_64 16 cores at 3.3 Ghz	128.0 GB	900 GB nvme ssd	1 A10G	24.0 GB
g5g.8xlarge aws	32x arm64 32 cores at 2.5 Ghz	64.0 GB	-	1 T4g	16.0 GB

<spare_cores>

>>> from sparecores import intro

c6g.4xlarge by Amazon Web Services

c6g.4xlarge is a Compute optimized [AWS Graviton processors] Gen6 4xlarge server offered by Amazon Web Services with 16 vCPUs, 32GB of memory and 0GB of storage. The pricing starts at $0.0627 per hour.

Visit Vendor | Status Page

16 vCPU

32GB Memory

Spare SCore

4684
(All-cores)

293
(Single-core)

Specifications

Server Details

General	
Vendor ID	aws
Server ID	c6g.4xlarge
Name	c6g.4xlarge
API Reference	c6g.4xlarge
Display Name	c6g.4xlarge
Description	Compute optimized [AWS Graviton processors] Gen6 4xlarge
Family	

View more details

Availability

REGION	SPOT	ONDEMAND
Hyderabad (IN)	$0.094567/hour	$0.3408/hour
Mumbai (IN)	$0.132533/hour	$0.3408/hour
Oregon (US)	$0.199025/hour	$0.544/hour
Ohio (US)	$0.194533/hour	$0.544/hour
Northern Virgina (US)	$0.25192/hour	$0.544/hour
Aragón (ES)	$0.216267/hour	$0.5837/hour
Dublin (IE)	$0	$0.5837/hour

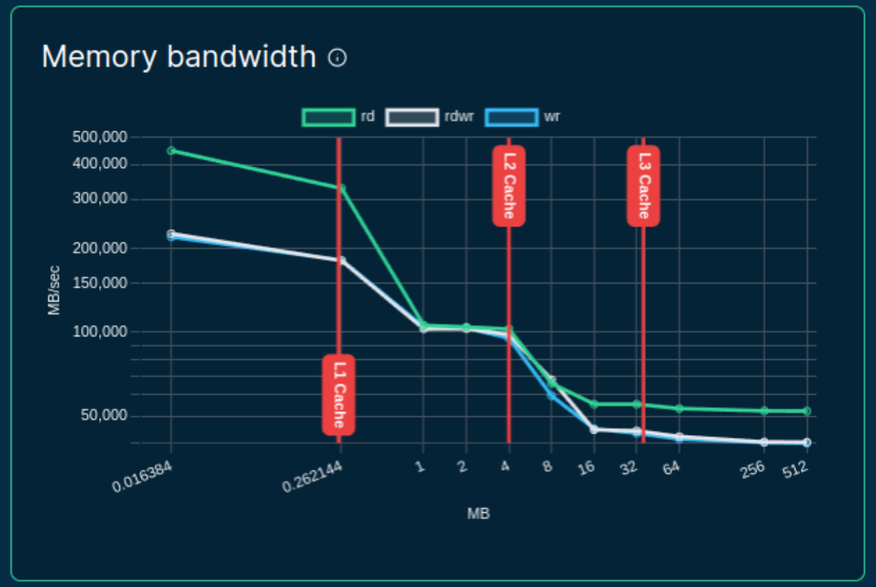
Show more

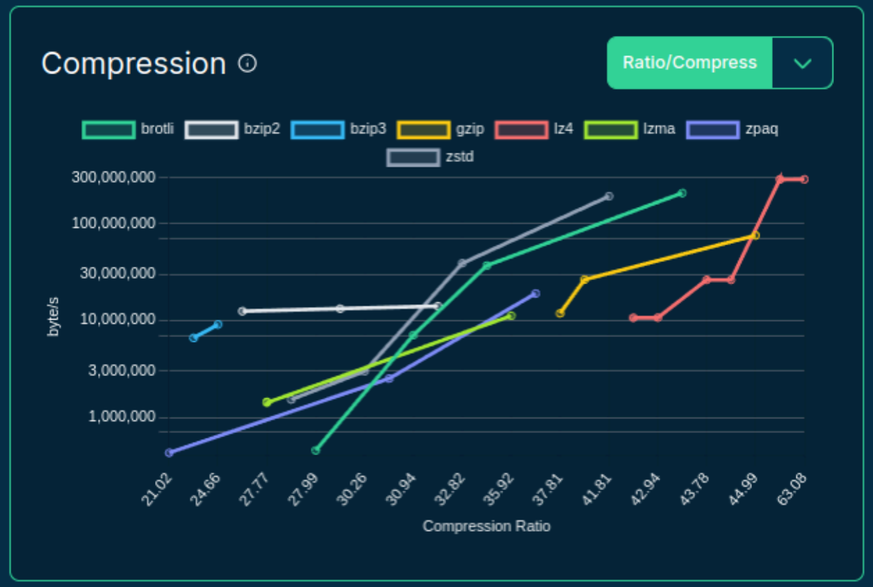
<spare_cores>

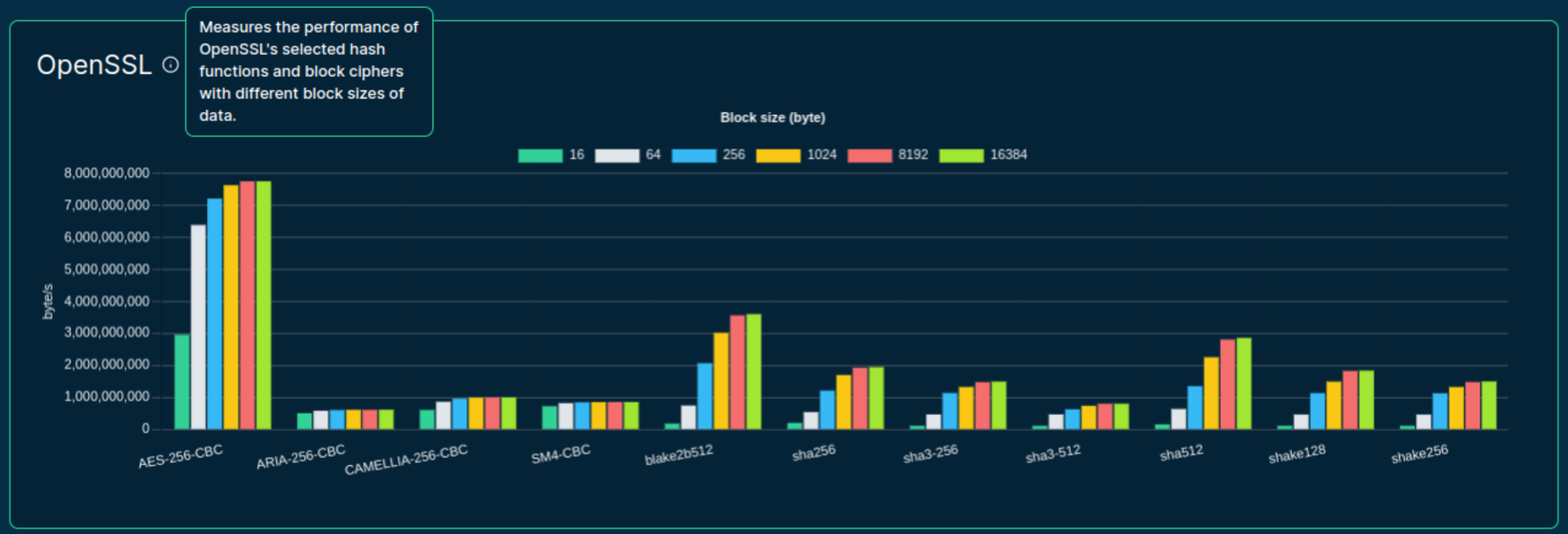
>>> from sparecores import intro

<spare_cores>

>>> from sparecores import intro

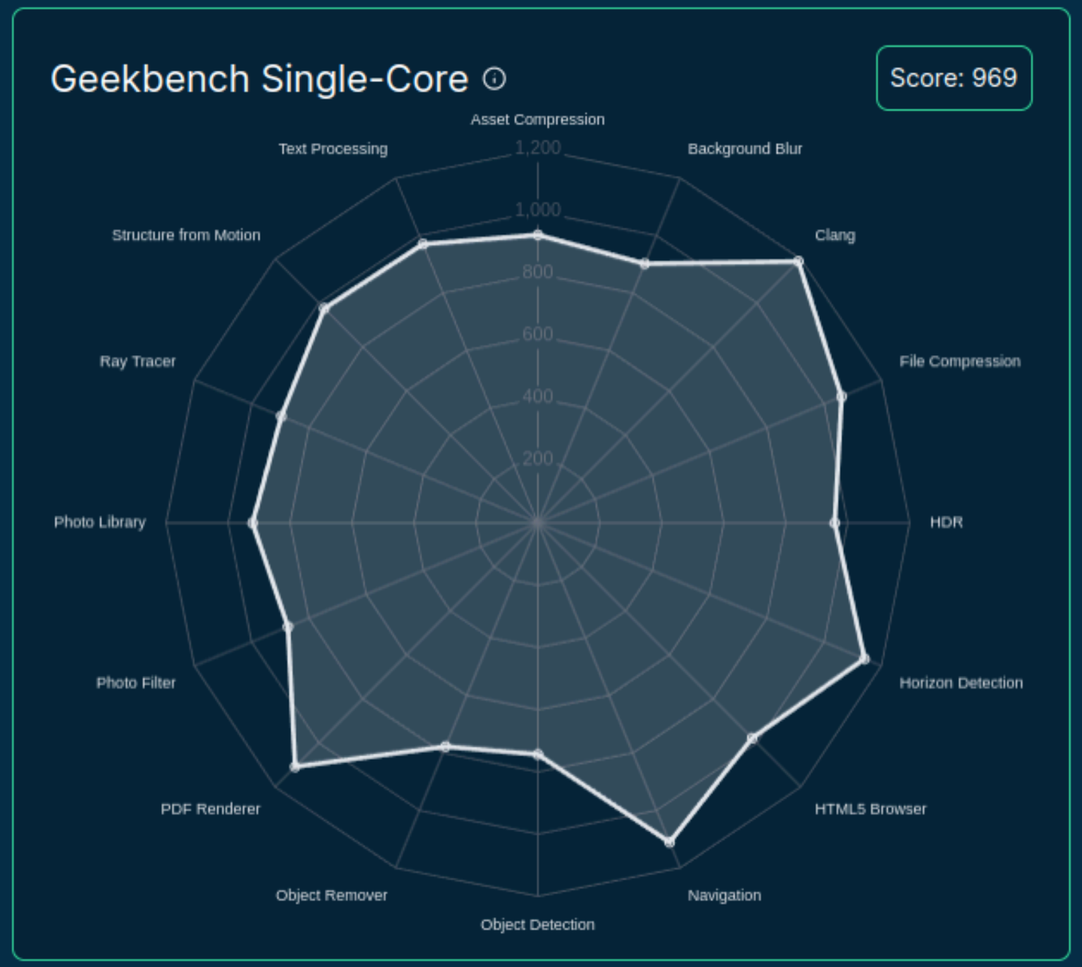


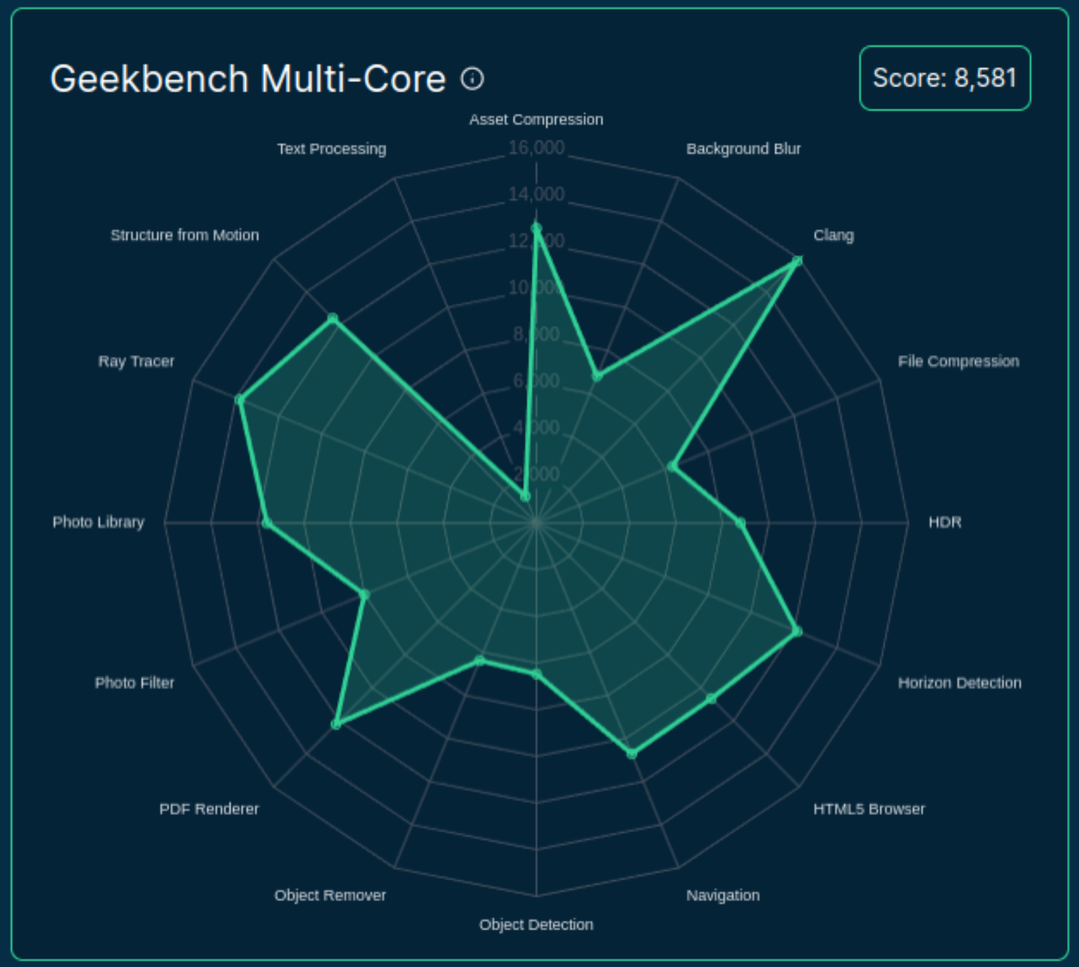




<spare_cores>

>>> from sparecores import intro





<spare_cores>

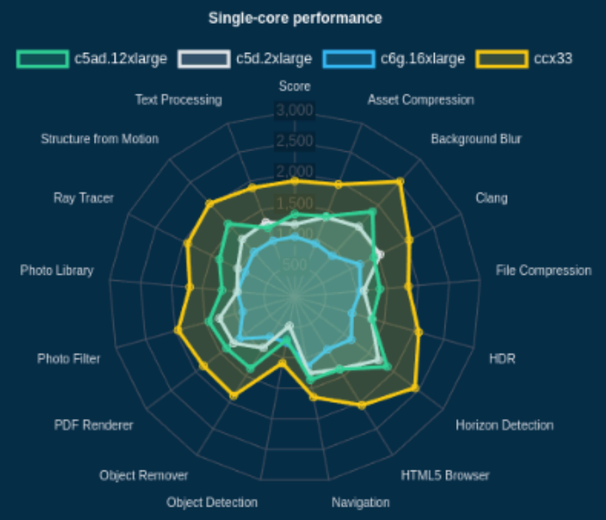
>>> from sparecores import intro

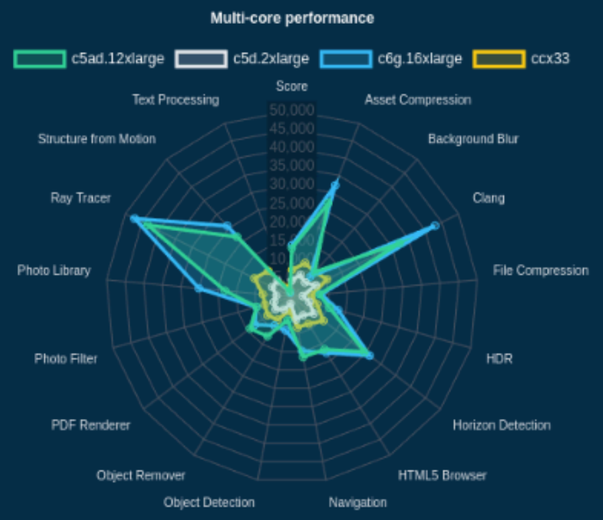
	C5AD.12XLARGE	C5D.2XLARGE	C6G.16XLARGE	CCX33
Vendor	aws	aws	aws	
SCore All	52208	9373	18709	10958
SCore Single	1387	1608	293	1757
Best ondemand price	2.064$	0.384$	1.3632$	0.0769$
vCPUs	48	8	64	8
Hypervisor	nitro	nitro	nitro	
CPU Allocation	Dedicated	Dedicated	Dedicated	Dedicated
CPU Cores	24	4	64	
CPU Speed	3.3 GHz	3.9 GHz	2.5 GHz	
CPU Architecture	x86_64	x86_64	arm64	x86_64
CPU Manufacturer	AMD	Intel	AWS	AMD
CPU Family	Zen	Xeon	ARMv8	
CPU Model	AMD EPYC 7R32	8275CL	AWS Graviton2	
CPU L1 Cache	2 MiB	256 KiB	8 MiB	256 KiB
CPU L2 Cache	12 MiB	4 MiB	64 MiB	2 MiB
CPU L3 Cache	96 MiB	36 MiB	32 MiB	32 MiB
Memory Amount	96 GB	16 GB	128 GB	32 GB
Memory Generation	DDR4	DDR4		
Memory Speed	2933 Mhz	2933 Mhz		

<spare_cores>

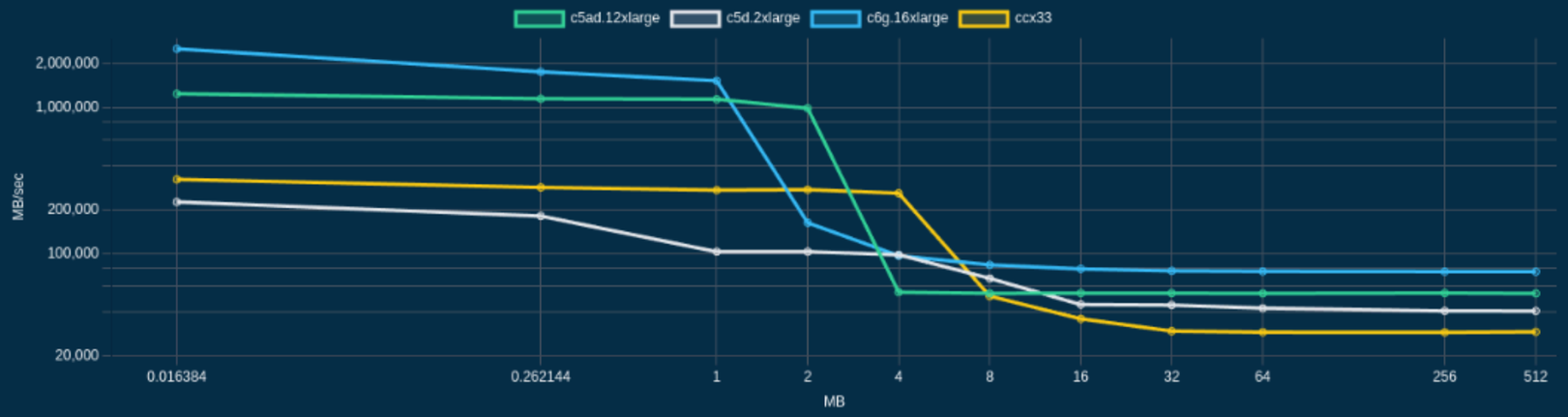
>>> from sparecores import intro

Geekbench





Memory bandwidth



>>> from sparecores import intro

Search...

- Licensing
- References
- Administrative endpoints
- Table dumps
- Table metadata
- Query Resources
  - GET Search Regions
  - GET Get Server
  - GET Search Servers
  - GET Search Server Prices
- AI

API docs by Redocly

Get Server

Query a single server by its vendor id and either the server or, or its API reference.

Return dictionary includes all server fields, along with the current prices per zone, and the available benchmark scores.

PATH PARAMETERS

vendor required	string (Vendor) Vendor ID.
server required	string (Server) Server ID or API reference.

Responses

200 Successful Response

RESPONSE SCHEMA: application/json

vendor_id required	string (Vendor Id) Reference to the Vendor.
server_id required	string (Server Id) Unique identifier, as called at the Vendor.
name required	string (Name) Human-friendly name.
api_reference required	string (Api Reference) How this resource is referenced in the vendor API calls. This is usually either the id or name of the resource, depening on the vendor and actual API endpoint.
display_name required	string (Display Name) Human-friendly reference (usually the id or name) of the resource.
description > required	Description (string) or Description (null) (Description) Short description.

GET /server/{vendor}/{server}

Response samples

200 422

Content type
application/json

Copy Expand all Collapse all


```
{
  "api_reference": "a1.2xlarge",
- "benchmark_scores": [
   + { … }
  ],
  "cpu_allocation": "Dedicated",
  "cpu_architecture": "arm64",
  "cpu_cores": 8,
  "cpu_family": "ARMv8",
- "cpu_flags": [
     "fp",
     "asimd",
     "evtstrm",
     "aes",
     "pmull",
     "sha1",
     "sha2",
     "crc32",
     "cpuid"
  ],
  "cpu_l1_cache": 655360,
  "cpu_l2_cache": 4194304,
  "cpu_manufacturer": "AWS",
  "cpu_model": "AWS Graviton",
  "cpu_speed": 2.5,
  "cpus": [ ],
  "description": "AWS Graviton Gen1 2xlarge",
  "display_name": "a1.2xlarge",
  "family": "a1"
```

>>> from sparecores import intro

```
>>> from rich import print as pp
>>> from sc_crawler.tables import Server
>>> from sqlmodel import create_engine, Session, select
>>> engine = create_engine("sqlite:///sc-data-all.db")
>>> session = Session(engine)
>>> server = session.exec(select(Server).where(Server.server_id == 'g4dn.xlarge')).one()
>>> pp(server)
Server(
    server_id='g4dn.xlarge',
    vendor_id='aws',
    display_name='g4dn.xlarge',
    api_reference='g4dn.xlarge',
    name='g4dn.xlarge',
    family='g4dn',
    description='Graphics intensive [Instance store volumes] [Network and EBS optimized] Gen4 xlarge',

    status=<Status.ACTIVE: 'active'>,
    observed_at=datetime.datetime(2024, 6, 6, 10, 18, 4, 127254),

    hypervisor='nitro',
```

>>> sparecores.__dir__()

COMPONENT	STATUS	REPOSITORY	DESCRIPTION
SC Crawler	Beta		Inventory cloud resources into a SQlite database.
SC Inspector	Beta		Inspect and benchmark cloud resources.
SC Data	Beta		Wrapper around data collected using the Crawler.
SC Keeper	Alpha		API to search the Data.
SC Scanner	Stable		Web frontend and programming language SDKs for Keeper.
SC Runner	Beta		Launching actual cloud instances.

<spare_cores>

>>> import sc_crawler

- ETL framework with database schema and inventory method definitions

>>> import sc_crawler

- ETL framework with database schema and inventory method definitions
- Database migration tool supporting multiple database engines

>>> import sc_crawler

- ETL framework with database schema and inventory method definitions
- Database migration tool supporting multiple database engines
- Manual list of vendors and metadata

>>> import sc_crawler

- ETL framework with database schema and inventory method definitions
- Database migration tool supporting multiple database engines
- Manual list of vendors and metadata
- Vendor API integrations to list regions, zones, servers, storages, prices, included free traffic and IPv4 addresses etc.

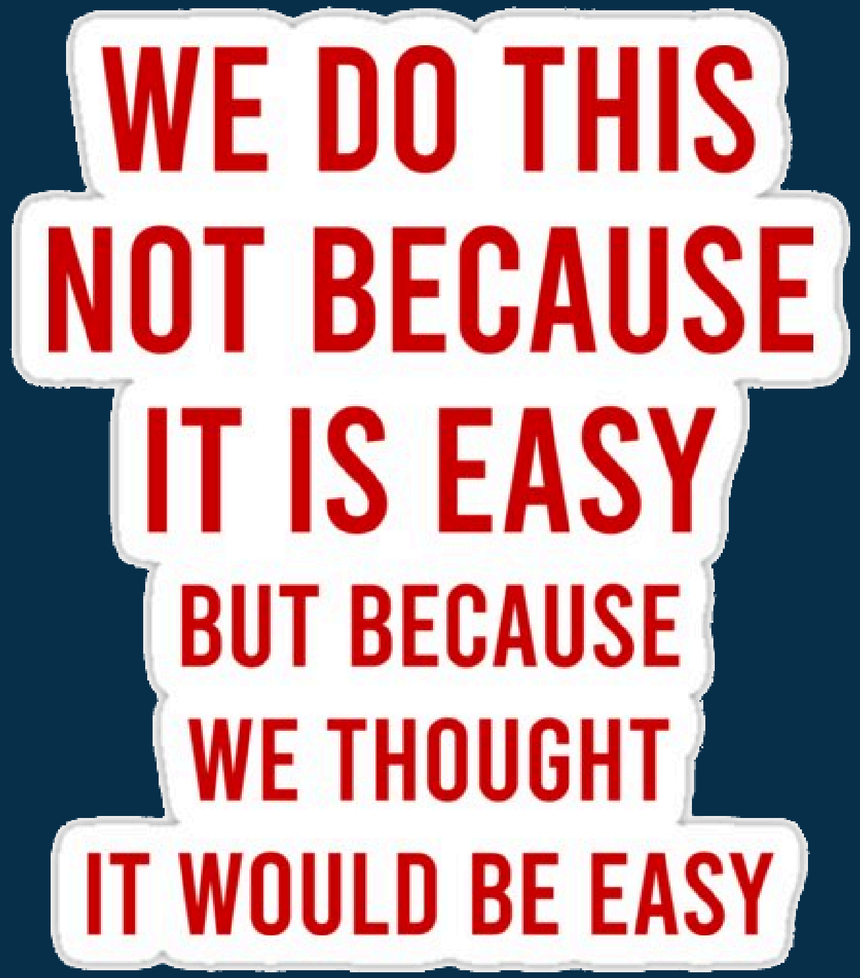
>>> import sc_crawler

- ETL framework with database schema and inventory method definitions
- Database migration tool supporting multiple database engines
- Manual list of vendors and metadata
- Vendor API integrations to list regions, zones, servers, storages, prices, included free traffic and IPv4 addresses etc.
- Spare Cores Inspector integration for hardware discovery and benchmark scores

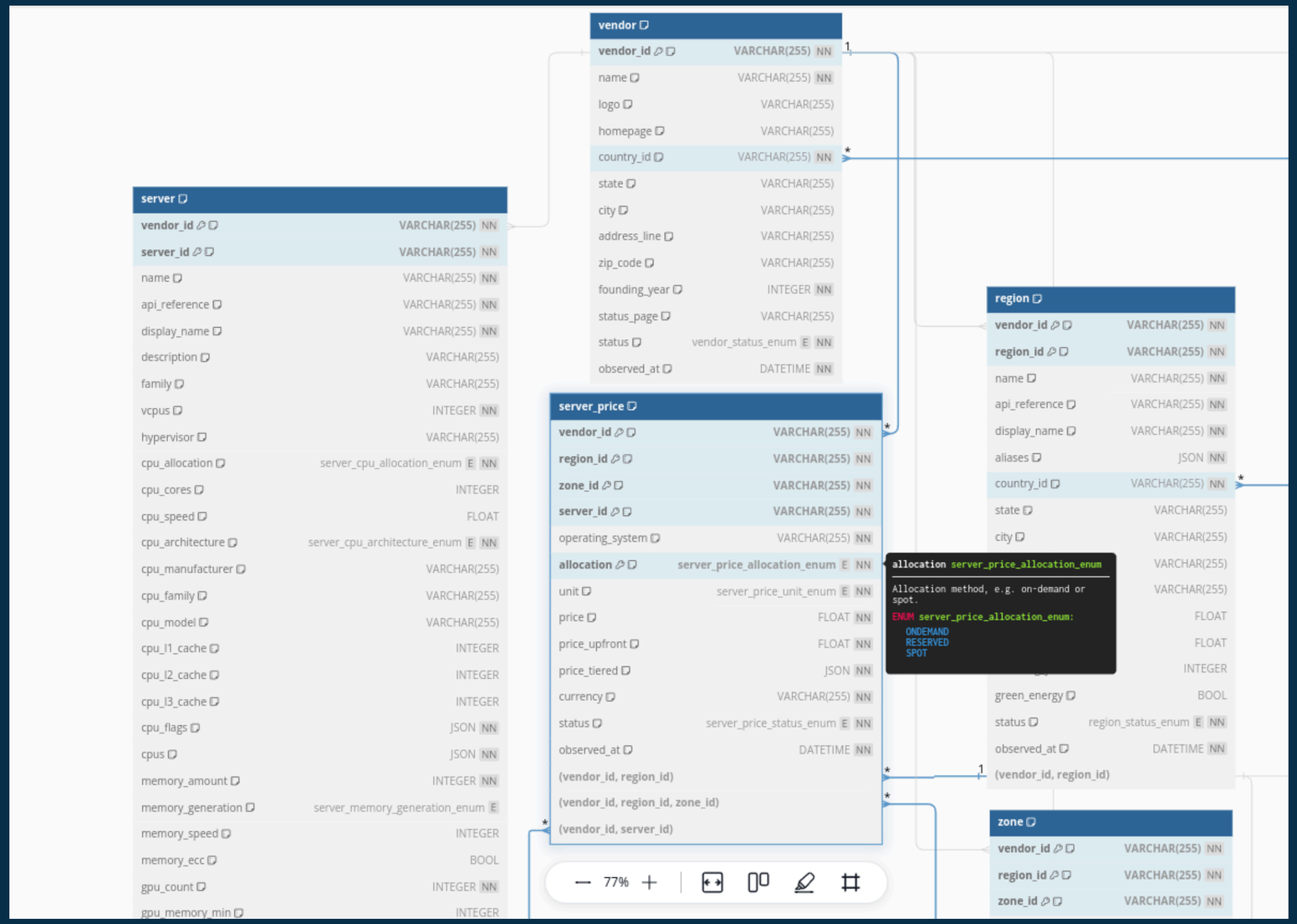
>>> import sc_crawler

- ETL framework with database schema and inventory method definitions
- Database migration tool supporting multiple database engines
- Manual list of vendors and metadata
- Vendor API integrations to list regions, zones, servers, storages, prices, included free traffic and IPv4 addresses etc.
- Spare Cores Inspector integration for hardware discovery and benchmark scores
- Dependency for other Spare Cores components (schemas)

>>> import sc_crawler



>>> from sc_crawler import fks



Source: dbdocs.io/spare-cores

>>> from sc_crawler import scd

Need to optionally track price etc. changes.

>>> from sc_crawler import scd

Need to optionally track price etc. changes.

```
class Scd(ScModel):
    """Override the `observed_at` column to be primary key in SCD tables."""

    observed_at: datetime = Field(
        primary_key=True,
        default_factory=datetime.utcnow,
        sa_column_kwargs={"onupdate": datetime.utcnow},
        description="Timestamp of the last observation.",
    )
```

>>> from sc_crawler import alembic

Let's update the `cpu_cores` column to be optional, as some vendors as shy sharing that over their APIs. We will backfill with the Spare Cores Inspector!

>>> from sc_crawler import alembic

Let's update the `cpu_cores` column to be optional, as some vendors as shy sharing that over their APIs. We will backfill with the Spare Cores Inspector!

```
"""v0.1.1 cores optional

Revision ID: 4691089690c2
Revises: 98894dffd37c
Create Date: 2024-04-10 00:59:03.509522

"""

from typing import Sequence, Union

import sqlalchemy as sa
import sqlmodel
from alembic import op
```

>>> from sc_crawler import alembic

```
$ sc-crawler schemas upgrade --sql
```

>>> from sc_crawler import alembic

```
$ sc-crawler schemas upgrade --sql
```

```sql
CREATE TABLE _alembic_tmp_server (
    vendor_id VARCHAR NOT NULL,
    server_id VARCHAR NOT NULL,
    name VARCHAR NOT NULL,
    vcpus INTEGER NOT NULL,
    hypervisor VARCHAR,
    cpu_allocation VARCHAR(9) NOT NULL,
    cpu_cores INTEGER,
    cpu_speed FLOAT,
    cpu_architecture VARCHAR(10) NOT NULL,
    cpu_manufacturer VARCHAR,
    cpu_family VARCHAR,
    cpu_model VARCHAR,
    cpus JSON NOT NULL,
    memory INTEGER NOT NULL,
    gpu_count INTEGER NOT NULL,
    gpu_memory_min INTEGER,
```

>>> from sc_crawler import hwinfo

- Varying quality and availability of data at different vendors.

>>> from sc_crawler import hwinfo

- Varying quality and availability of data at different vendors.

how much storage does the d2.4xlarge AWS instance type provide?

Searched 6 sites

The AWS d2.4xlarge instance type provides 48 TB of HDD storage through 12 x 2 TB HDDs. This configuration is designed for storage-intensive applications such as massively parallel processing (MPP) data warehousing, Hadoop distributed computing, and log or data-processing applications (Vantage) (Azure, AWS and GCP Specs and Pricing) (CloudySave).

>>> from sc_crawler import hwinfo

- Varying quality and availability of data at different vendors.

how much storage does the d2.4xlarge AWS instance type provide?

Searched 6 sites

The AWS d2.4xlarge instance type provides 48 TB of HDD storage through 12 x 2 TB HDDs. This configuration is designed for storage-intensive applications such as massively parallel processing (MPP) data warehousing, Hadoop distributed computing, and log or data-processing applications (Vantage) (Azure, AWS and GCP Specs and Pricing) (CloudySave).

>>> from sc_crawler import hwinfo

- Varying quality and availability of data at different vendors.

>>> from sc_crawler import hwinfo

- Varying quality and availability of data at different vendors.
  - No SSD info via the API. Parse from server description! 🙀

>>> from sc_crawler import hwinfo

- Varying quality and availability of data at different vendors.
  - No SSD info via the API. Parse from server description! 🙀
  - No CPU info via the API. Extracting from homepage! 😿

>>> from sc_crawler import hwinfo

- Varying quality and availability of data at different vendors.
  - No SSD info via the API. Parse from server description! 🙀
  - No CPU info via the API. Extracting from homepage! 😿
  - No hypervisor info via the API. Manual mappings! 🙀😿

>>> from sc_crawler import hwinfo

- Varying quality and availability of data at different vendors.
  - No SSD info via the API. Parse from server description! 😿
  - No CPU info via the API. Extracting from homepage! 😿
  - No hypervisor info via the API. Manual mappings! 😿😿
- Region, right?

>>> from sc_crawler import hwinfo

- Varying quality and availability of data at different vendors.
  - No SSD info via the API. Parse from server description! 🙀
  - No CPU info via the API. Extracting from homepage! 😿
  - No hypervisor info via the API. Manual mappings! 🙀😿
- Region, right?
  - ID, eg `eu-west-1`

>>> from sc_crawler import hwinfo

- Varying quality and availability of data at different vendors.
  - No SSD info via the API. Parse from server description! 🙀
  - No CPU info via the API. Extracting from homepage! 😿
  - No hypervisor info via the API. Manual mappings! 🙀😿
- Region, right?
  - ID, eg `eu-west-1`
  - Name, eg `Europe (Ireland)`

>>> from sc_crawler import hwinfo

- Varying quality and availability of data at different vendors.
  - No SSD info via the API. Parse from server description! 🙀
  - No CPU info via the API. Extracting from homepage! 😿
  - No hypervisor info via the API. Manual mappings! 🙀😿
- Region, right?
  - ID, eg `eu-west-1`
  - Name, eg `Europe (Ireland)`
  - Alias, eg `EU (Ireland)`

<spare_cores>

>>> from sc_crawler import hwinfo

- Varying quality and availability of data at different vendors.
  - No SSD info via the API. Parse from server description! 🙀
  - No CPU info via the API. Extracting from homepage! 😿
  - No hypervisor info via the API. Manual mappings! 🙀😿
- Region, right?
  - ID, eg `eu-west-1`
  - Name, eg `Europe (Ireland)`
  - Alias, eg `EU (Ireland)`
  - API reference, eg `eu-west-1`

<spare_cores>

>>> from sc_crawler import hwinfo

- Varying quality and availability of data at different vendors.
  - No SSD info via the API. Parse from server description! 🙀
  - No CPU info via the API. Extracting from homepage! 😿
  - No hypervisor info via the API. Manual mappings! 🙀😿
- Region, right?
  - ID, eg `eu-west-1`
  - Name, eg `Europe (Ireland)`
  - Alias, eg `EU (Ireland)`
  - API reference, eg `eu-west-1`
  - Display name, eg `Dublin (IE)`

>>> from sc_crawler import hwinfo

- Varying quality and availability of data at different vendors.
  - No SSD info via the API. Parse from server description! 🙀
  - No CPU info via the API. Extracting from homepage! 😿
  - No hypervisor info via the API. Manual mappings! 🙀😿
- Region, right?
  - ID, eg `eu-west-1`
  - Name, eg `Europe (Ireland)`
  - Alias, eg `EU (Ireland)`
  - API reference, eg `eu-west-1`
  - Display name, eg `Dublin (IE)`
  - Exact location? Energy source?

>>> from sc_crawler import pricing

- No way to find SKUs by filtering in the API call. Get all, search locally.

>>> from sc_crawler import pricing

- No way to find SKUs by filtering in the API call. Get all, search locally.
- `f1-micro` is one out of 2 instances with simple pricing.
  - For other instances, lookup SKUs for CPU + RAM and do the math.

>>> from sc_crawler import pricing

- No way to find SKUs by filtering in the API call. Get all, search locally.
- `f1-micro` is one out of 2 instances with simple pricing.
  - For other instances, lookup SKUs for CPU + RAM and do the math.
- Match instance family with SKU via search in description, e.g. `C2D`.

>>> from sc_crawler import pricing

- No way to find SKUs by filtering in the API call. Get all, search locally.
- `f1-micro` is one out of 2 instances with simple pricing.
  - For other instances, lookup SKUs for CPU + RAM and do the math.
- Match instance family with SKU via search in description, e.g. `C2D`.
- Except for `c2`, which is called “Compute optimized”.

>>> from sc_crawler import pricing

- No way to find SKUs by filtering in the API call. Get all, search locally.
- `f1-micro` is one out of 2 instances with simple pricing.
  - For other instances, lookup SKUs for CPU + RAM and do the math.
- Match instance family with SKU via search in description, e.g. `C2D`.
- Except for `c2`, which is called “Compute optimized”.
- And `m2` is actually priced at a premium on the top of `m1`.

<spare_cores>

>>> from sc_crawler import pricing

- No way to find SKUs by filtering in the API call. Get all, search locally.
- `f1-micro` is one out of 2 instances with simple pricing.
  - For other instances, lookup SKUs for CPU + RAM and do the math.
- Match instance family with SKU via search in description, e.g. `C2D`.
- Except for `c2`, which is called “Compute optimized”.
- And `m2` is actually priced at a premium on the top of `m1`.
- The `n1` resource group is not CPU/RAM, but `N1Standard`, extract if it’s CPU or RAM price from description.

>>> import sc_crawler

DBHub.io

Login / Register

sparecores / sc-data-priceless.db

Watch 1 | Star 1 | Fork 0

Data | Visualise | Discussions: 0 | Merge Requests: 0

Visibility: Public **Last Commit:** a5dd449a (7 minutes ago) **Licence:** CC-BY-SA-4.0 **Size:** 1,092 KB

Structured data on cloud compute resource collected by sc-crawler.
Source: https://github.com/SpareCores/sc-data/actions/runs/9437577578

Commits: 110 | **Branches: 1** | **Tags: 0** | **Releases: 0** | **Contributors: 1**

Table/view: server | **Branch:** main | Clone database in DB4S | Download database

- benchmark
- benchmark_score
- compliance_framew
- country
- ipv4_price
- region
- server
- server_price

1-25 of 1,120 total rows

		name	api_reference	display_name	description	family	vcpus	hypervisor	cpu_allocation	cpu_cores	cpu_speed	cpu_architecture	cpu_manufacturer	cpu_family
		m7a.metal-48xl	m7a.metal-4...	m7a.metal-4...	General purpose [AMD processors] Gen7 metal-48xl	m7a	192		DEDICATED	96	3.7	X86_64	AMD	Zen
		c7a.metal-48xl	c7a.metal-48xl	c7a.metal-48xl	Compute optimized [AMD processors] Gen7 metal-48xl	c7a	192		DEDICATED	96	3.7	X86_64	AMD	Zen
		c7a.48xlarge	c7a.48xlarge	c7a.48xlarge	Compute optimized [AMD processors] Gen7 48xlarge	c7a	192	nitro	DEDICATED	96	3.7	X86_64	AMD	Zen
		m7a.32xlarge	m7a.32xlarge	m7a.32xlarge	General purpose [AMD processors] Gen7 32xlarge	m7a	128	nitro	DEDICATED	64	3.7	X86_64	AMD	Zen
		c7a.32xlarge	c7a.32xlarge	c7a.32xlarge	Compute optimized [AMD processors] Gen7 32xlarge	c7a	128	nitro	DEDICATED	64	3.7	X86_64	AMD	Zen
		m7a.24xlarge	m7a.24xlarge	m7a.24xlarge	General purpose [AMD processors] Gen7 24xlarge	m7a	96	nitro	DEDICATED	96	3.7	X86_64	AMD	Zen
aws	m6a.m...	m6a.metal	m6a.metal	m6a.metal	General purpose [AMD processors] Gen6 metal	m6a	192		DEDICATED	48	3.725	X86_64	AMD	Zen
aws	m6a.48...	m6a.48xlarge	m6a.48xlarge	m6a.48xlarge	General purpose [AMD processors] Gen6 48xlarge	m6a	192	nitro	DEDICATED	48	3.725	X86_64	AMD	Zen
aws	c7a.24x...	c7a.24xlarge	c7a.24xlarge	c7a.24xlarge	Compute optimized [AMD processors] Gen7 24xlarge	c7a	96	nitro	DEDICATED	96	3.7	X86_64	AMD	Zen
aws	c6a.me...	c6a.metal	c6a.metal	c6a.metal	Compute optimized [AMD processors] Gen6 metal	c6a	192		DEDICATED	48	3.725	X86_64	AMD	Zen
aws	c6a.48x...	c6a.48xlarge	c6a.48xlarge	c6a.48xlarge	Compute optimized [AMD processors] Gen6 48xlarge	c6a	192	nitro	DEDICATED	48	3.725	X86_64	AMD	Zen
aws	r6a.32x...	r6a.32xlarge	r6a.32xlarge	r6a.32xlarge	Memory optimized [AMD processors] Gen6 32xlarge	r6a	128	nitro	DEDICATED	64	3.6	X86_64	AMD	
aws	m7a.16...	m7a.16xlarge	m7a.16xlarge	m7a.16xlarge	General purpose [AMD processors] Gen7 16xlarge	m7a	64	nitro	DEDICATED	64	3.7	X86_64	AMD	Zen
aws	m6a.32...	m6a.32xlarge	m6a.32xlarge	m6a.32xlarge	General purpose [AMD processors] Gen6 32xlarge	m6a	128	nitro	DEDICATED	32	3.725	X86_64	AMD	Zen
aws	c7a.16x...	c7a.16xlarge	c7a.16xlarge	c7a.16xlarge	Compute optimized [AMD processors] Gen7 16xlarge	c7a	64	nitro	DEDICATED	64	3.7	X86_64	AMD	Zen

https://dbhub.io/branches/sparecores/sc-data-priceless.db

Source: dbhub.io/sparecores

<spare_cores>

>>> import sc_data

>>> import sc_data

- GitHub Action set up to run the Crawler every 5 minutes.

>>> import sc_data

- GitHub Action set up to run the Crawler every 5 minutes.
  - ~30,000 GHA runs

>>> import sc_data

- GitHub Action set up to run the Crawler every 5 minutes.
  - ~30,000 GHA runs
  - ~900 releases (with non-price changes)

>>> import sc_data

- GitHub Action set up to run the Crawler every 5 minutes.
  - ~30,000 GHA runs
  - ~900 releases (with non-price changes)
- Make the data available in a public (CC BY-SA) SQLite database:

>>> import sc_data

- GitHub Action set up to run the Crawler every 5 minutes.
  - ~30,000 GHA runs
  - ~900 releases (with non-price changes)
- Make the data available in a public (CC BY-SA) SQLite database:
  - 350 MiB SQLite

>>> import sc_data

- GitHub Action set up to run the Crawler every 5 minutes.
  - ~30,000 GHA runs
  - ~900 releases (with non-price changes)
- Make the data available in a public (CC BY-SA) SQLite database:
  - 350 MiB SQLite
  - 2,000+ active servers and their ~275k prices tracked

>>> import sc_data

- GitHub Action set up to run the Crawler every 5 minutes.
  - ~30,000 GHA runs
  - ~900 releases (with non-price changes)
- Make the data available in a public (CC BY-SA) SQLite database:
  - 350 MiB SQLite
  - 2,000+ active servers and their ~275k prices tracked
  - 800k+ measured scores across 24 benchmarks

>>> import sc_data

- GitHub Action set up to run the Crawler every 5 minutes.
  - ~30,000 GHA runs
  - ~900 releases (with non-price changes)
- Make the data available in a public (CC BY-SA) SQLite database:
  - 350 MiB SQLite
  - 2,000+ active servers and their ~275k prices tracked
  - 800k+ measured scores across 24 benchmarks
- Thin Python package to keep the data updated from S3.

<spare_cores>

>>> import sc_data

- GitHub Action set up to run the Crawler every 5 minutes.
  - ~30,000 GHA runs
  - ~900 releases (with non-price changes)
- Make the data available in a public (CC BY-SA) SQLite database:
  - 350 MiB SQLite
  - 2,000+ active servers and their ~275k prices tracked
  - 800k+ measured scores across 24 benchmarks
- Thin Python package to keep the data updated from S3.
  - Package version is tied to Crawler version.

>>> import sc_inspector

Information collected from vendor APIs is very limited, so we run:

>>> import sc_inspector

Information collected from vendor APIs is very limited, so we run:

- Hardware inspection tools:

>>> import sc_inspector

Information collected from vendor APIs is very limited, so we run:

- Hardware inspection tools:
  - `dmidecode`

>>> import sc_inspector

Information collected from vendor APIs is very limited, so we run:

- Hardware inspection tools:
  - `dmidecode`
  - `lscpu`

>>> import sc_inspector

Information collected from vendor APIs is very limited, so we run:

- Hardware inspection tools:
  - `dmidecode`
  - `lscpu`
  - `lshw`

>>> import sc_inspector

Information collected from vendor APIs is very limited, so we run:

- Hardware inspection tools:
  - `dmidecode`
  - `lscpu`
  - `lshw`
  - `nvidia-smi`

>>> import sc_inspector

Information collected from vendor APIs is very limited, so we run:

- Hardware inspection tools:
  - `dmidecode`
  - `lscpu`
  - `lshw`
  - `nvidia-smi`
- Benchmarking workloads:

>>> import sc_inspector

Information collected from vendor APIs is very limited, so we run:

- Hardware inspection tools:
  - `dmidecode`
  - `lscpu`
  - `lshw`
  - `nvidia-smi`
- Benchmarking workloads:
  - `bw_mem`

>>> import sc_inspector

Information collected from vendor APIs is very limited, so we run:

- Hardware inspection tools:
  - `dmidecode`
  - `lscpu`
  - `lshw`
  - `nvidia-smi`
- Benchmarking workloads:
  - `bw_mem`
  - Compression algos

<spare_cores>

>>> import sc_inspector

Information collected from vendor APIs is very limited, so we run:

- Hardware inspection tools:
  - `dmidecode`
  - `lscpu`
  - `lshw`
  - `nvidia-smi`
- Benchmarking workloads:
  - `bw_mem`
  - Compression algos
  - OpenSSL hash functions and block ciphers

<spare_cores>

>>> import sc_inspector

Information collected from vendor APIs is very limited, so we run:

- Hardware inspection tools:
  - `dmidecode`
  - `lscpu`
  - `lshw`
  - `nvidia-smi`
- Benchmarking workloads:
  - `bw_mem`
  - Compression algos
  - OpenSSL hash functions and block ciphers
  - Geekbench 6

<spare_cores>

>>> import sc_inspector

Information collected from vendor APIs is very limited, so we run:

- Hardware inspection tools:
  - `dmidecode`
  - `lscpu`
  - `lshw`
  - `nvidia-smi`
- Benchmarking workloads:
  - `bw_mem`
  - Compression algos
  - OpenSSL hash functions and block ciphers
  - Geekbench 6
  - `stress-ng`

>>> import sc_inspector

Data is collected in public: `sc-inspector-data` repo on GitHub.

SpareCores / **sc-inspector-data**

Type / to search

Code | Issues | Pull requests | **Actions** | Security | Insights | Settings

Actions New workflow

All workflows

Clean up resources created by the sta...

Delete old workflow runs

Parse outputs

Start instances for collecting data

Management

Caches

Attestations

Runners

All workflows

Showing runs from all workflows

Filter workflow runs

3,543 workflow runs Event Status Branch Actor

Clean up resources created by the start job Clean up resources created by the start job #3056: Scheduled	main	5 minutes ago 3m 32s
Start instances for collecting data Start instances for collecting data #446: Scheduled	main	8 minutes ago 2m 11s
Clean up resources created by the start job Clean up resources created by the start job #3055: Scheduled	main	14 minutes ago 4m 46s
Start instances for collecting data Start instances for collecting data #445: Scheduled	main	25 minutes ago 2m 45s
Clean up resources created by the start job Clean up resources created by the start job #3054: Scheduled	main	26 minutes ago 3m 28s
Clean up resources created by the start job Clean up resources created by the start job #3053: Scheduled	main	47 minutes ago 3m 29s
Start instances for collecting data Start instances for collecting data #444: Scheduled	main	53 minutes ago 2m 11s

<spare_cores>

>>> import sc_inspector

SpareCores / **sc-inspector-data**

Type / to search

Code | Issues | Pull requests | Actions | Security | Insights | Settings

Files

main

Go to file t

- openssl
  - meta.json
  - parsed.json
  - stdout
- stressng
- stressngsinglecore
- x1.16xlarge
- x1.32xlarge
- x1e.16xlarge
- x1e.2xlarge
- x1e.32xlarge
- x1e.4xlarge
- x1e.8xlarge
- x1e.xlarge

sc-inspector-data / data / aws / **t4g.xlarge** /

Add file

Spare Cores Inspecting server from https://github.com/SpareCores/sc-inspector-dat... ce948b8 · yesterday History

Name	Last commit message	Last commit date
..		
bw_mem	Inspecting server from https://github.com/SpareCores/sc-ins...	yesterday
compression_text	Inspecting server from https://github.com/SpareCores/sc-ins...	yesterday
dmidecode	Parsed outputs in https://github.com/SpareCores/sc-inspect...	yesterday
geekbench	Inspecting server from https://github.com/SpareCores/sc-ins...	yesterday
lscpu	Inspecting server from https://github.com/SpareCores/sc-ins...	yesterday
lshw	Inspecting server from https://github.com/SpareCores/sc-ins...	yesterday
openssl	Parsed outputs in https://github.com/SpareCores/sc-inspect...	yesterday
stressng	Inspecting server from https://github.com/SpareCores/sc-ins...	yesterday
stressngsinglecore	Inspecting server from https://github.com/SpareCores/sc-ins...	yesterday

<spare_cores>

>>> import sc_inspector

```
docker run --rm -ti -v /var/run/docker.sock:/var/run/docker.sock \
  -e GITHUB_TOKEN=${GITHUB_TOKEN} \
  -e BENCHMARK_SECRETS_PASSPHRASE=${BENCHMARK_SECRETS_PASSPHRASE} \
  ghcr.io/sparecores/sc-inspector:main \
  inspect --vendor ${VENDOR} --instance ${INSTANCE} --gpu-count ${GPU_COUNT}
```

>>> import sc_inspector

```
docker run --rm -ti -v /var/run/docker.sock:/var/run/docker.sock \
    -e GITHUB_TOKEN=${GITHUB_TOKEN} \
    -e BENCHMARK_SECRETS_PASSPHRASE=${BENCHMARK_SECRETS_PASSPHRASE} \
    ghcr.io/sparecores/sc-inspector:main \
    inspect --vendor ${VENDOR} --instance ${INSTANCE} --gpu-count ${GPU_COUNT}
```

6 packages

sc-inspector Published on Apr 5 by Spare Cores in **SpareCores/sc-inspector**	8.44k
hwinfo Published 3 weeks ago by Spare Cores in **SpareCores/sc-images**	2.62k
benchmark Published 3 weeks ago by Spare Cores in **SpareCores/sc-images**	1.52k
dmidecode Published last month by Spare Cores in **SpareCores/sc-images**	1.37k
sc-runner Published on May 1 by Spare Cores in **SpareCores/sc-runner**	284
sc-keeper Published on Apr 12 by Spare Cores in **SpareCores/sc-keeper**	70

<spare_cores>

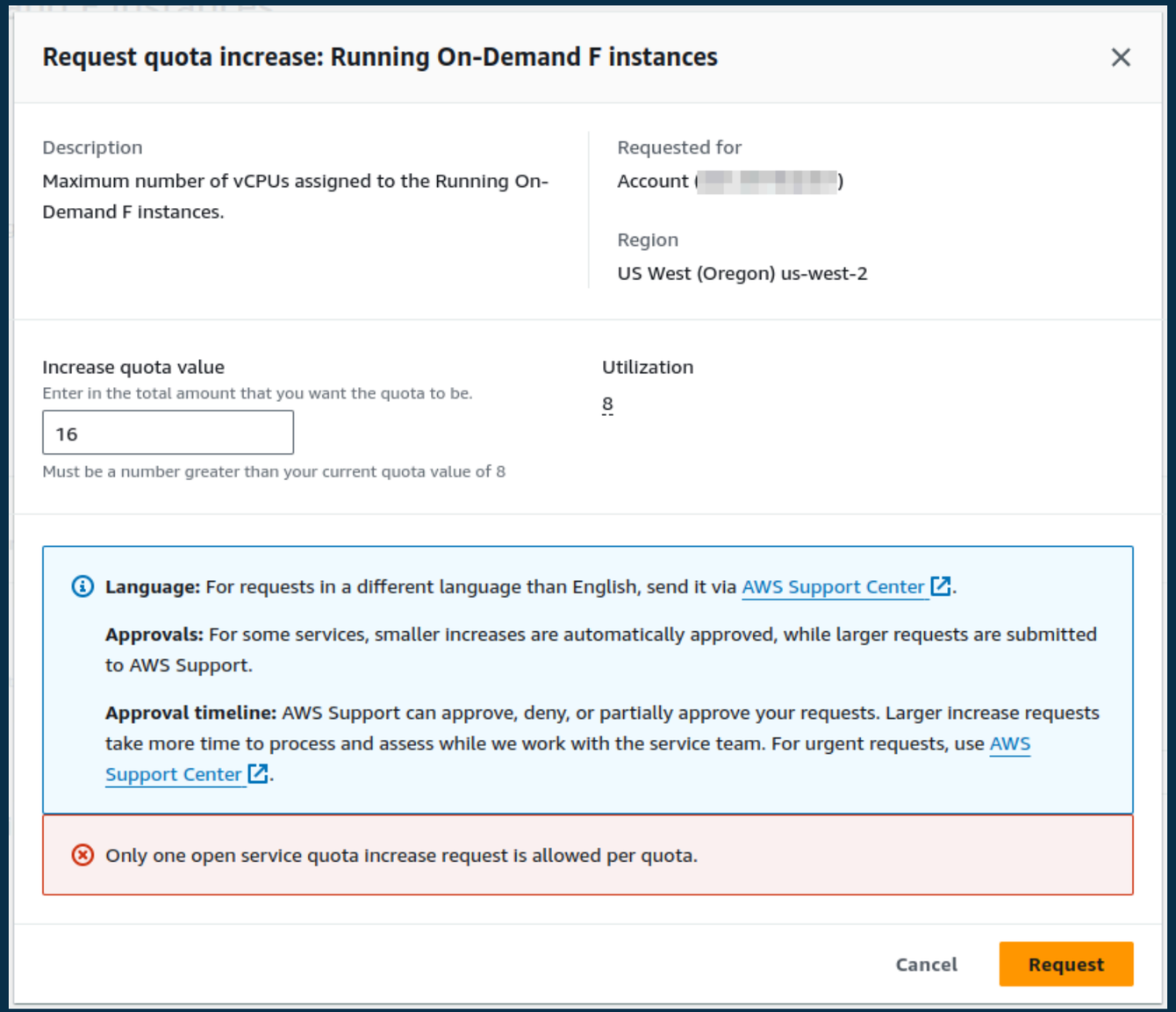
>>> import sc_runner

```
$ docker run --rm -ti \
  ghcr.io/sparecores/sc-runner:main \
    create aws --instance t4g.nano

Updating (aws.us-west-2.None.t4g.nano):

    pulumi:pulumi:Stack runner-aws.us-west-2.None.t4g.nano running
 +  pulumi:providers:aws us-west-2 creating (0s)
@ updating....
 +  pulumi:providers:aws us-west-2 created (0.29s)
 +  aws:ec2:SecurityGroup t4g.nano creating (0s)
@ updating.....
 +  aws:ec2:SecurityGroup t4g.nano created (2s)
@ updating....
 +  aws:vpc:SecurityGroupIngressRule t4g.nano-0 creating (0s)
 +  aws:vpc:SecurityGroupIngressRule t4g.nano-1 creating (0s)
 +  aws:ec2:Instance t4g.nano creating (0s)
 +  aws:vpc:SecurityGroupEgressRule t4g.nano-1 creating (0s)
 +  aws:vpc:SecurityGroupEgressRule t4g.nano-0 creating (0s)
@ updating....
```

>>> import sc_runner



Request quota increase: Running On-Demand F instances

Description

Maximum number of vCPUs assigned to the Running On-Demand F instances.

Requested for

Account ( )

Region

US West (Oregon) us-west-2

Increase quota value

Enter in the total amount that you want the quota to be.

16

Must be a number greater than your current quota value of 8

Utilization

8

Language: For requests in a different language than English, send it via AWS Support Center.

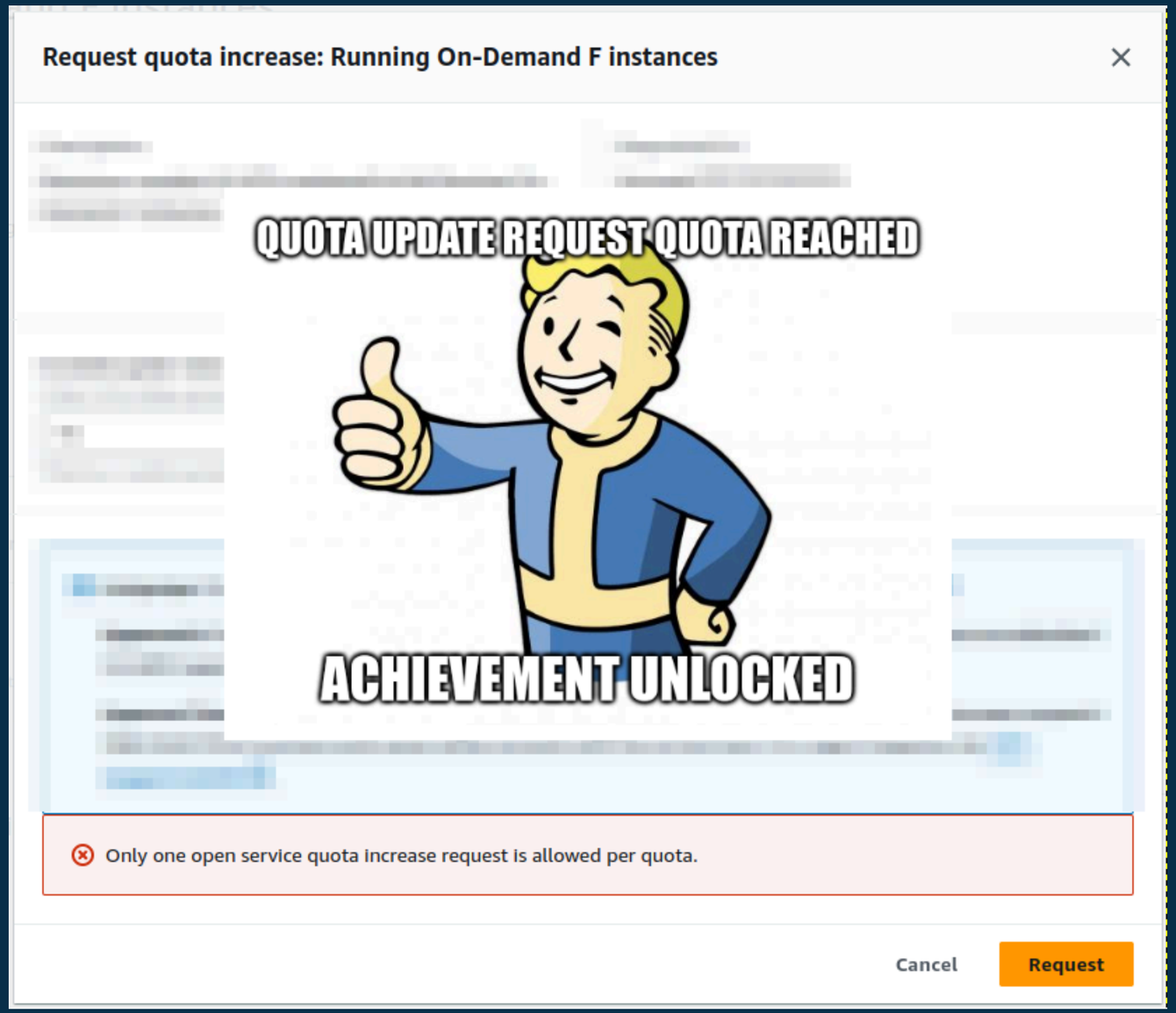
Approvals: For some services, smaller increases are automatically approved, while larger requests are submitted to AWS Support.

Approval timeline: AWS Support can approve, deny, or partially approve your requests. Larger increase requests take more time to process and assess while we work with the service team. For urgent requests, use AWS Support Center.

Only one open service quota increase request is allowed per quota.

Cancel Request

>>> import sc_runner



>>> import sc_keeper

```
$ curl https://keeper.sparecores.net/server/aws/g4dn.xlarge | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  111k  100  111k    0     0  79795      0  0:00:01  0:00:01 --:--:-- 79799
```


```
{
  "vendor_id": "aws",
  "server_id": "g4dn.xlarge",
  "name": "g4dn.xlarge",
  "api_reference": "g4dn.xlarge",
  "display_name": "g4dn.xlarge",
  "description": "Graphics intensive [Instance store volumes] [Network and EBS optimized] Gen4 xlarge",
  "family": "g4dn",
  "vcpus": 4,
  "hypervisor": "nitro",
  "cpu_allocation": "Dedicated",
  "cpu_cores": 2,
  "cpu_speed": 3.5,
  "cpu_architecture": "x86_64",
```

>>> import sc_keeper

Search...

- Licensing
- References
- Administrative endpoints
- Table dumps
- Table metadata
- Query Resources
  - GET Search Regions
  - GET Get Server
  - GET Search Servers
  - GET Search Server Prices
- AI

API docs by Redocly

QUERY PARAMETERS

`vendor` ›
Array of Vendor id (strings) or Vendor id (null) (Vendor id)
Enum: `"aws"` `"gcp"` `"hcloud"`
Identifier of the cloud provider vendor.

Responses

200 Successful Response

RESPONSE SCHEMA: application/json

Array [

- `vendor_id` required — string (Vendor Id)
  Reference to the Vendor.
- `region_id` required — string (Region Id)
  Unique identifier, as called at the Vendor.
- `name` required — string (Name)
  Human-friendly name.
- `api_reference` required — string (Api Reference)
  How this resource is referenced in the vendor API calls. This is usually either the id or name of the resource, depening on the vendor and actual API endpoint.
- `display_name` required — string (Display Name)
  Human-friendly reference (usually the id or name) of the resource.
- `aliases` — Array of strings (Aliases)
  Default: `[]`
  List of other commonly used names for the same Region.
- `country_id` required — string (Country Id)
  Reference to the Country, where the Region is located.
- `state` › — State (string) or State (null) (State)
  Optional state/administrative area of the Region's location within the Country.
- `city` › — City (string) or City (null) (City)

Response samples

200 | 422

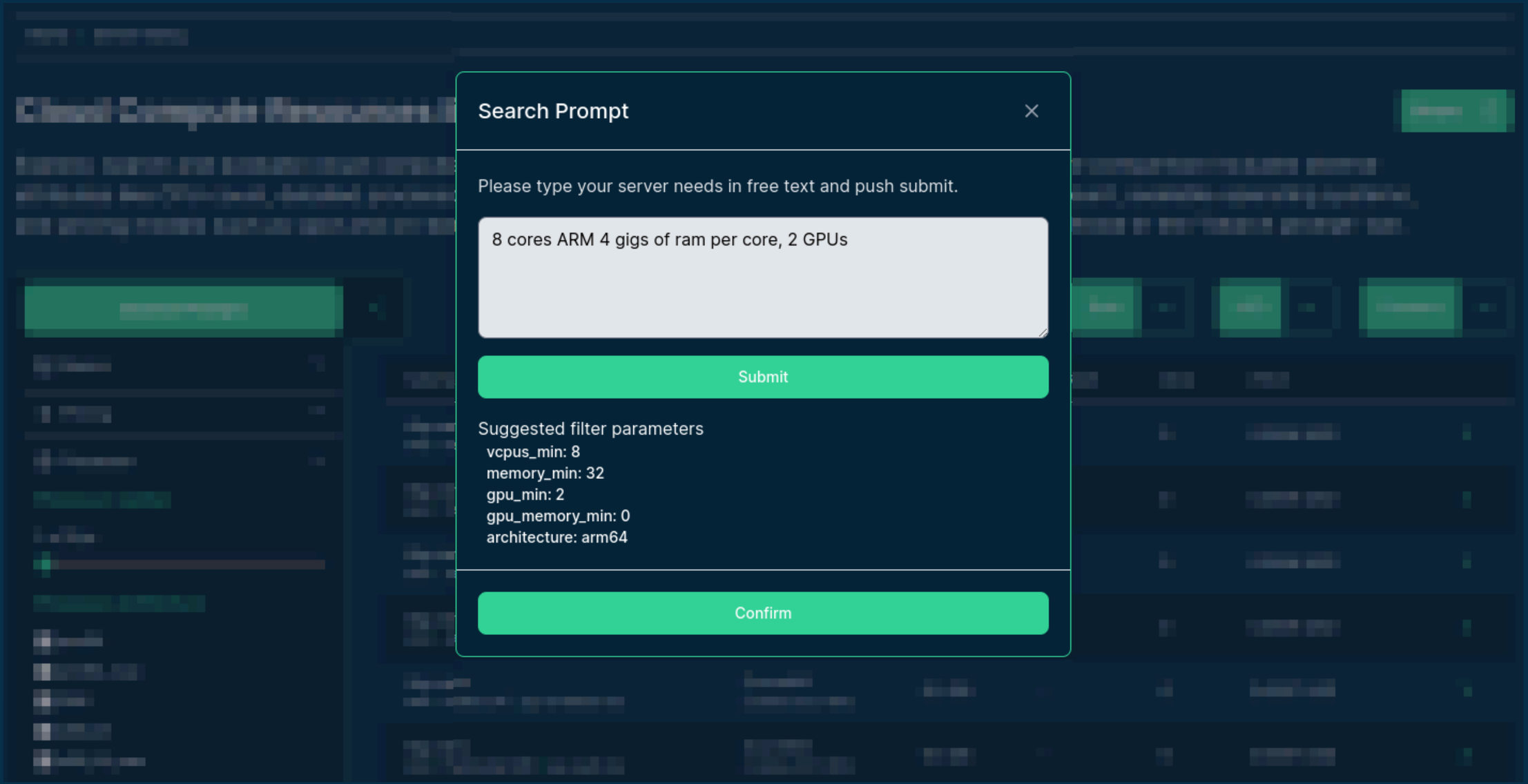
Content type
application/json

Copy Expand all Collapse all


```
[
 - {
      "aliases": [ ],
      "api_reference": "af-south-1",
      "city": "Cape Town",
      "country_id": "ZA",
      "display_name": "Cape Town (ZA)",
      "founding_year": 2020,
      "green_energy": false,
      "lat": -33.914651,
      "lon": 18.3758801,
      "name": "Africa (Cape Town)",
      "observed_at": "2024-06-06T07:18:39.420298",
      "region_id": "af-south-1",
      "status": "active",
    + "vendor": { … },
      "vendor_id": "aws"
   }
]
```

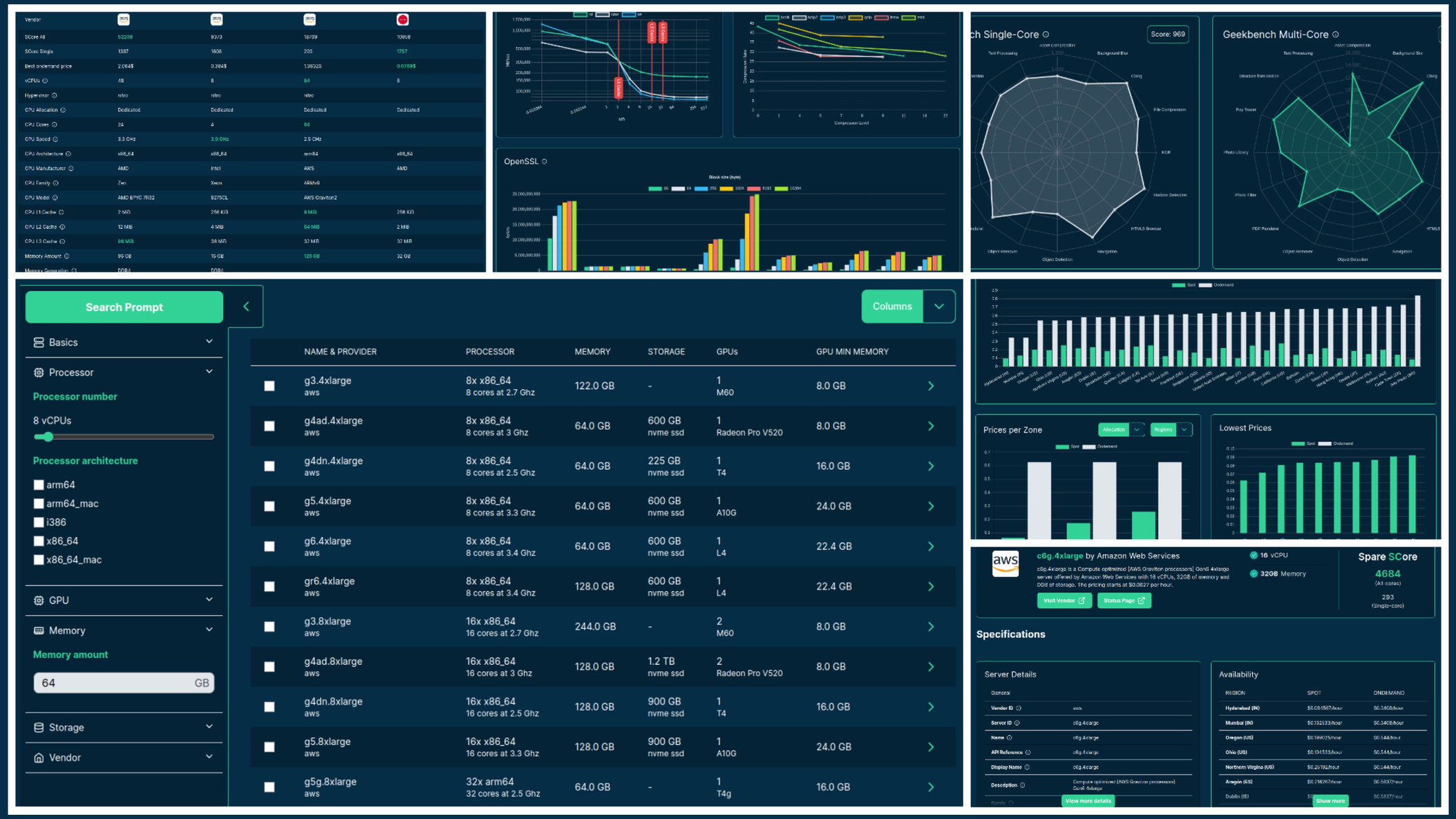

<spare_cores>

>>> import sc_keeper



<spare_cores>

import { AppModule } from ‘sc-www’;



<spare_cores>

import { AppModule } from ‘sc-www’;

Harnessing the compute resources of the cloud to optimize efficiency and costs of batch and service tasks.

- Open Data
- Python
- Benchmarks
- Start a server

Check out servers

Learn more >

Source: sparecores.com

<spare_cores>

>>> import __future__

>>> import __future__

- Add support for more vendors

>>> import __future__

- Add support for more vendors
  - Crawler (vendor API integration)

>>> import __future__

- Add support for more vendors
  - Crawler (vendor API integration)
  - Runner (pulumi)

>>> import __future__

- Add support for more vendors
  - Crawler (vendor API integration)
  - Runner (pulumi)
- More SDKs (PyPI, npm, CRAN, etc.)

>>> import __future__

- Add support for more vendors
  - Crawler (vendor API integration)
  - Runner (pulumi)
- More SDKs (PyPI, npm, CRAN, etc.)
- More benchmarks (e.g. LLM inference speed)

<spare_cores>

>>> import __future__

- Add support for more vendors
  - Crawler (vendor API integration)
  - Runner (pulumi)
- More SDKs (PyPI, npm, CRAN, etc.)
- More benchmarks (e.g. LLM inference speed)
- Data analysis, blog posts

<spare_cores>

>>> import __future__

- Add support for more vendors
  - Crawler (vendor API integration)
  - Runner (pulumi)
- More SDKs (PyPI, npm, CRAN, etc.)
- More benchmarks (e.g. LLM inference speed)
- Data analysis, blog posts
- My Spare Cores (dashboard)

<spare_cores>

>>> import __future__

- Add support for more vendors
  - Crawler (vendor API integration)
  - Runner (pulumi)
- More SDKs (PyPI, npm, CRAN, etc.)
- More benchmarks (e.g. LLM inference speed)
- Data analysis, blog posts
- My Spare Cores (dashboard)
- SaaS 👀

<spare_cores>

>>> from sparecores import team

@bra-fsn

@palabola

@daroczig

>>> from sparecores import team

@bra-fsn

Infrastructure and Python veteran.

@palabola

Guardian of the front-end and Node.js tools.

@daroczig

Hack of all trades, master of `NaN`.

>>> from sparecores import support

```
>>> import os
>>> import signal
>>> os.kill(os.getpid(), signal.SIGKILL)
```

```
>>> import os
>>> import signal
>>> os.kill(os.getpid(), signal.SIGKILL)

>>> visit('https://sparecores.com')
>>> email('daroczig@sparecores.com')
>>> follow('@SpareCores')
```

<spare_cores>

```
>>> import os
>>> import signal
>>> os.kill(os.getpid(), signal.SIGKILL)

>>> visit('https://sparecores.com')
>>> email('daroczig@sparecores.com')
>>> follow('@SpareCores')

>>> os._exit(status=0)
Process finished at 17:55:00 (Nov 8, 2024)
```

<spare_cores>