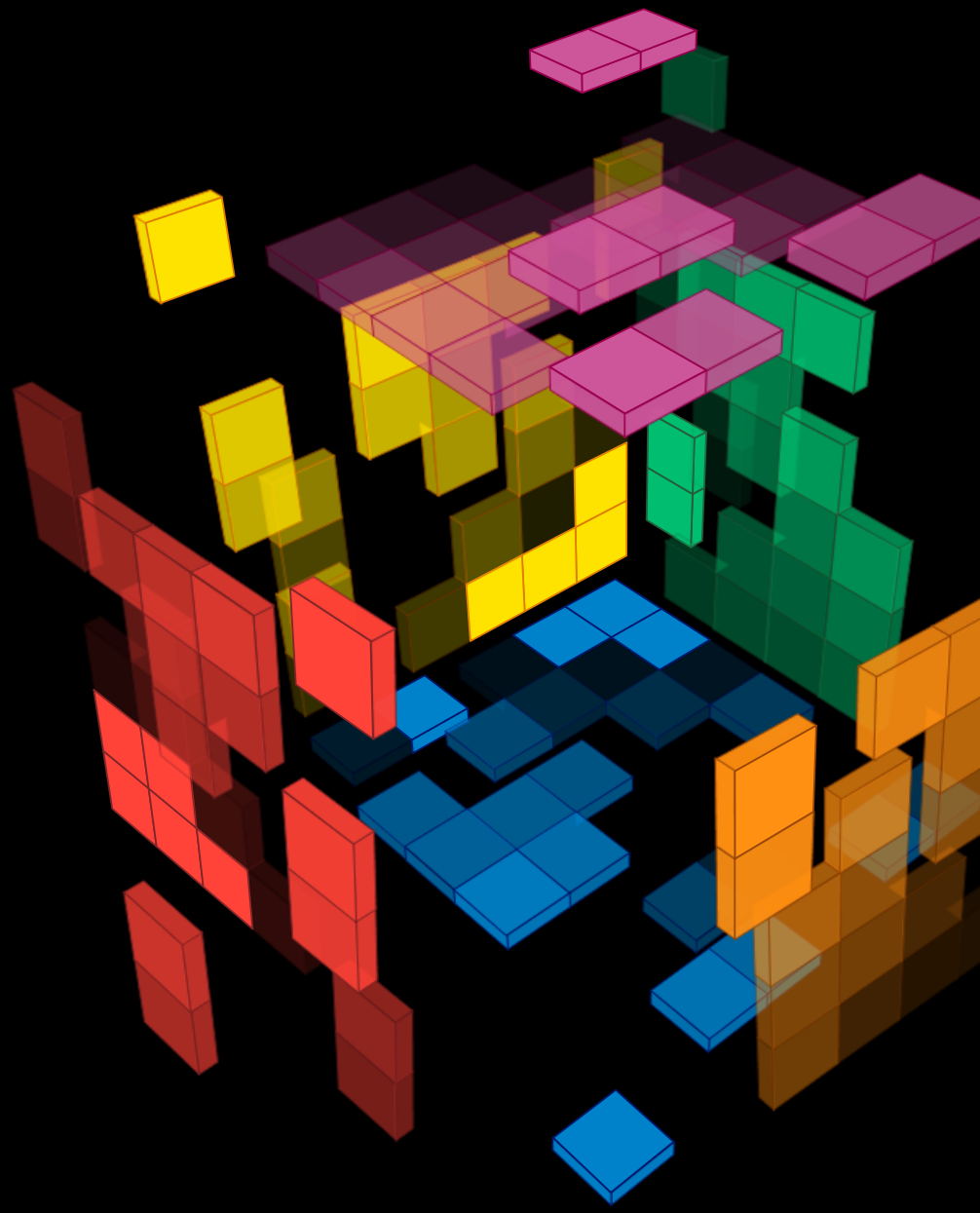
Building Efficiently & Sustainably In The Cloud

Your path to de-carbonise your cloud environment



Shane Baldacchino

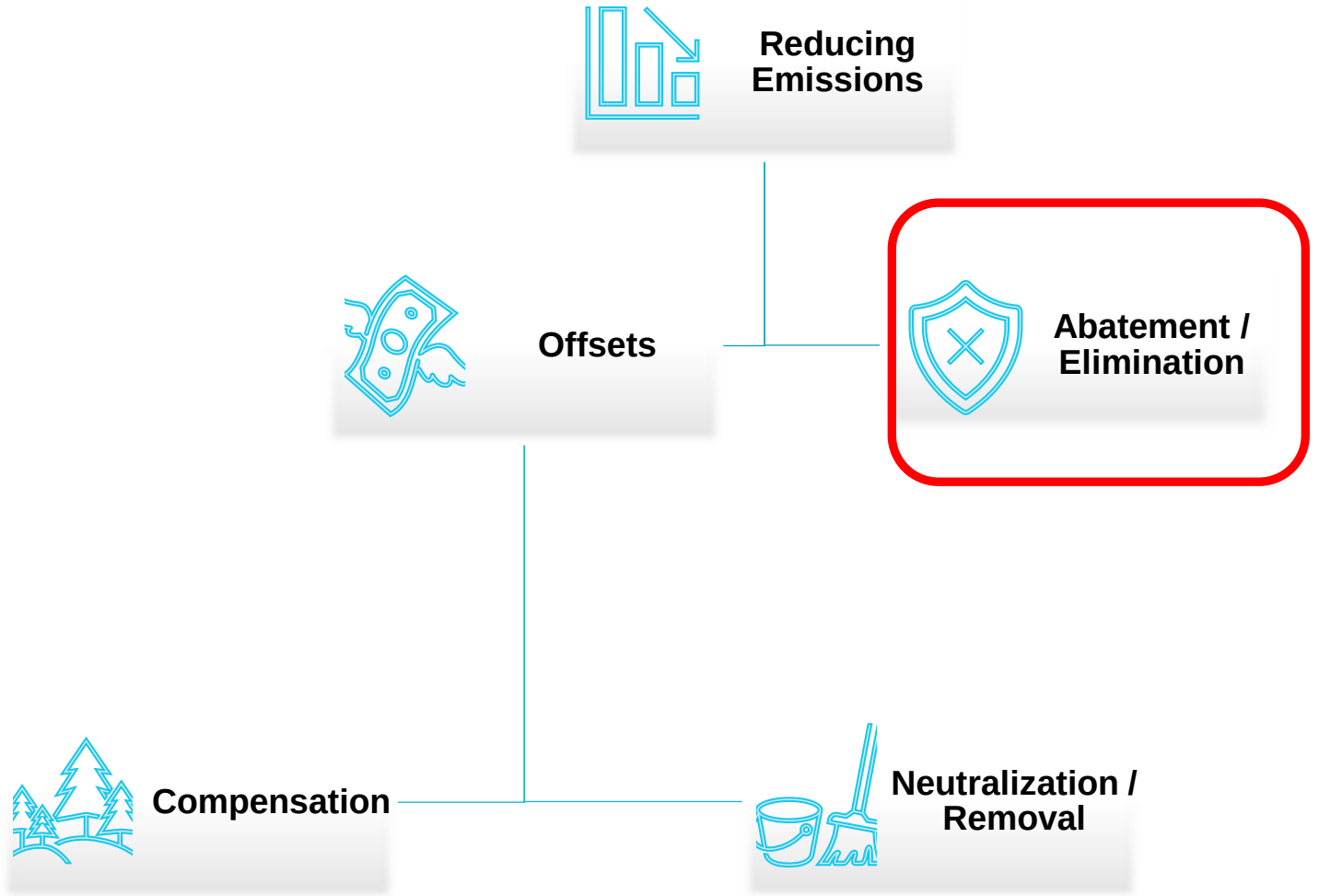
Why
Go Green?


STEAM
GAME PASS
Y
X
B
A

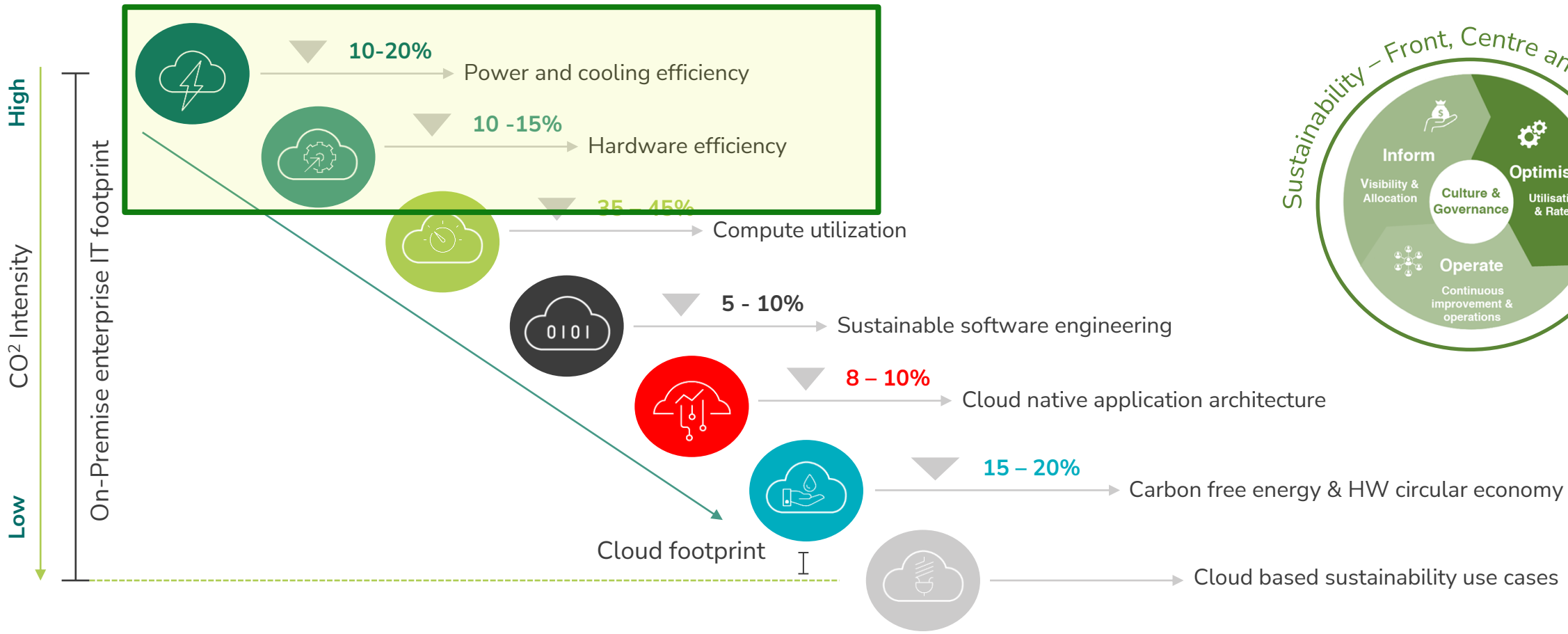
Green
Software Principles

Carbon Reduction

How is the world tackling this problem today?



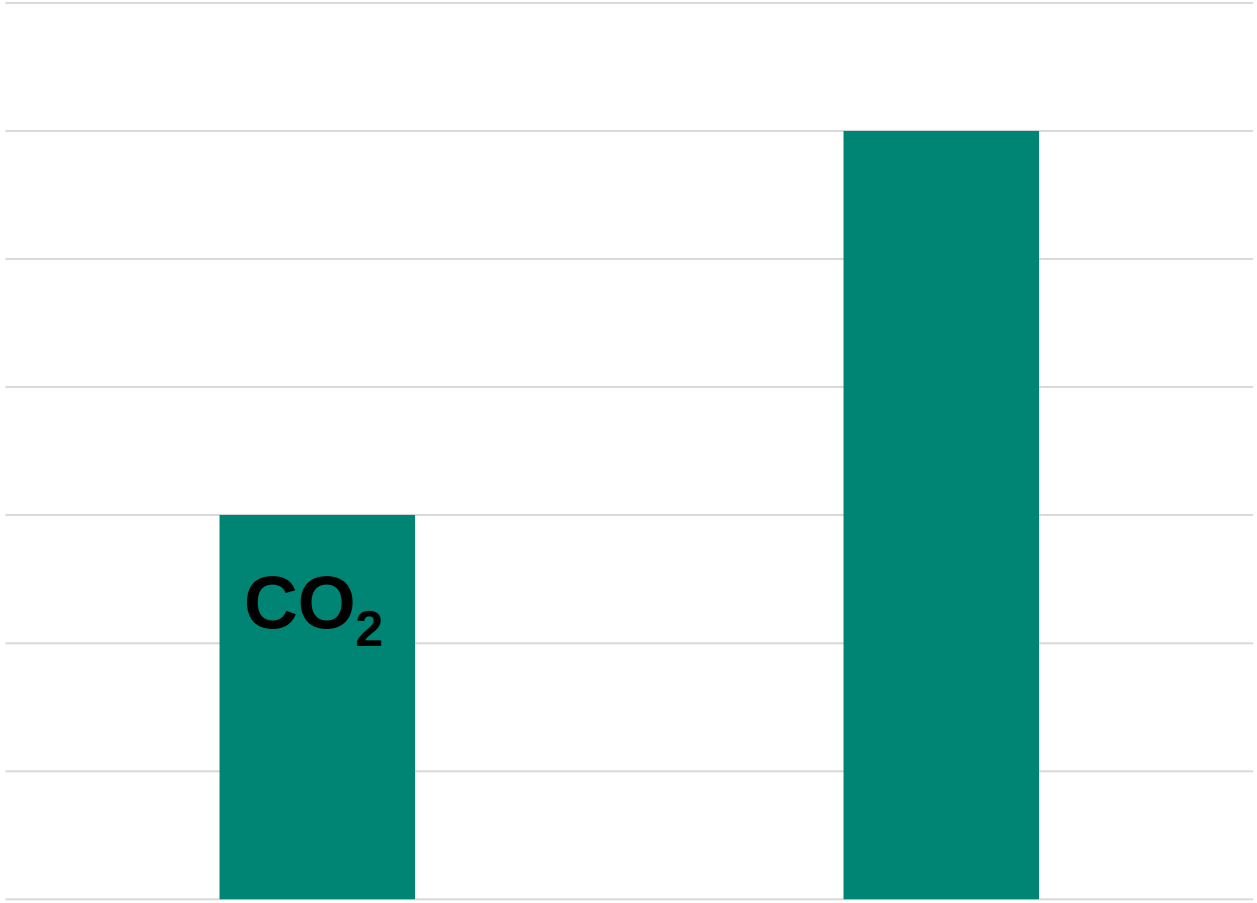
Reductions can be higher



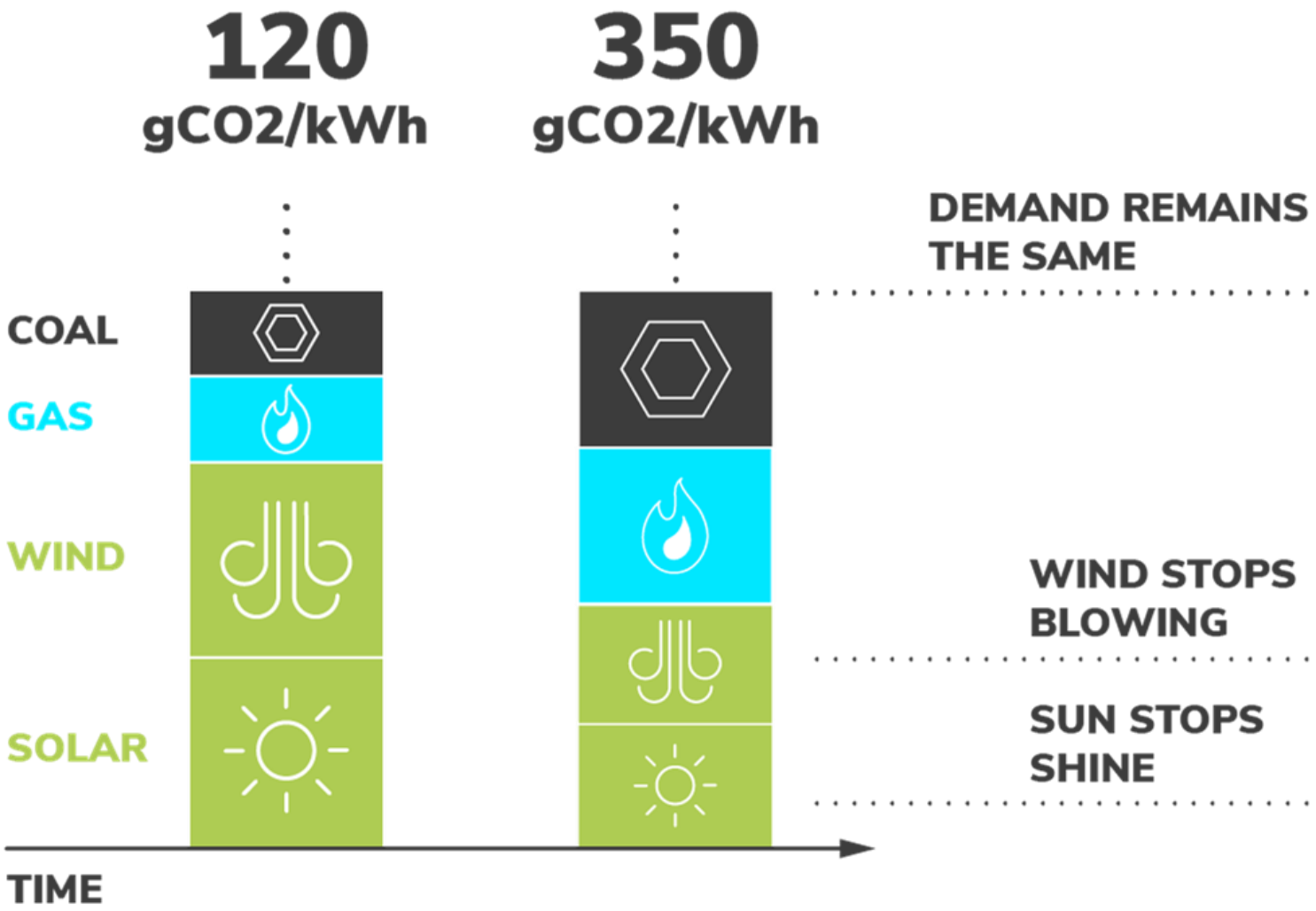
IAAS migrations can reduce carbon emissions by more than 84% compared with conventional infrastructure. Reductions can be pushed even higher – By up to an amazing 98% - by designing applications specifically for the cloud.

Carbon Awareness

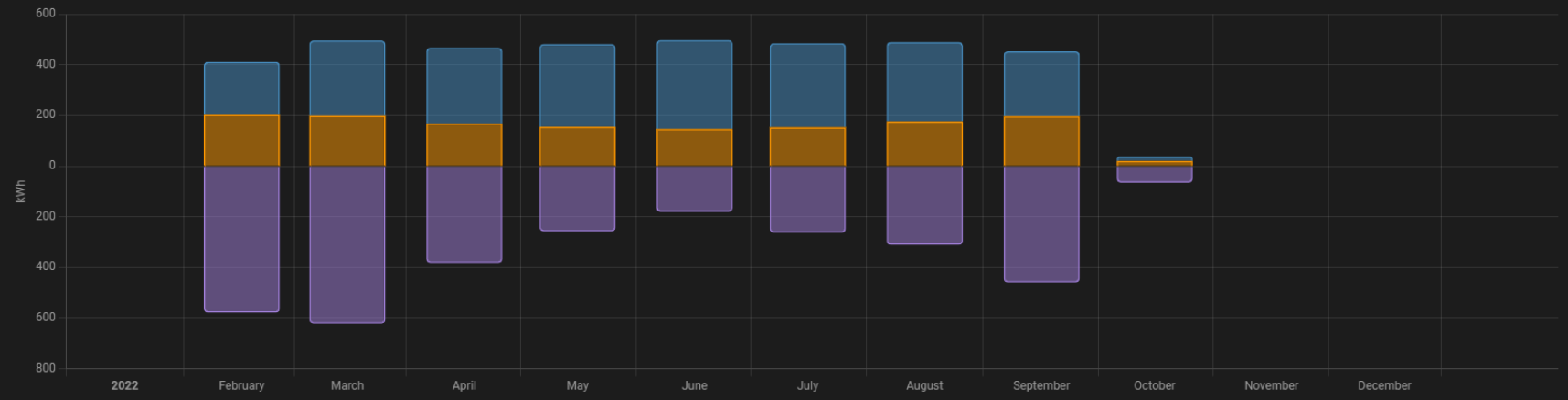
Do more when electricity is clean and less when its dirty



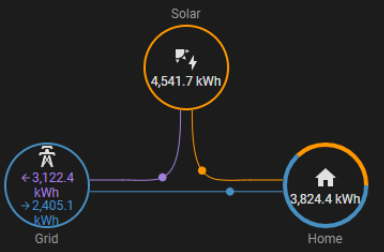
Varies by time

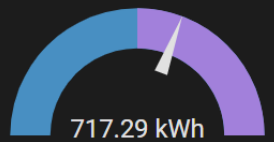


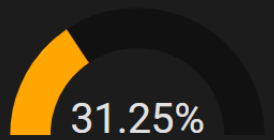
Energy usage



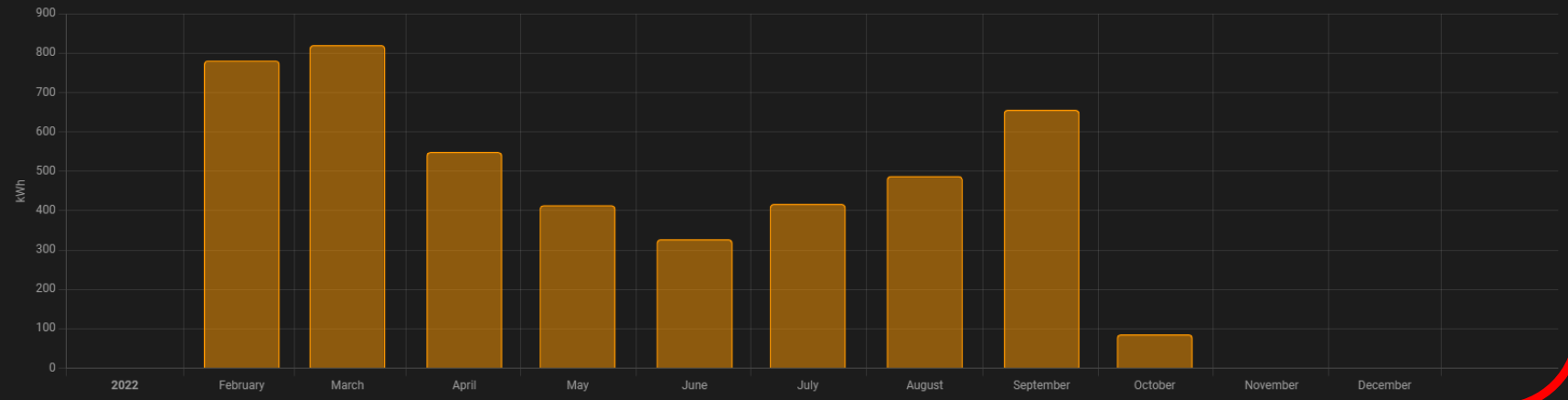
Energy distribution







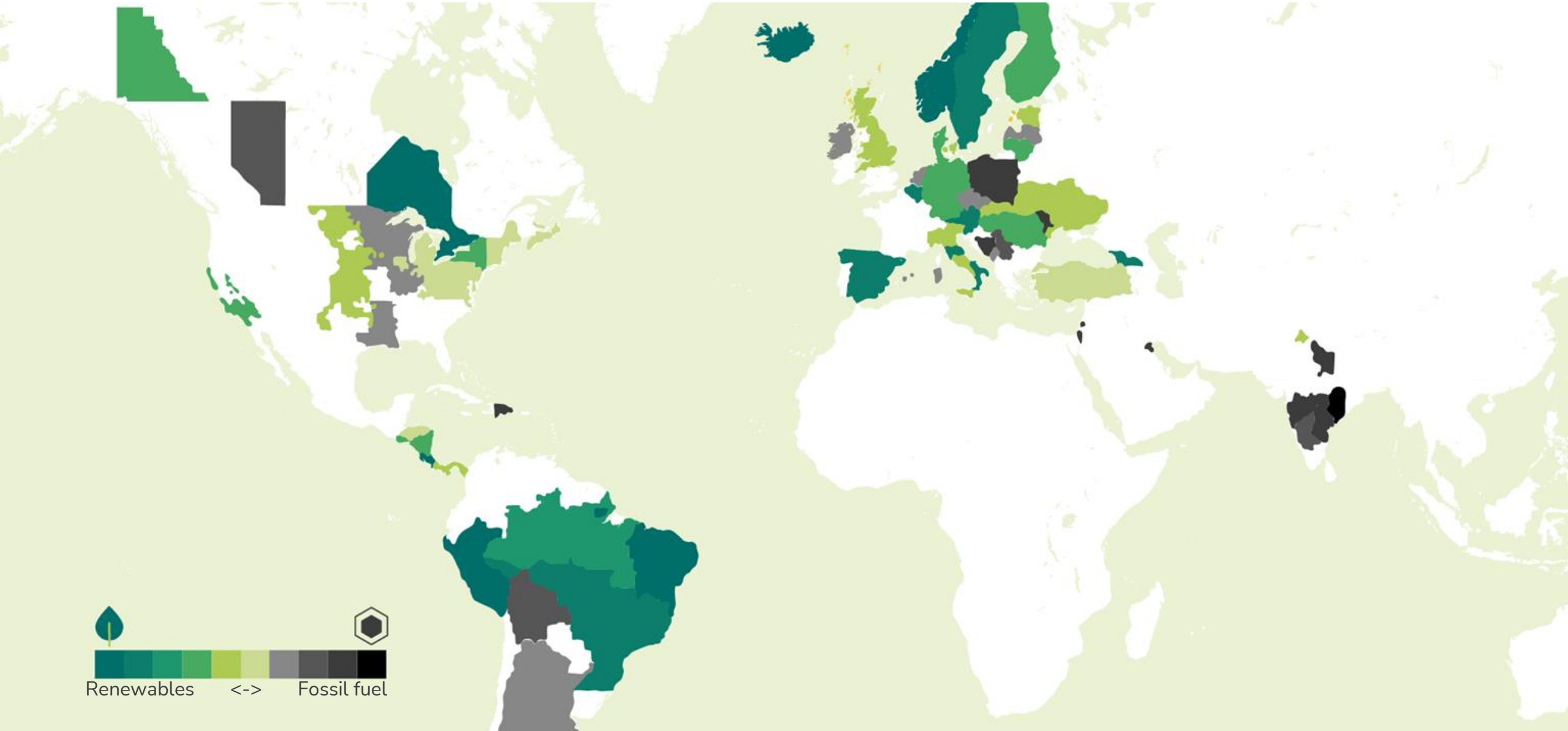
Solar production



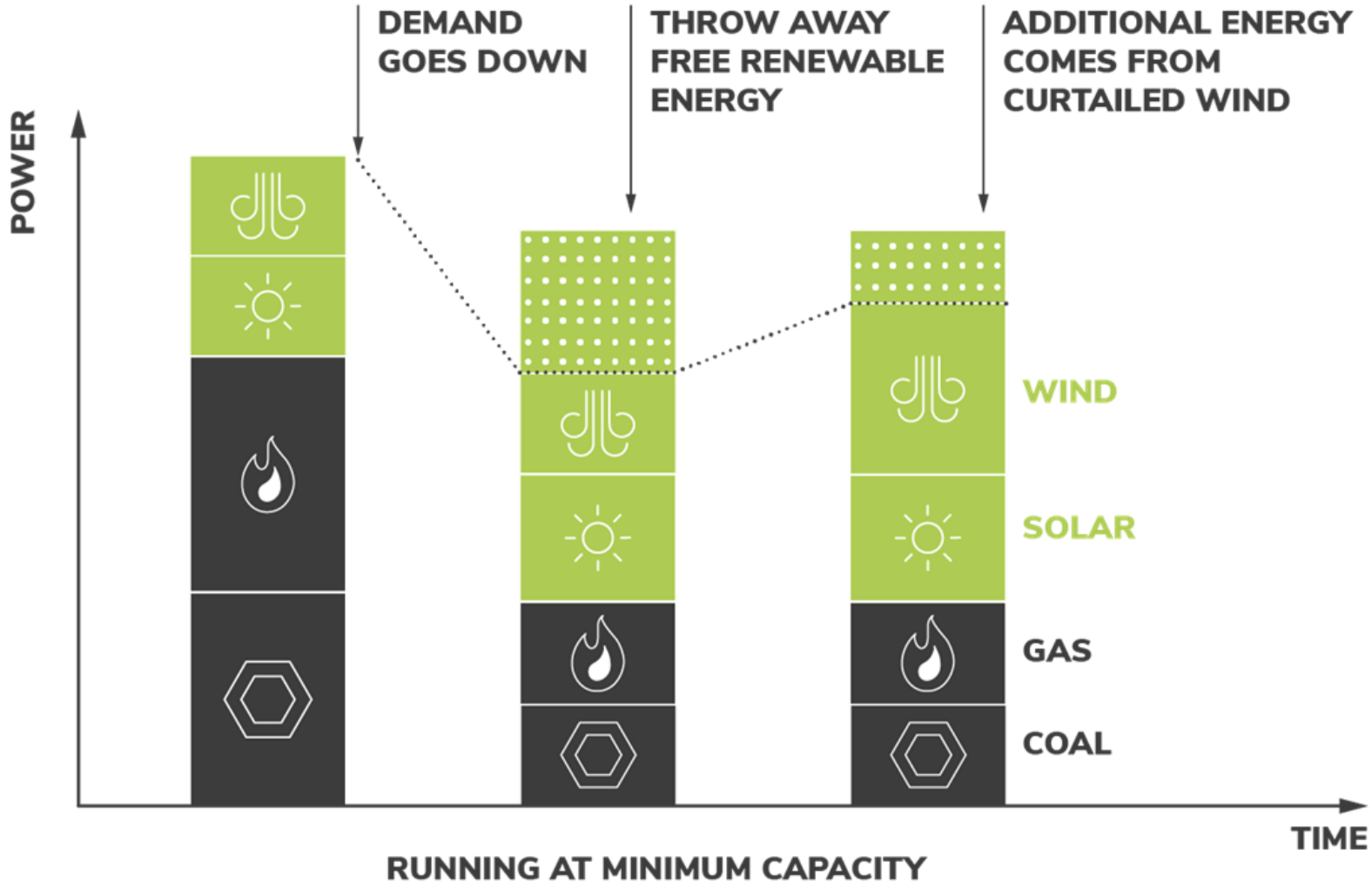
Sources

Source	Energy	Cost
SN: 3004189289 pv_gen_meter	4,541.72 kWh	
Solar total	**4,541.72 kWh**	
SN: 3004189289 metering_total_absorbed	2,405.13 kWh	A$528.53
SN: 3004189289 metering_total_yield	-3,122.43 kWh	-A$265.29
Grid total	**-717.29 kWh**	**A$263.24**

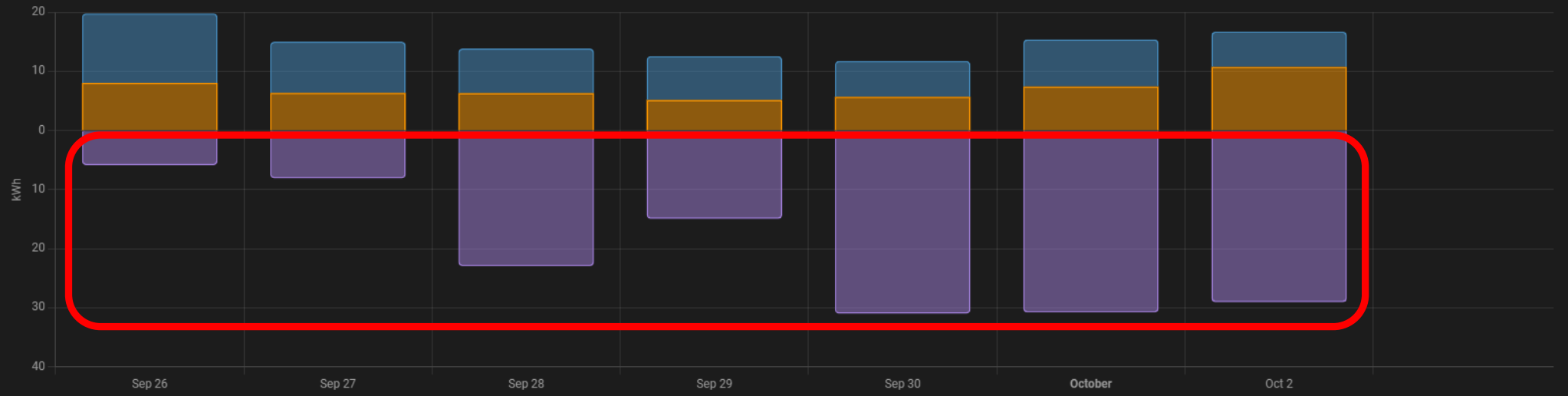
But also location



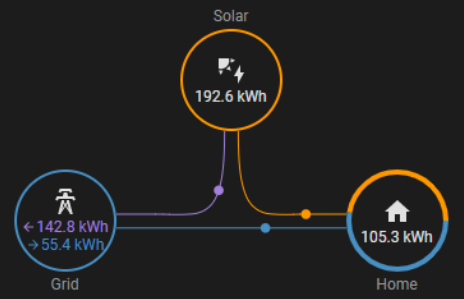
Sometimes we throw clean energy away

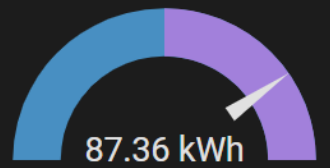


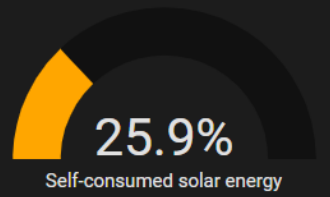
Energy usage



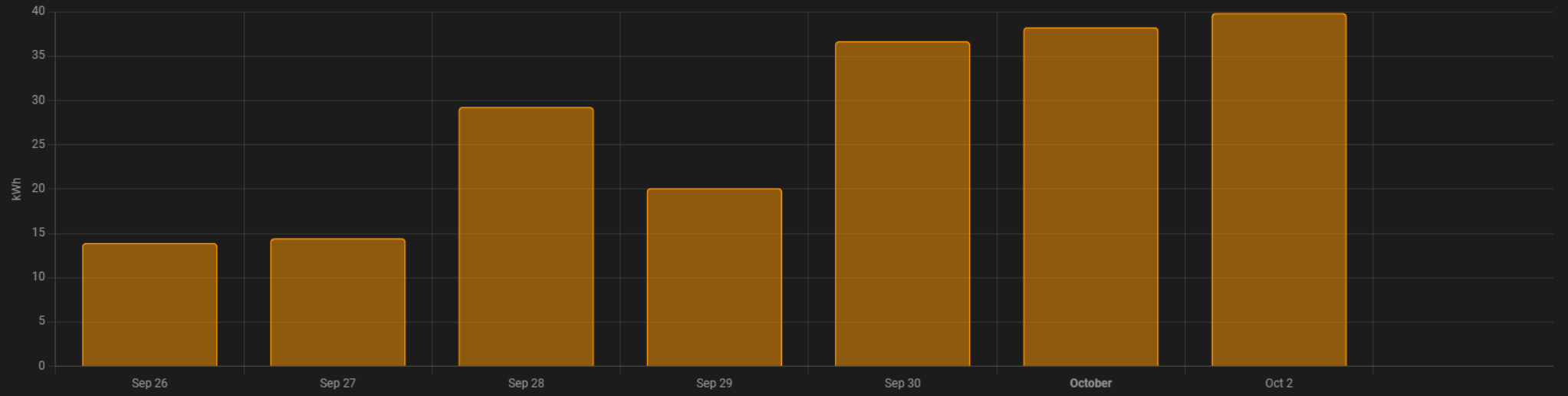
Energy distribution







Solar production



TIME

Calculating Carbon In Your Cloud Tech Stack

What is the SCI?

The SCI is a rate of carbon emissions, not a total and is bias towards actions that eliminates carbon emissions. The equation is a simple and elegant solution to the extremely complex problem behind it:

Energy consumed by a tech stack in kWh

Functional unit; this is how software scales, for example by user, device or API request

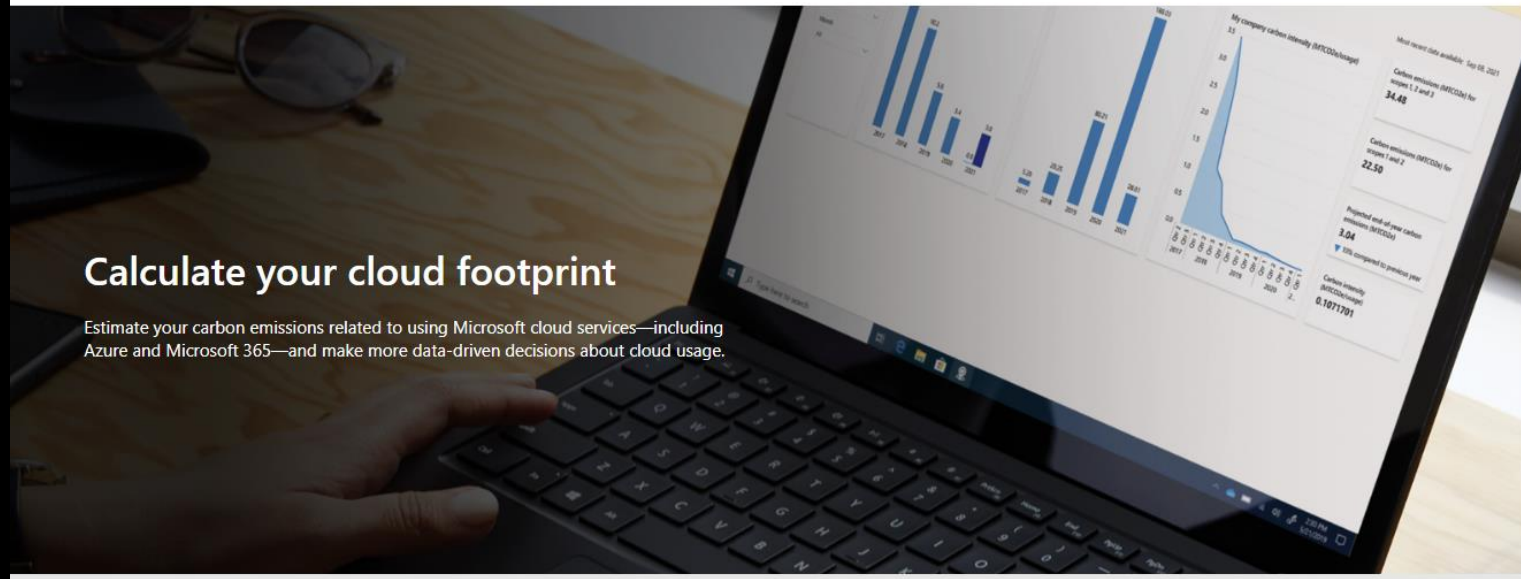
$$SCI = ((E * I) + M) \text{ per } R$$

Carbon emitted per kWh of energy, in gCO2 / kWh

Embodied carbon emissions from the creation (and destruction) of hardware that the software is running on.

Microsoft | Sustainability

Products & solutions | Sustainability guide | Learning center | Industries | Partners | Our approach | More | All Microsoft | Search



Calculate your cloud footprint

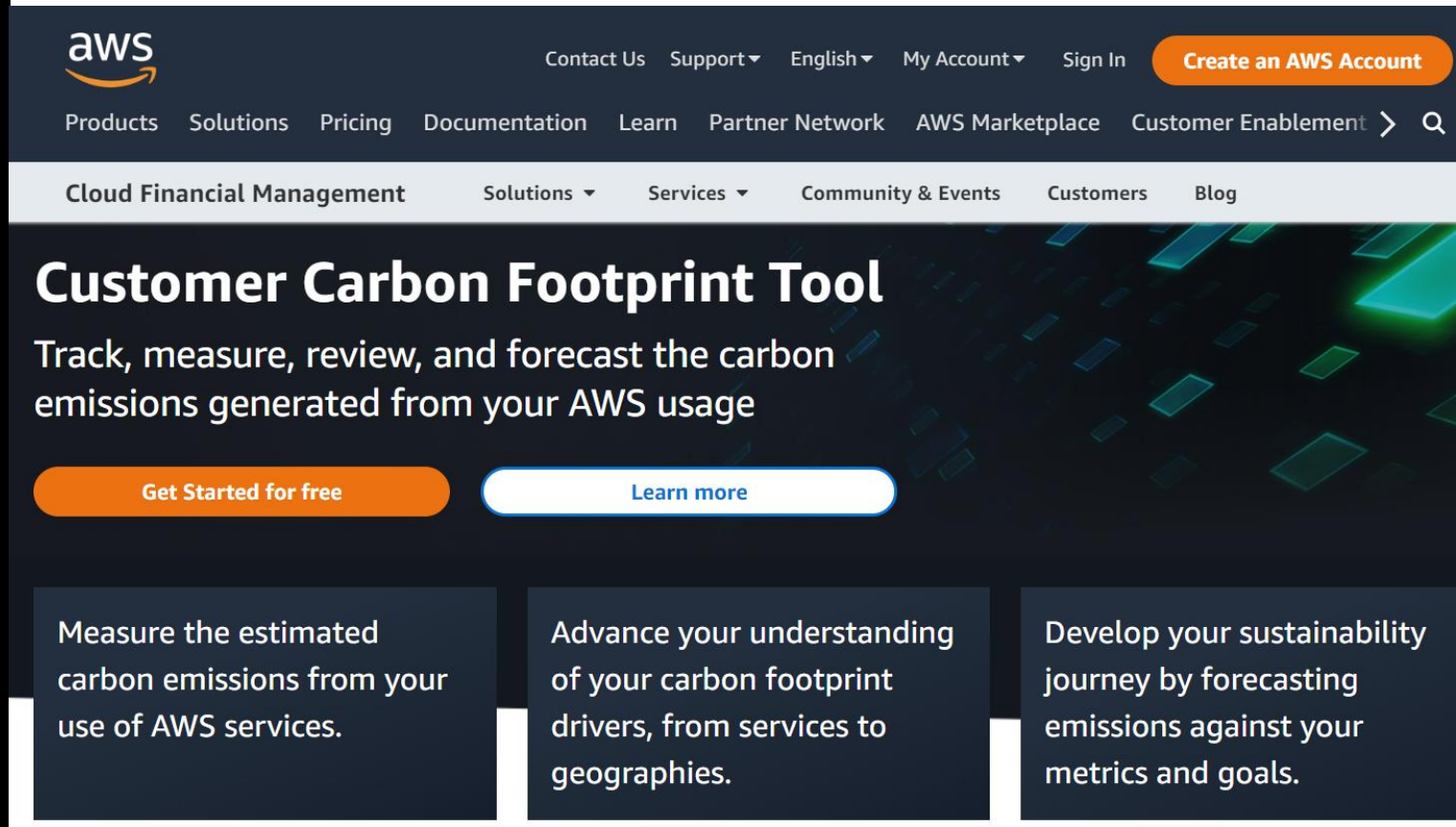
Estimate your carbon emissions related to using Microsoft cloud services—including Azure and Microsoft 365—and make more data-driven decisions about cloud usage.

Cloud carbon tracking for your path to net zero

Using Power BI template apps and consistent, third party–validated carbon accounting, the Emissions Impact Dashboard for Azure and the Emissions Impact Dashboard for Microsoft 365 (now in preview) help you measure your Microsoft Cloud–based emissions and carbon savings potential.

App for Azure > App for Microsoft 365 >


aws


aws
Contact Us
Support
English
My Account
Sign In
Create an AWS Account
Products
Solutions
Pricing
Documentation
Learn
Partner Network
AWS Marketplace
Customer Enablement
Cloud Financial Management
Solutions
Services
Community & Events
Customers
Blog
Customer Carbon Footprint Tool
Track, measure, review, and forecast the carbon emissions generated from your AWS usage
Get Started for free
Learn more
Measure the estimated carbon emissions from your use of AWS services.
Advance your understanding of your carbon footprint drivers, from services to geographies.
Develop your sustainability journey by forecasting emissions against your metrics and goals.

Google Cloud Overview Solutions Produc Docs Support English Console

Carbon Footprint Contact Us

Join us at the first-ever Google Cloud Sustainability Summit on June 28.

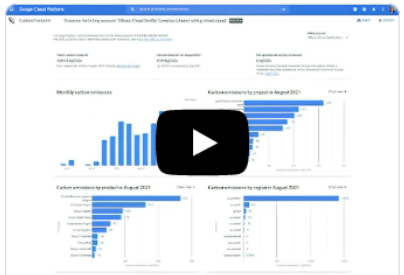
JUMP TO

Carbon Footprint PREVIEW

Measure, report, and reduce your cloud carbon emissions.

Go to console

- Include gross carbon emissions data in reports and disclosures
- Visualize carbon insights via dashboards and charts
- Reduce the gross emissions of cloud applications and infrastructure



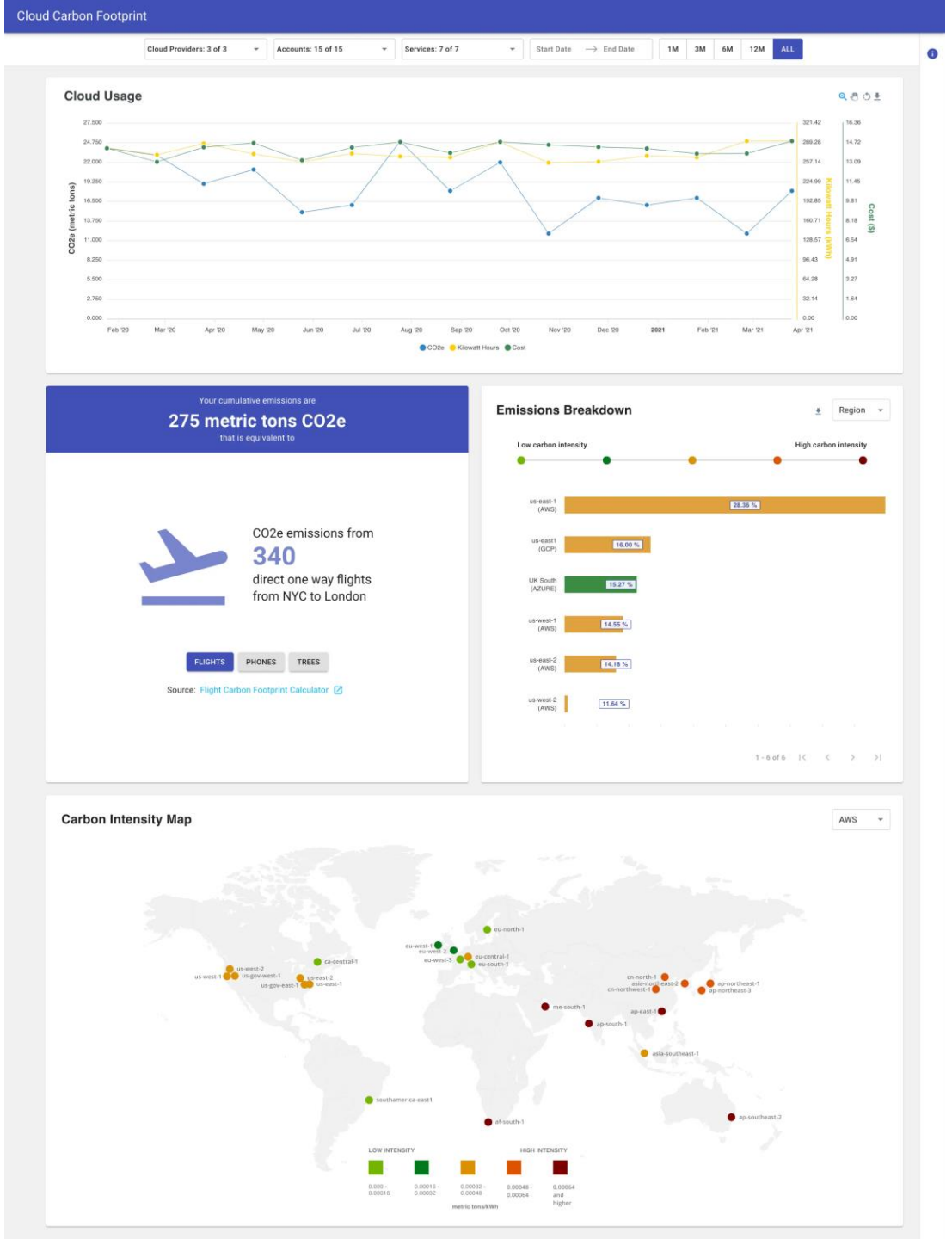
VIDEO

Watch this session and learn how to measure carbon emissions on Google Cloud

17:00

cloudcarbonfootprint.org

cloudcarbonfootprint.org



Carbon Aware SDK

Integrated

An SDK to enable the creation of carbon aware applications, applications that do more when the electricity is clean and do less when the electricity is dirty.

For developers to build carbon aware software, there is a need for a unified baseline to be implemented.

The Carbon Aware Core API will look to standardize and simplify carbon awareness for developers through a unified API, **command line interface**, and modular carbon-aware-logic plugin architecture.

CLI

Example 1 - Get the current emissions data for a specified location

```
$ ./CarbonAwareCLI -l westus -d "azure-emissions-data.json"
```

Response

```
[
  {
    "Location": "westus",
    "Time": "2021-11-17T04:45:11.5104572+00:00",
    "Rating": 31.0
  }
]
```

API

Method:

> *<The request type>*
>
> `GET` | `POST` | `DELETE` | `PUT`

Success Response:

> *<What should the status code be on success and is there any returned data? This is useful when people need to to know what their callbacks should expect!>*

- **Code:** 200 **Content:** `{ id : 12 }`

Carbon Aware CLI Reference

The following is the documentation for the Carbon Aware CLI

Format

```
$ CarbonAwareCLI -t <time> -l <location 1> <location 2> -d <path to data file>
```

Parameters

Short	Long	Required / Optional	Description
-l	--location	Required	The location is a comma seperated list of named locations or regions specific to the emissions data provided.
-d	--data-file	Required	Path to the emissions source data file
-t	--fromTime	Optional	The desired date and time to retrieve the emissions for. Defaults to 'now'.
-o	--output	Optional	Output format. Options: console, json. Default is `json`
-v	--verbose	Optional	Verbose output
	--lowest	Optional	Only return the results with the lowest emissions.

Examples

Example 1 - Get the current emissions data for a specified location

```
$ ./CarbonAwareCLI -l westus -d "azure-emissions-data.json"
```

Response

```
[
  {
    "Location": "westus",
    "Time": "2021-11-17T04:45:11.5104572+00:00",
    "Rating": 31.0
  }
]
```

Example 2 - Get the current emissions for multiple locations

```
$ ./CarbonAwareCLI -l westus eastus -d "azure-emissions-data.json"
```

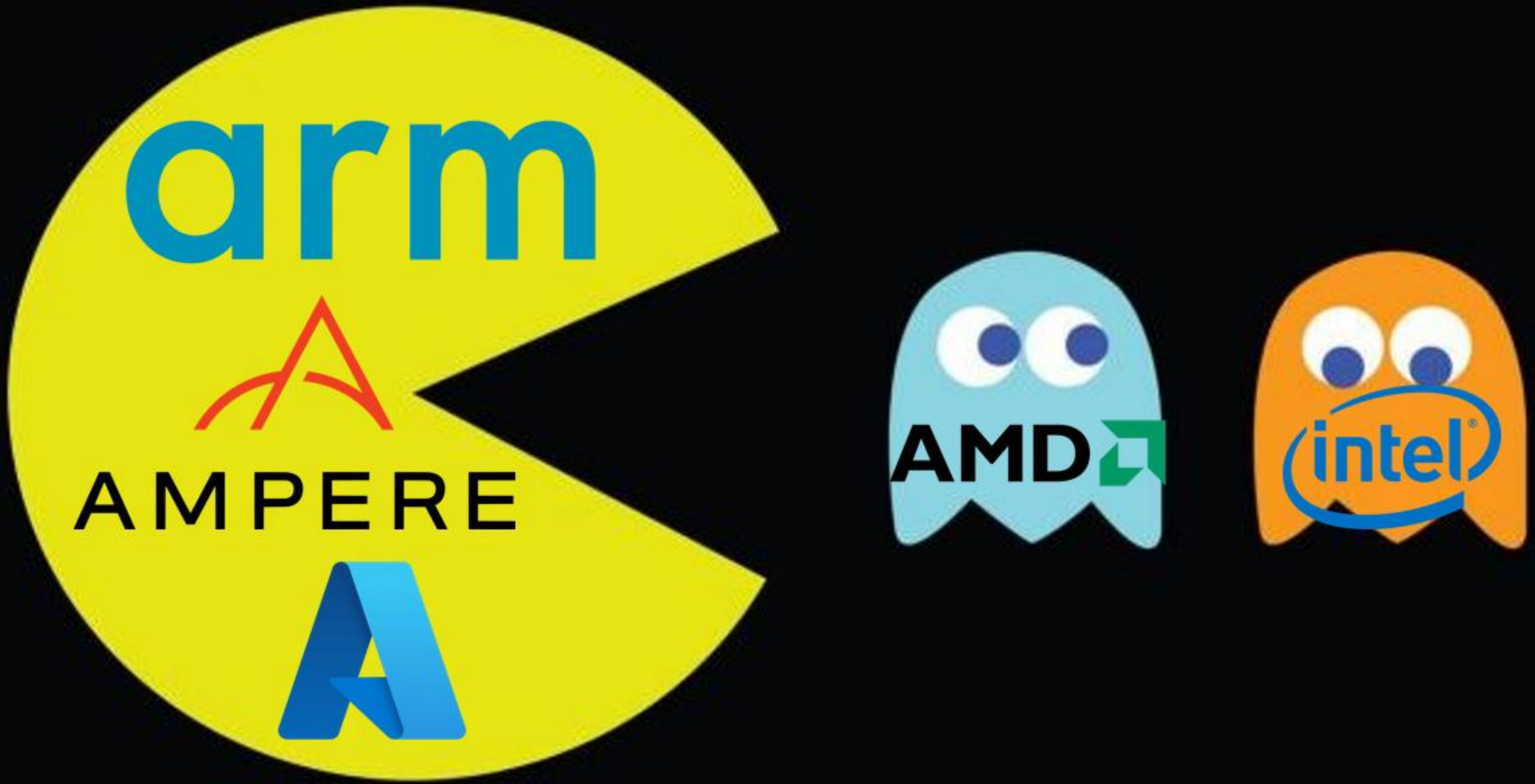
Response

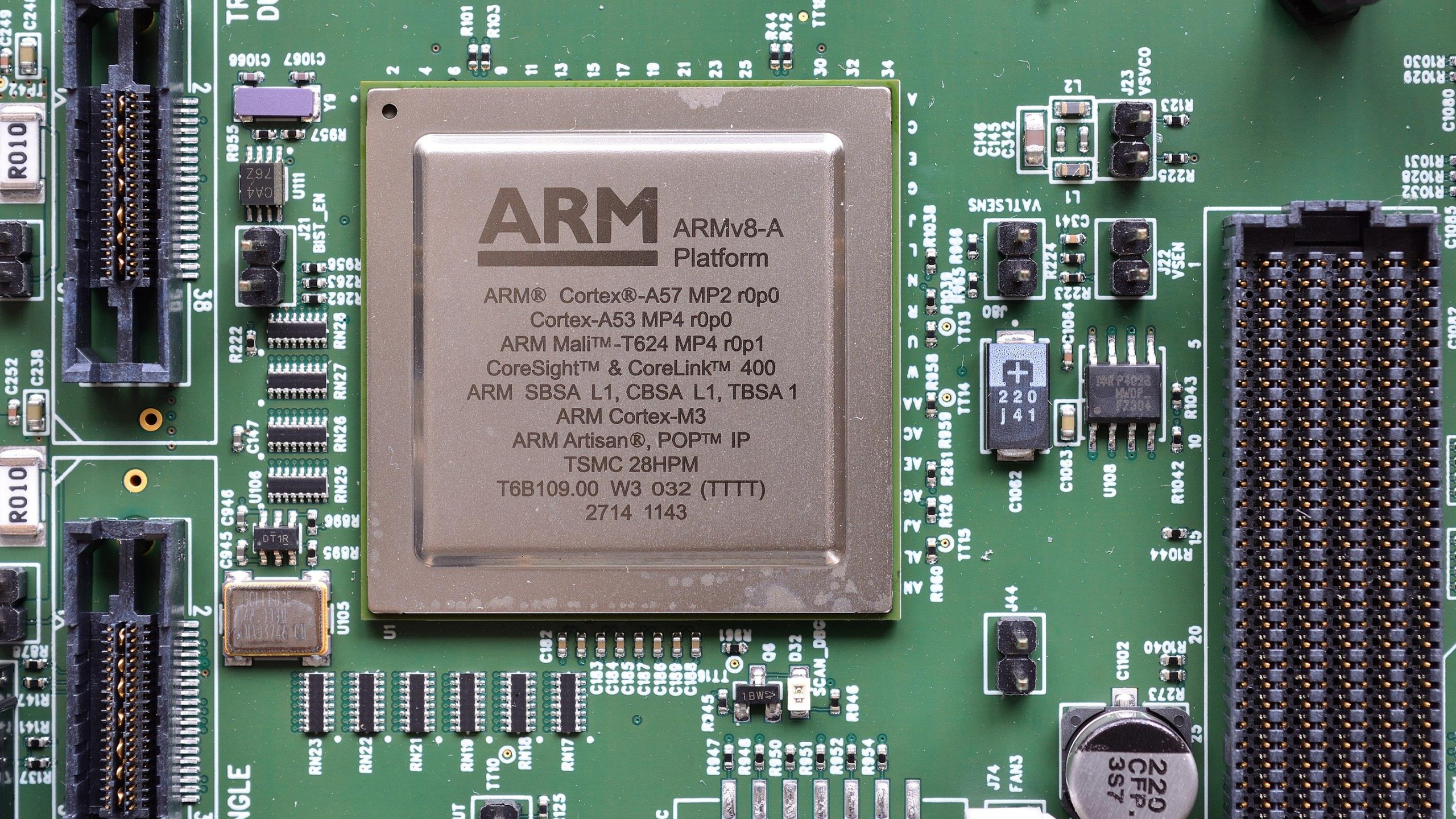
```
[
  {
    "Location": "westus",
    "Time": "2021-11-17T04:45:11.5104572+00:00",
    "Rating": 31.0
  },
  {
    "Location": "eastus",
    "Time": "2021-11-17T04:45:11.509182+00:00",
    "Rating": 59.0
  }
]
```

```
# Set the base image as the .NET 6.0 SDK (this includes the runtime)
FROM mcr.microsoft.com/dotnet/sdk:6.0 as build-env

# Copy everything and publish the release (publish implicitly restores and builds)
COPY ./src/ ./
COPY ./entrypoint.sh ./


#WORKDIR /src/

RUN dotnet publish  ./CarbonAware.CLI/CarbonAware.CLI.csproj -c Release -o out --no-self-contained
RUN cp ./CarbonAware.CLI/carbon-aware.json out
RUN cp -r  ./data/data-files/ out

RUN cp ./entrypoint.sh out


# Label the container
#LABEL maintainer="Green-Software-Foundation"
LABEL repository="https://github.com/Green-Software-Foundation/carbon-aware-sdk"
LABEL homepage="https://github.com/Green-Software-Foundation/carbon-aware-sdk"

# Label as GitHub action
LABEL com.github.actions.name="CarbonAware"
LABEL com.github.actions.description="A Github Action to enable the creation of carbon aware applications, applications that do more when the electricity is clean and do less
LABEL com.github.actions.icon="sliders"
LABEL com.github.actions.color="purple"

# Relayer the .NET SDK, anew with the build output
FROM mcr.microsoft.com/dotnet/runtime:6.0
COPY --from=build-env /out .
RUN apt-get update && apt-get install jq -y

RUN chmod +x entrypoint.sh
#ENTRYPOINT ["/CarbonAwareCLI"]
ENTRYPOINT ["/entrypoint.sh"]
```

Energy Efficiency

Consume the least amount of electricity possible

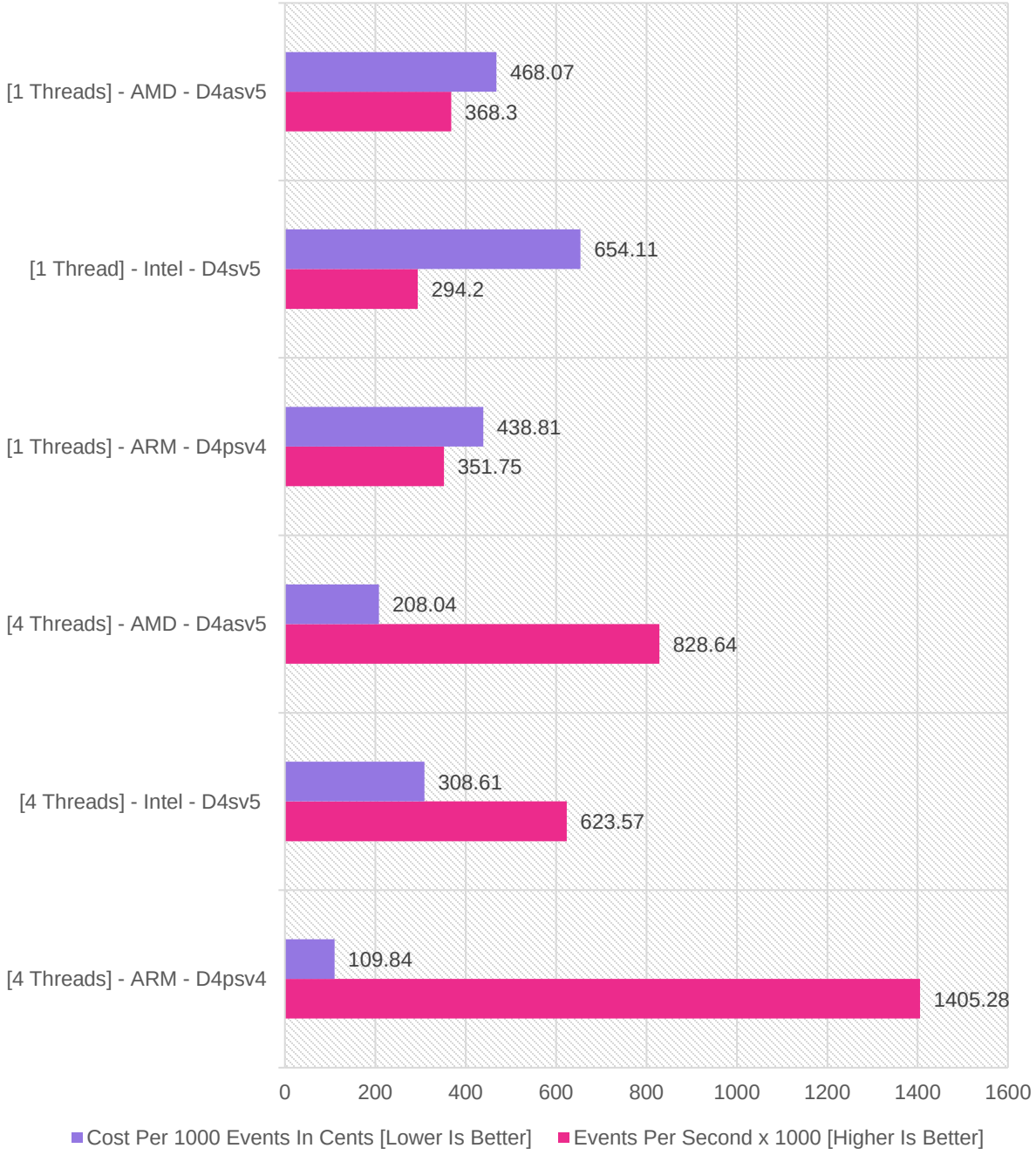

arm
AMPERE
AMD
intel


ARM
ARMv8-A
Platform
ARM® Cortex®-A57 MP2 r0p0
Cortex-A53 MP4 r0p0
ARM Mali™-T624 MP4 r0p1
CoreSight™ & CoreLink™ 400
ARM SBSA L1, CBSA L1, TBSA 1
ARM Cortex-M3
ARM Artisan®, POP™ IP
TSMC 28HPM
T6B109.00 W3 032 (TTTT)
2714 1143

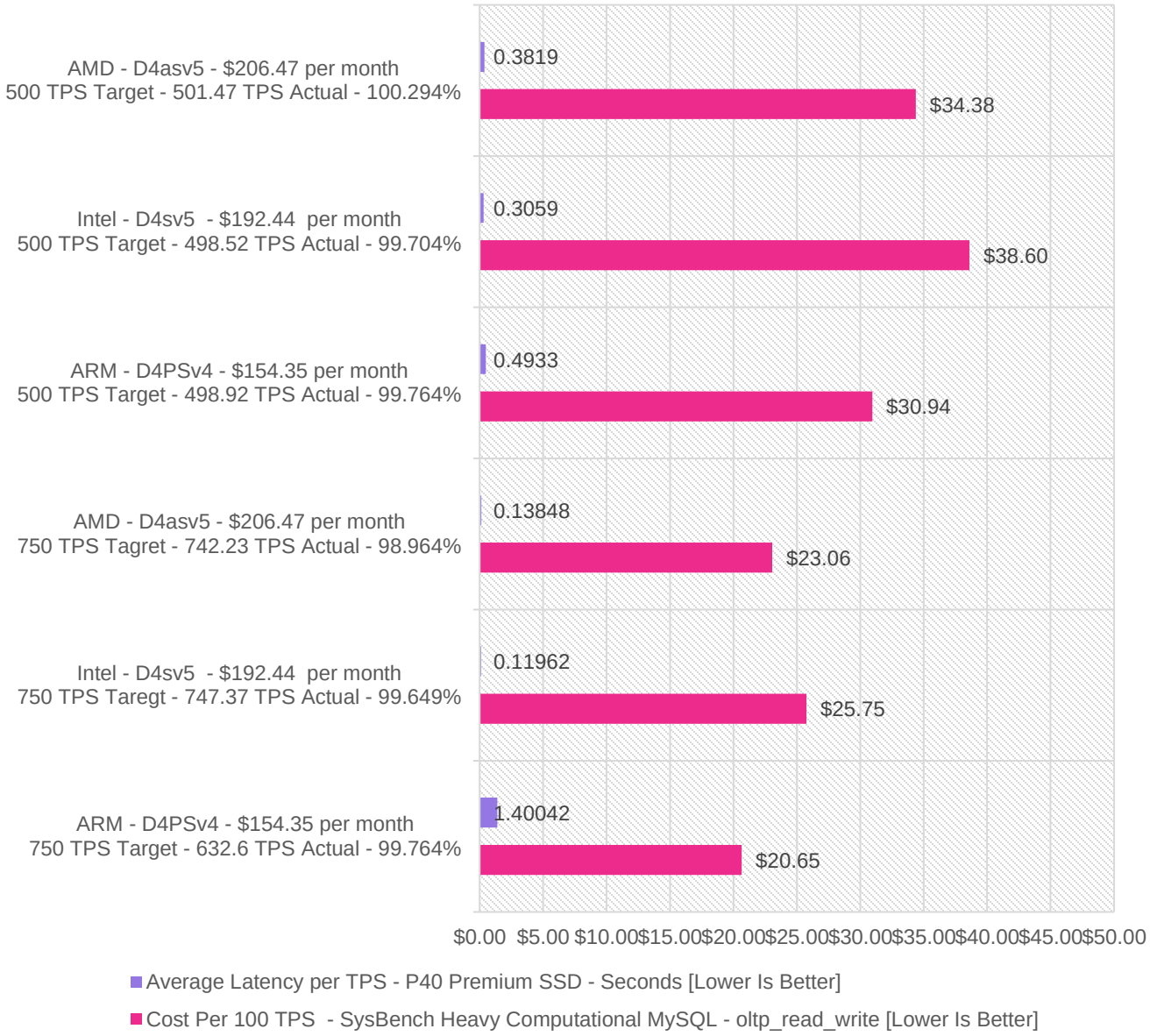
Azure SKU – CPU Architecture	Cores / Threads	Clock Rate	TDP	Watts Per Core
Dpsv5 – Ampere Altra Q80-30	80 (80 Threads)	3.0 GHz	210 W	2.625W
Dsv5 – Intel® Xeon® Platinum 8370C (Ice Lake)	64 (128 Threads)	3.5 GHz	270 W	4.218 W
Dasv5 – AMD's 3rd Generation EPYC™ 7763v	64 (128 Threads)	3.5 GHz	280 W	4.375 W

Comparison of Azure VM SKU's – Cores / Threads vs. Frequency vs. TDP

SysBench CPU Prime Number Generation



SysBench Heavy Computational MySQL - oltp_read_write
Cost Per 100 TPS vs. Latency (Seconds)
P40 Premium SSD



```
fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca c
mov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx p
dpe1gb rdtscp lm constant_tsc rep_good nopl xtopology tsc_
reliable nonstop_tsc cpuid aperfmperf pni pclmulqdq vmx ss
se3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_de
adline_timer aes xsave avx f16c rdrand hypervisor lahf_lm
abm 3dnowprefetch invpcid_single tpr_shadow vnmi ept vpid
ept_ad fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms in
vpcid rtm avx512f avx512dq rdseed adx smap avx512ifma clfl
ushopt clwb avx512cd sha_ni avx512bw avx512vl xsaveopt xsa
vec xgetbv1 xsaves avx512vbmi umip avx512_vbmi2 gfni vaes
vpclmulqdq avx512_vnni avx512_bitalg avx512_vpopcntdq la57
rdpid fsrm arch_capabilities
```

```
fp asimd evtstrm aes pmull sha1 sha2 crc32 atomics fphp a simdhp
cpuid asimdrdm lrcpc dcpop asimddp
```

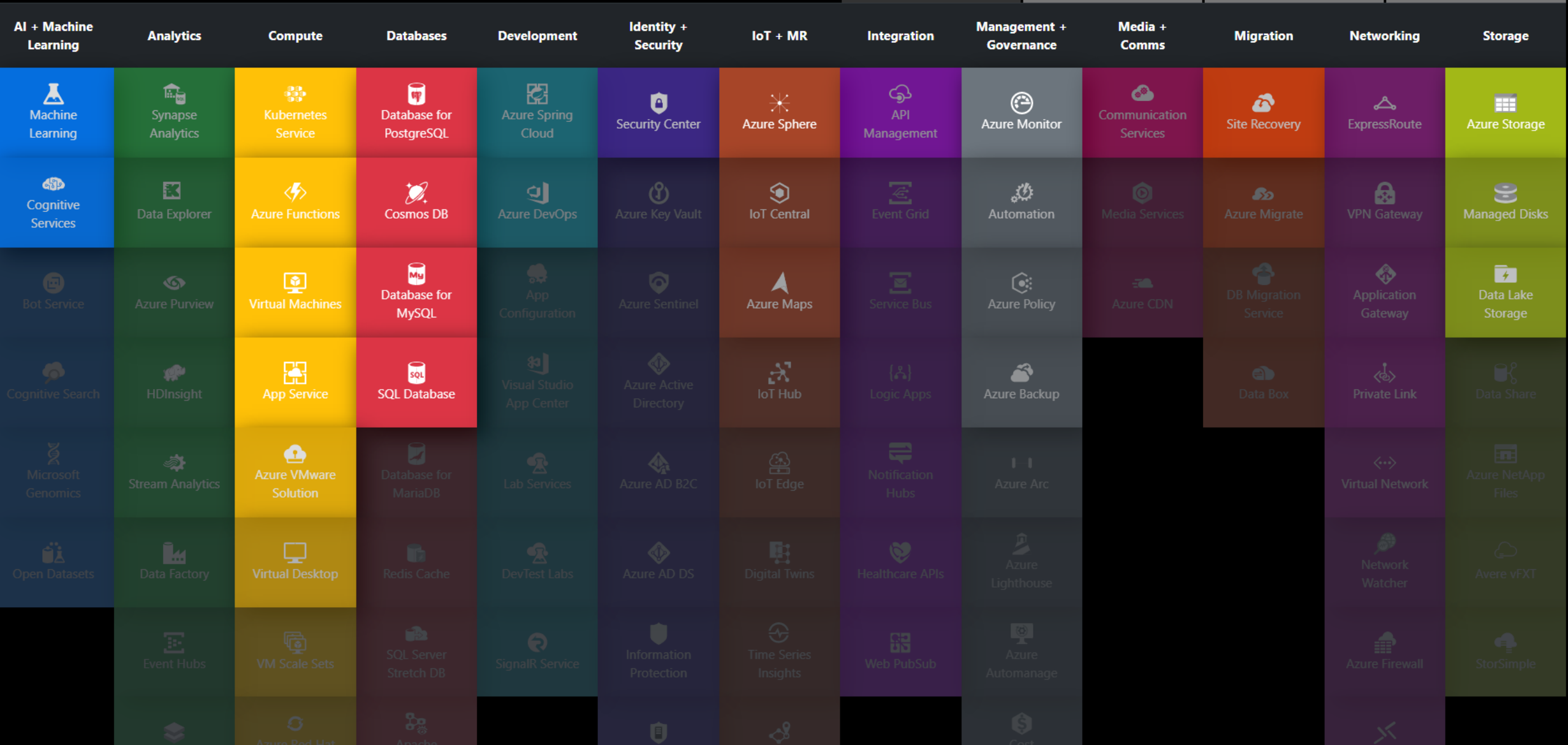
Architectural Levers

Do more with less

Azure Heat Map

Azure Updates data for last 6 months visualized. Rebuilt 51 minutes 34 seconds ago.

ALL UPDATES EQUAL | LATEST MORE IMPORTANT | ONLY LAST 7 DAYS | ALL MENTIONS



Reduce carbon intensity with Cloud Changes

Report current infrastructure → **Identify overhead** → **”Right-size” infrastructure**

03:00 on May 19th 2022

15:00 on May 19th 2022

Source of electrical data: energymap.org

RIGHT-SIZING YOUR AZURE CLOUD INFRASTRUCTURE

Virtual Machine
Ensure machines are right-sized.

- D2s_v3: 4.302 kWh (in 24 hours), 2,275.76 gCO_2eq, **A$7* for 24 hours**
- NC6s_v3: 3.3 kWh (in 24 hours), 1,745.7 gCO_2eq, **A$149* for 24 hours**

App Services
Shift suitable services to app services.

Containers
Shift suitable services to containers on demand.

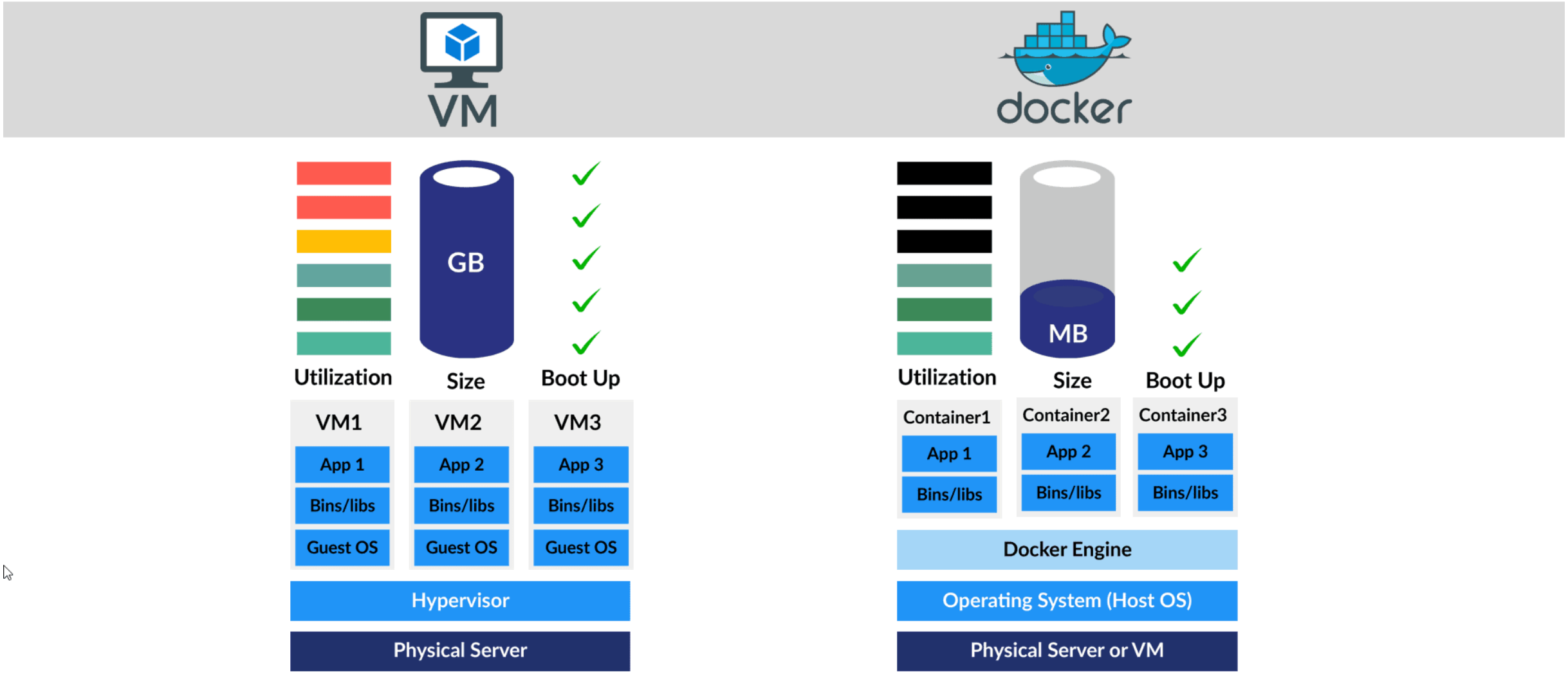
Functions
Shift to a serverless architecture, run on demand.

- Azure Function: 0.886 kWh (in 24 hours), 468.69 gCO_2eq, **A$0.68* for 24 hours**

Azure Sustainability APIs
Query carbon emissions for each service to carbon-shift software.

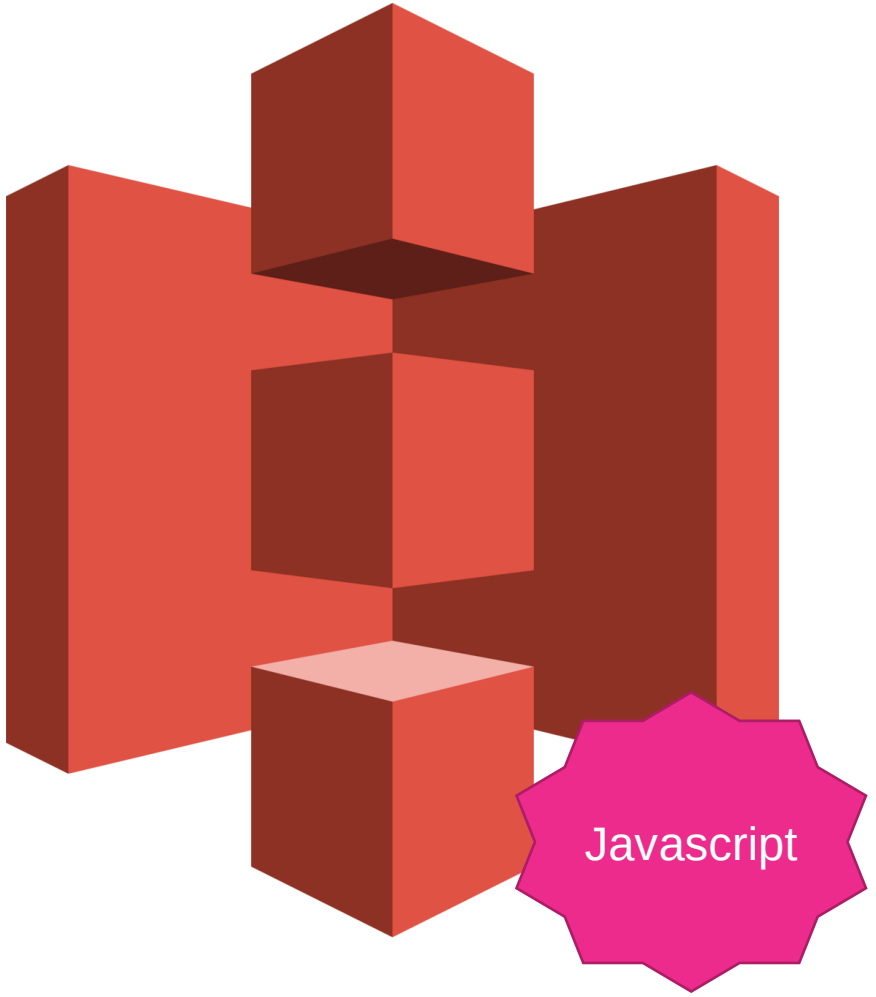
Source of cost data: Azure Calculator, as at 19th May 2022, in Australia Southeast & East region. Cost indicative only

VM's vs Containers vs Serverless – Its all tin.

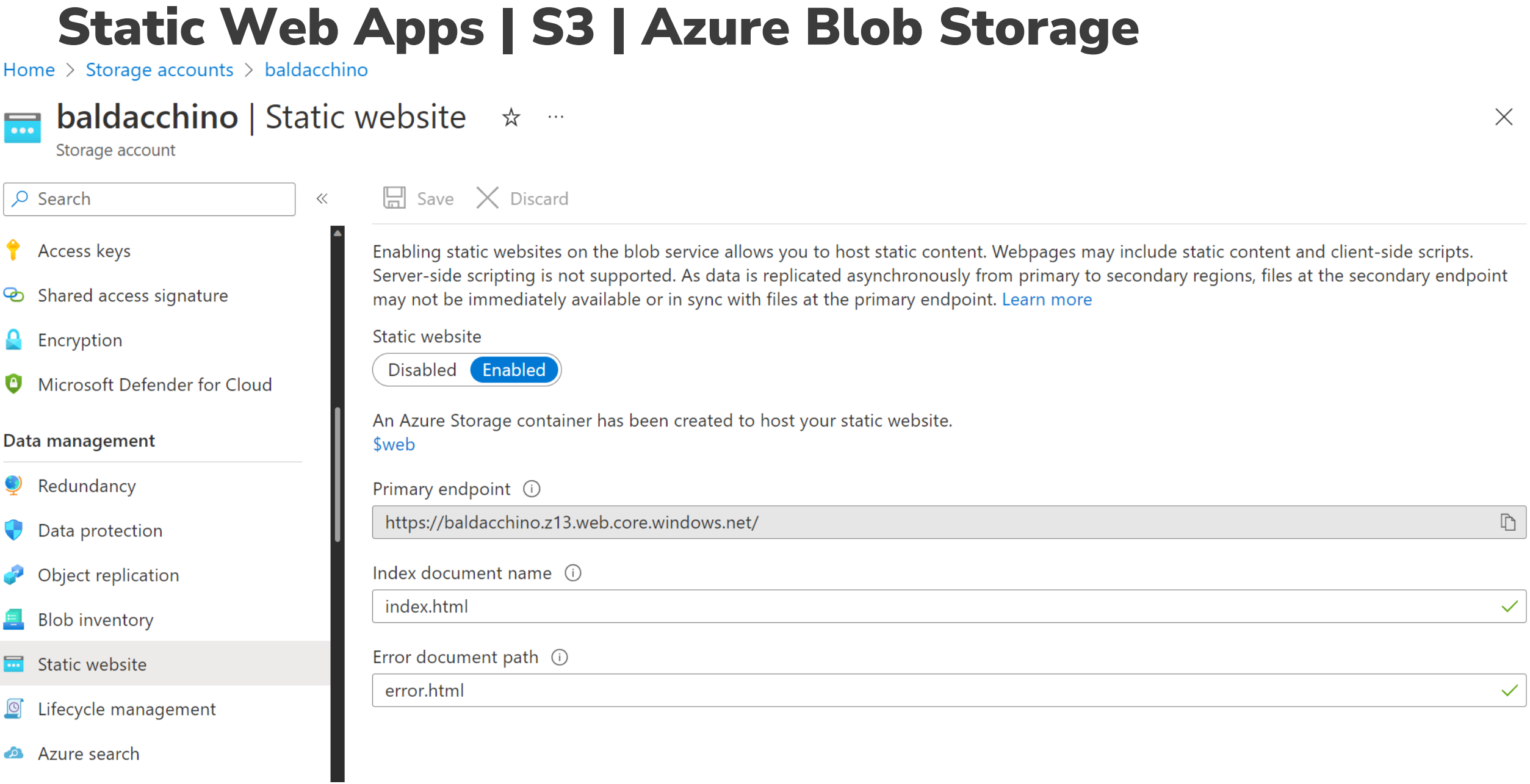


Eliminate Your Web Server Tier

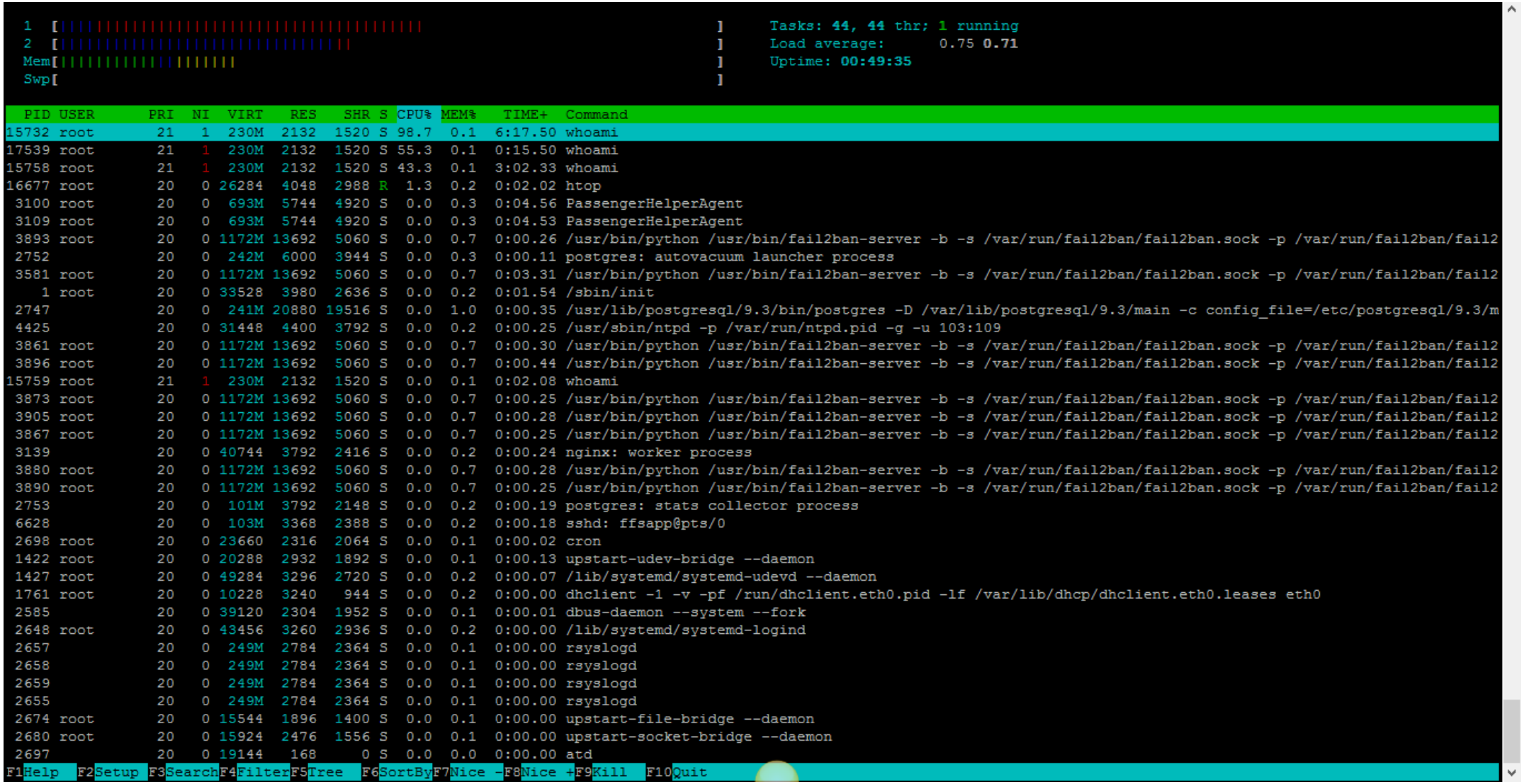
Static Web Apps | S3 | Azure Blob Storage



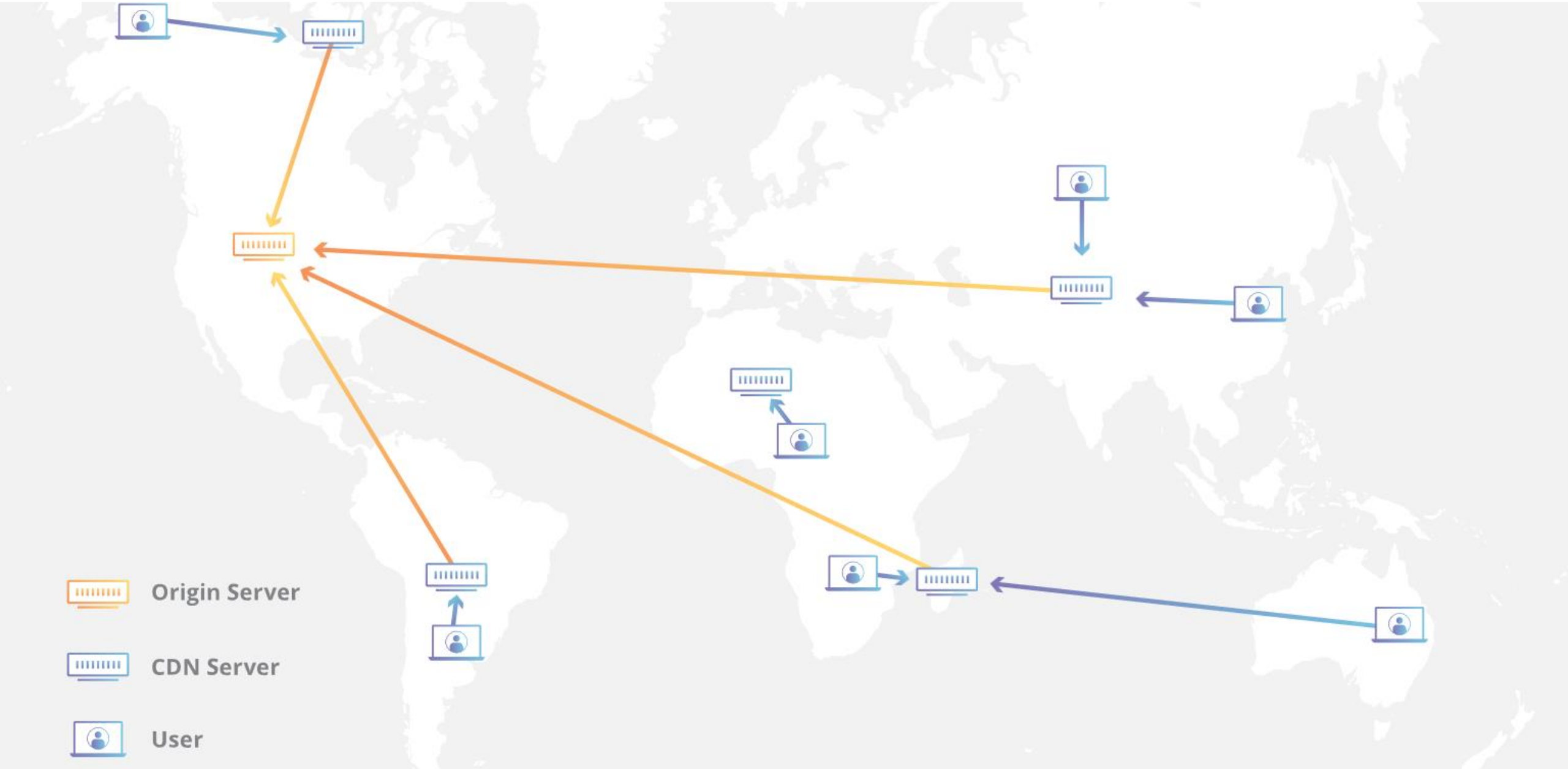
Static Web Apps | S3 | Azure Blob Storage



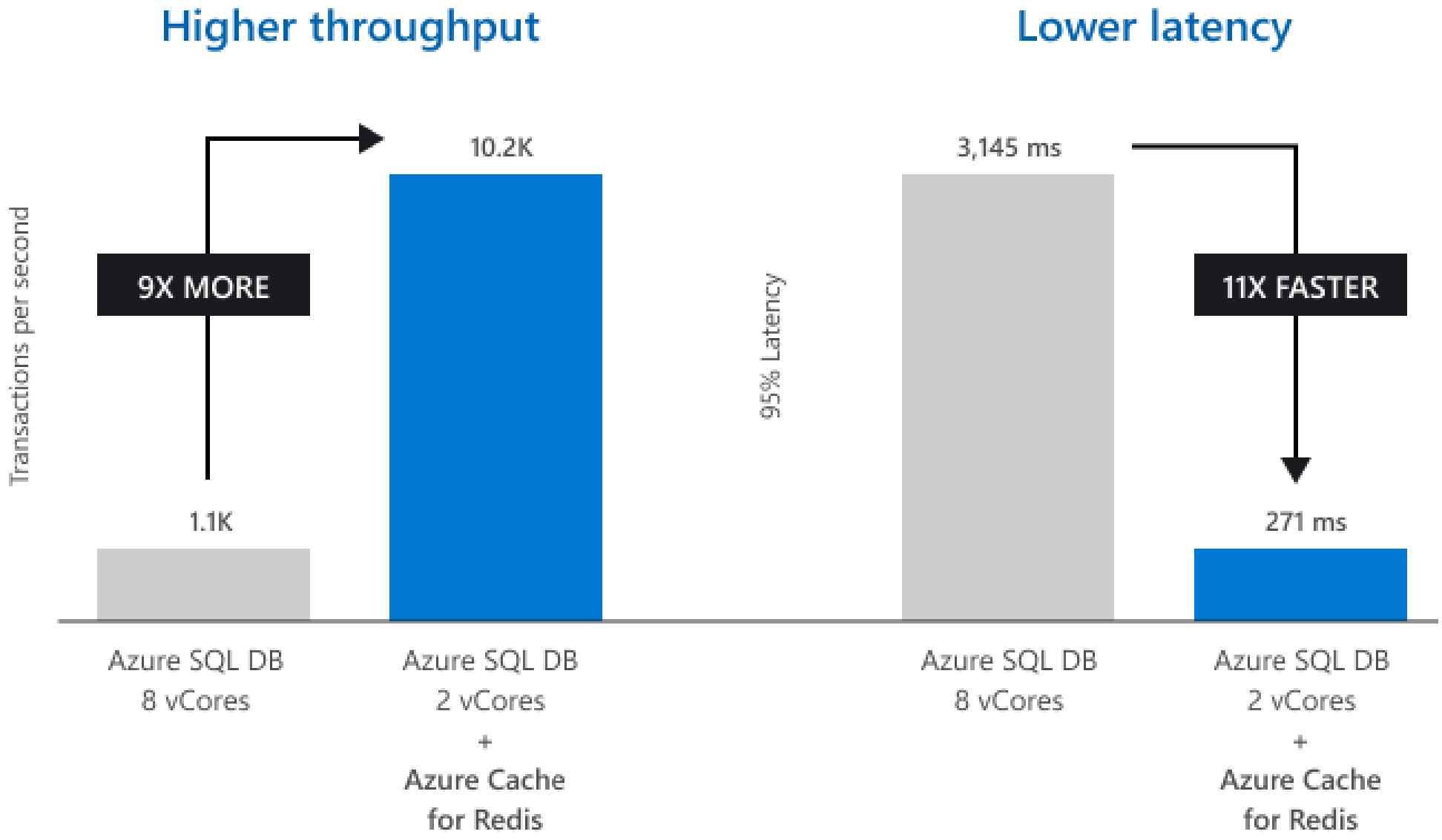
Run Hotter


1 [
2 [
Mem[
Swp[
Tasks: 44, 44 thr; 1 running
Load average: 0.75 0.71
Uptime: 00:49:35
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
15732 root 21 1 230M 2132 1520 S 98.7 0.1 6:17.50 whoami
17539 root 21 1 230M 2132 1520 S 55.3 0.1 0:15.50 whoami
15758 root 21 1 230M 2132 1520 S 43.3 0.1 3:02.33 whoami
16677 root 20 0 26284 4048 2988 R 1.3 0.2 0:02.02 htop
3100 root 20 0 693M 5744 4920 S 0.0 0.3 0:04.56 PassengerHelperAgent
3109 root 20 0 693M 5744 4920 S 0.0 0.3 0:04.53 PassengerHelperAgent
3893 root 20 0 1172M 13692 5060 S 0.0 0.7 0:00.26 /usr/bin/python /usr/bin/fail2ban-server -b -s /var/run/fail2ban/fail2ban.sock -p /var/run/fail2ban/fail2
2752 20 0 242M 6000 3944 S 0.0 0.3 0:00.11 postgres: autovacuum launcher process
3581 root 20 0 1172M 13692 5060 S 0.0 0.7 0:03.31 /usr/bin/python /usr/bin/fail2ban-server -b -s /var/run/fail2ban/fail2ban.sock -p /var/run/fail2ban/fail2
1 root 20 0 33528 3980 2636 S 0.0 0.2 0:01.54 /sbin/init
2747 20 0 241M 20880 19516 S 0.0 1.0 0:00.35 /usr/lib/postgresql/9.3/bin/postgres -D /var/lib/postgresql/9.3/main -c config_file=/etc/postgresql/9.3/m
4425 20 0 31448 4400 3792 S 0.0 0.2 0:00.25 /usr/sbin/ntpd -p /var/run/ntpd.pid -g -u 103:109
3861 root 20 0 1172M 13692 5060 S 0.0 0.7 0:00.30 /usr/bin/python /usr/bin/fail2ban-server -b -s /var/run/fail2ban/fail2ban.sock -p /var/run/fail2ban/fail2
3896 root 20 0 1172M 13692 5060 S 0.0 0.7 0:00.44 /usr/bin/python /usr/bin/fail2ban-server -b -s /var/run/fail2ban/fail2ban.sock -p /var/run/fail2ban/fail2
15759 root 21 1 230M 2132 1520 S 0.0 0.1 0:02.08 whoami
3873 root 20 0 1172M 13692 5060 S 0.0 0.7 0:00.25 /usr/bin/python /usr/bin/fail2ban-server -b -s /var/run/fail2ban/fail2ban.sock -p /var/run/fail2ban/fail2
3905 root 20 0 1172M 13692 5060 S 0.0 0.7 0:00.28 /usr/bin/python /usr/bin/fail2ban-server -b -s /var/run/fail2ban/fail2ban.sock -p /var/run/fail2ban/fail2
3867 root 20 0 1172M 13692 5060 S 0.0 0.7 0:00.25 /usr/bin/python /usr/bin/fail2ban-server -b -s /var/run/fail2ban/fail2ban.sock -p /var/run/fail2ban/fail2
3139 20 0 40744 3792 2416 S 0.0 0.2 0:00.24 nginx: worker process
3880 root 20 0 1172M 13692 5060 S 0.0 0.7 0:00.28 /usr/bin/python /usr/bin/fail2ban-server -b -s /var/run/fail2ban/fail2ban.sock -p /var/run/fail2ban/fail2
3890 root 20 0 1172M 13692 5060 S 0.0 0.7 0:00.25 /usr/bin/python /usr/bin/fail2ban-server -b -s /var/run/fail2ban/fail2ban.sock -p /var/run/fail2ban/fail2
2753 20 0 101M 3792 2148 S 0.0 0.2 0:00.19 postgres: stats collector process
6628 20 0 103M 3368 2388 S 0.0 0.2 0:00.18 sshd: ffsapp@pts/0
2698 root 20 0 23660 2316 2064 S 0.0 0.1 0:00.02 cron
1422 root 20 0 20288 2932 1892 S 0.0 0.1 0:00.13 upstart-udev-bridge --daemon
1427 root 20 0 49284 3296 2720 S 0.0 0.2 0:00.07 /lib/systemd/systemd-udevd --daemon
1761 root 20 0 10228 3240 944 S 0.0 0.2 0:00.00 dhclient -1 -v -pf /run/dhclient.eth0.pid -lf /var/lib/dhcp/dhclient.eth0.leases eth0
2585 20 0 39120 2304 1952 S 0.0 0.1 0:00.01 dbus-daemon --system --fork
2648 root 20 0 43456 3260 2936 S 0.0 0.2 0:00.00 /lib/systemd/systemd-logind
2657 20 0 249M 2784 2364 S 0.0 0.1 0:00.00 rsyslogd
2658 20 0 249M 2784 2364 S 0.0 0.1 0:00.00 rsyslogd
2659 20 0 249M 2784 2364 S 0.0 0.1 0:00.00 rsyslogd
2655 20 0 249M 2784 2364 S 0.0 0.1 0:00.00 rsyslogd
2674 root 20 0 15544 1896 1400 S 0.0 0.1 0:00.00 upstart-file-bridge --daemon
2680 root 20 0 15924 2476 1556 S 0.0 0.1 0:00.00 upstart-socket-bridge --daemon
2697 20 0 19144 168 0 S 0.0 0.0 0:00.00 atd
F1Help F2Setup F3SearchF4FilterF5Tree F6SortByF7Nice -F8Nice +F9Kill F10Quit

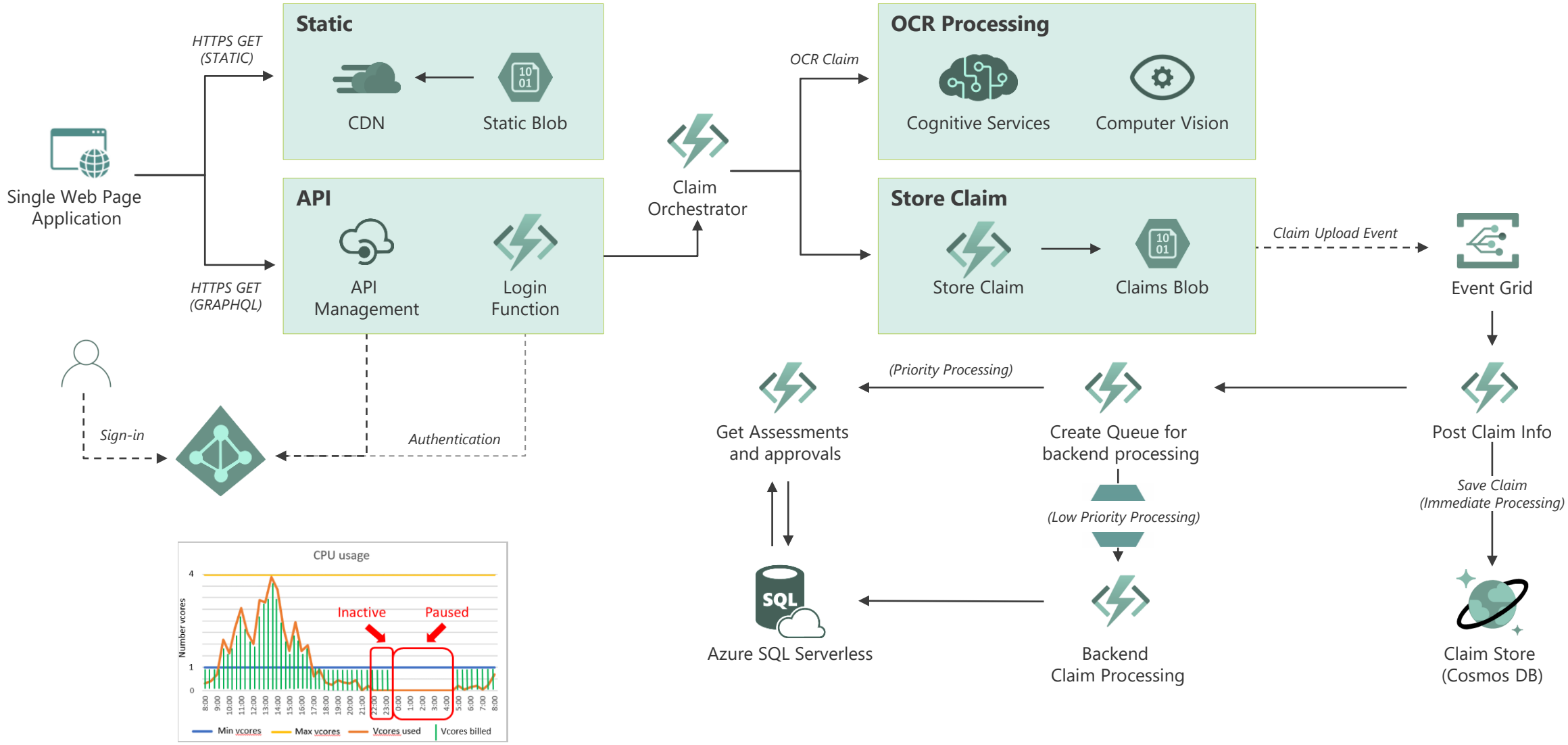
Cache – Everywhere - CDN



Cache – Everywhere – In Memory Cache



Sustainable Architecture



Your Language Matters

	Energy
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(c) Chapel	2.18
(v) Lisp	2.27
(c) Ocaml	2.40
(c) Fortran	2.52
(c) Swift	2.79
(c) Haskell	3.10
(v) C#	3.14
(c) Go	3.23
(i) Dart	3.83
(v) F#	4.13
(i) JavaScript	4.45
(v) Racket	7.91
(i) TypeScript	21.50
(i) Hack	24.02
(i) PHP	29.30
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	69.91
(i) Python	75.88
(i) Perl	79.58

	Time
(c) C	1.00
(c) Rust	1.04
(c) C++	1.56
(c) Ada	1.85
(v) Java	1.89
(c) Chapel	2.14
(c) Go	2.83
(c) Pascal	3.02
(c) Ocaml	3.09
(v) C#	3.14
(v) Lisp	3.40
(c) Haskell	3.55
(c) Swift	4.20
(c) Fortran	4.20
(v) F#	6.30
(i) JavaScript	6.52
(i) Dart	6.67
(v) Racket	11.27
(i) Hack	26.99
(i) PHP	27.64
(v) Erlang	36.71
(i) Jruby	43.44
(i) TypeScript	46.20
(i) Ruby	59.34
(i) Perl	65.79
(i) Python	71.90
(i) Lua	82.91

	Mb
(c) Pascal	1.00
(c) Go	1.05
(c) C	1.17
(c) Fortran	1.24
(c) C++	1.34
(c) Ada	1.47
(c) Rust	1.54
(v) Lisp	1.92
(c) Haskell	2.45
(i) PHP	2.57
(c) Swift	2.71
(i) Python	2.80
(c) Ocaml	2.82
(v) C#	2.85
(i) Hack	3.34
(v) Racket	3.52
(i) Ruby	3.97
(c) Chapel	4.00
(v) F#	4.25
(i) JavaScript	4.59
(i) TypeScript	4.69
(v) Java	6.01
(i) Perl	6.62
(i) Lua	6.72
(v) Erlang	7.20
(i) Dart	8.64
(i) Jruby	19.84

Rust

A language empowering everyone
to build reliable and efficient software.

GET STARTED

Version 1.64.0

Why Rust?

Performance

Rust is blazingly fast and memory-efficient: with no runtime or garbage collector, it can power performance-critical services, run on embedded devices, and easily integrate with other languages.

Reliability

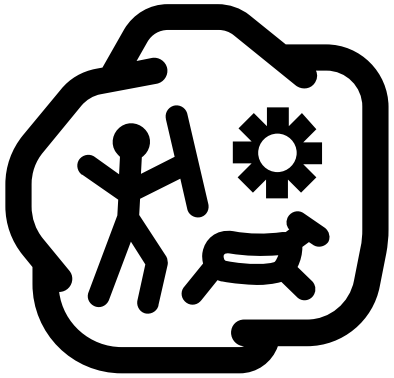
Rust's rich type system and ownership model guarantee memory-safety and thread-safety — enabling you to eliminate many classes of bugs at compile-time.

Productivity

Rust has great documentation, a friendly compiler with useful error messages, and top-notch tooling — an integrated package manager and build tool, smart multi-editor support with auto-completion and type inspections, an auto-formatter, and more.

Architectures Can Evolve

Efficiency == Sustainability



Toilet Finder Service

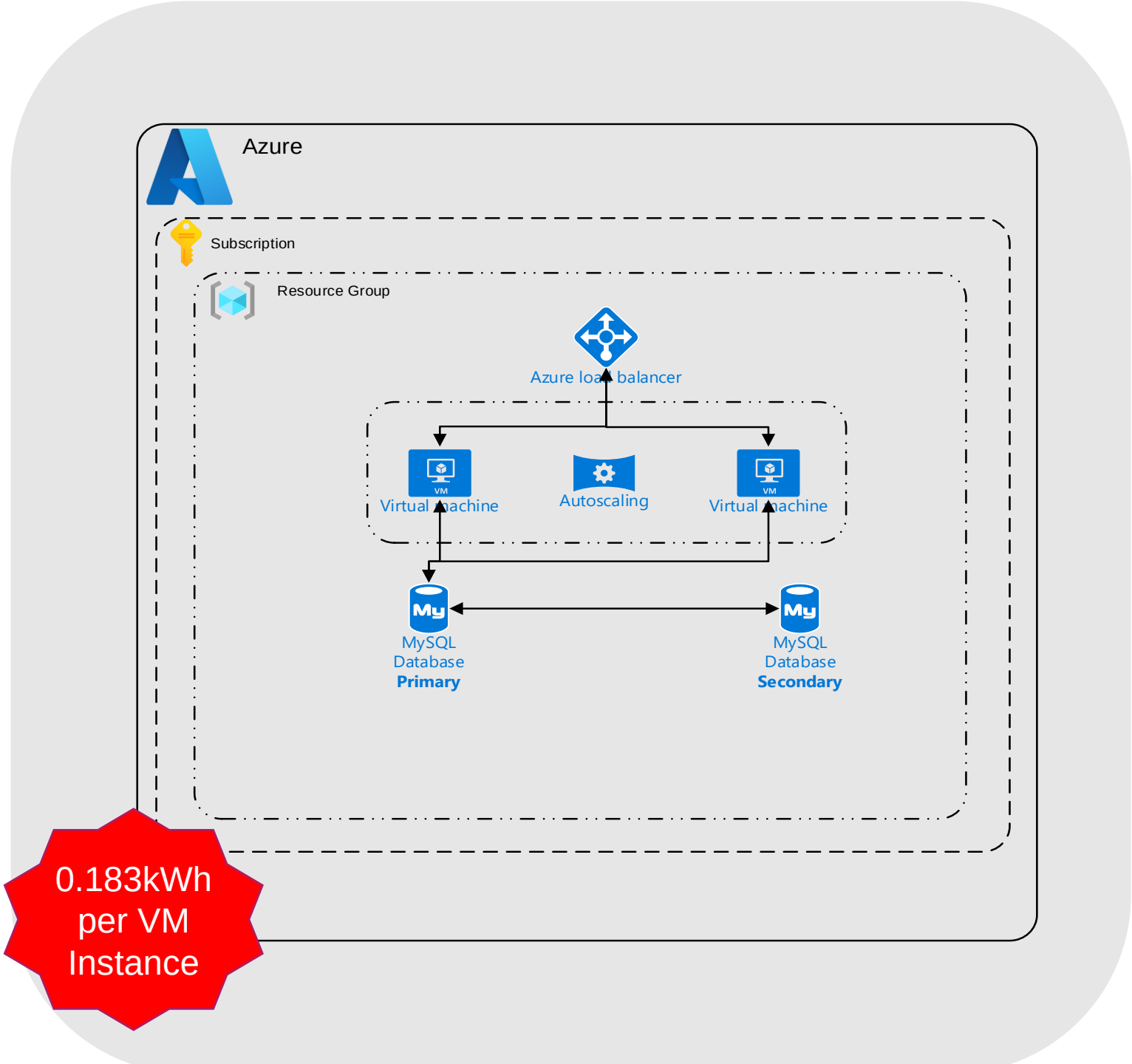
Find me all the toilets in a particular postcode in Australia

Lookup table with 18,408 records

https://data.gov.au/data/dataset/national-public-toilet-map

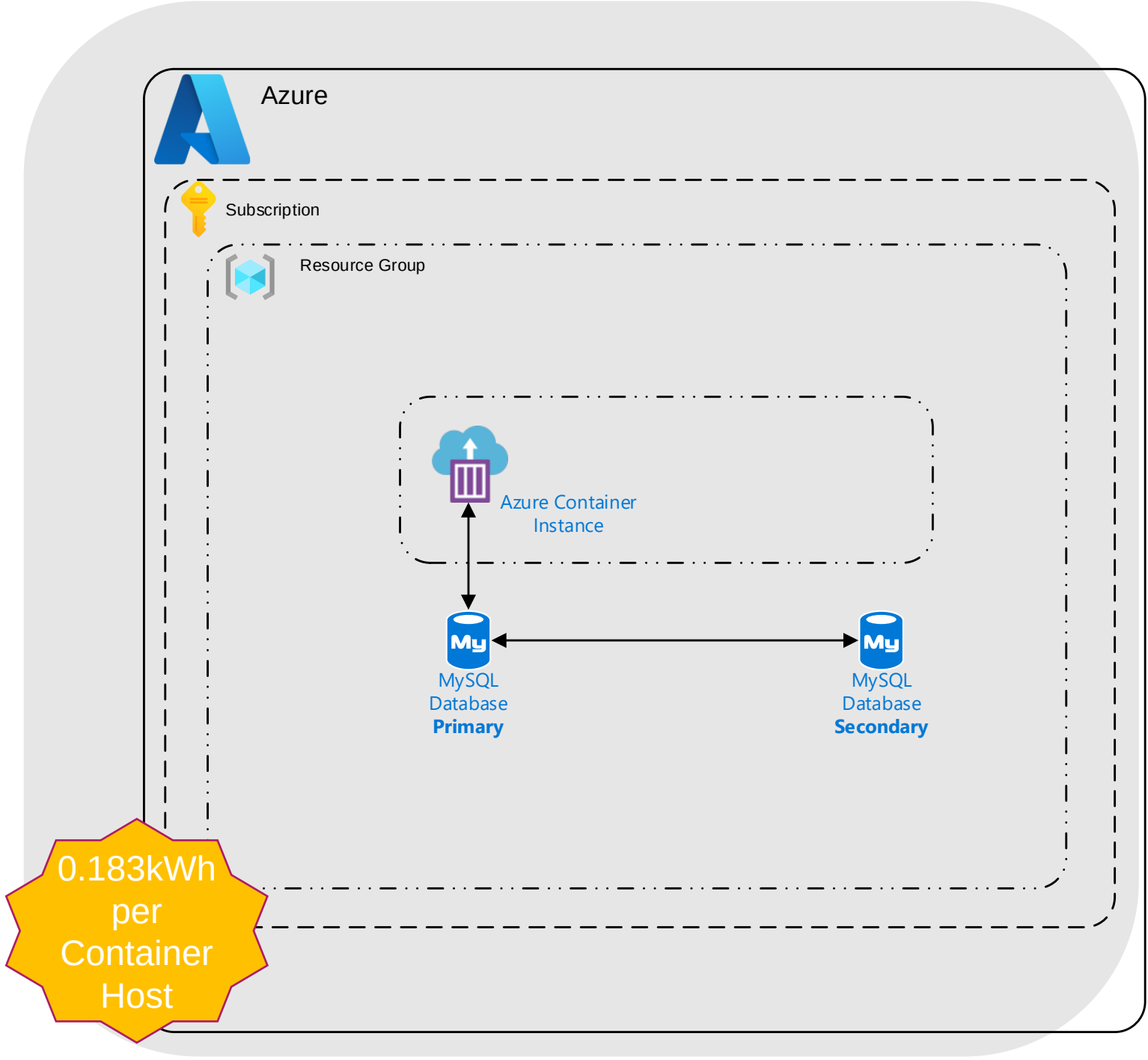
```python
import os
import mysql.connector
import json
import string
from bottle import route, run, template


@route('/')
def index():
    return 'Hello World'


@route('/postcode/<postcode>')
def postcode(postcode):

    returnData = "INVALID input"
    #Validate input - can only be 4 numerics
    if len(postcode) == 4 and postcode.isdigit():

        #MYSQL Connection
        cnx = mysql.connector.connect(user='xxxxxx@toilet-mysql', password='xxxxxxxxx',
                                      host='toilet-mysql.mysql.database.azure.com',
                                      database='toiletdata')
        #Query
        cursor = cnx.cursor()

        query = ("SELECT name, address1, town FROM toilets "
         "WHERE postcode=" + postcode)

        cursor.execute(query)

        resultRow = ""

        for (name, address1, town) in cursor:
            resultRow = resultRow + json.dumps({'name' : name, 'address1': address1, 'town': town}) + ","


        #resultRow = string.rstrip(resultRow, ",")
        resultRow = resultRow.rstrip(",")

        cursor.close()

        cnx.close()
        returnData = "[" + resultRow + "]"

    return returnData

#print __name__
if __name__ == '__main__':
    port = int(os.environ.get('PORT', 8080))
    run(host='0.0.0.0', port=port, debug=True)
```

Typical VM Deployment



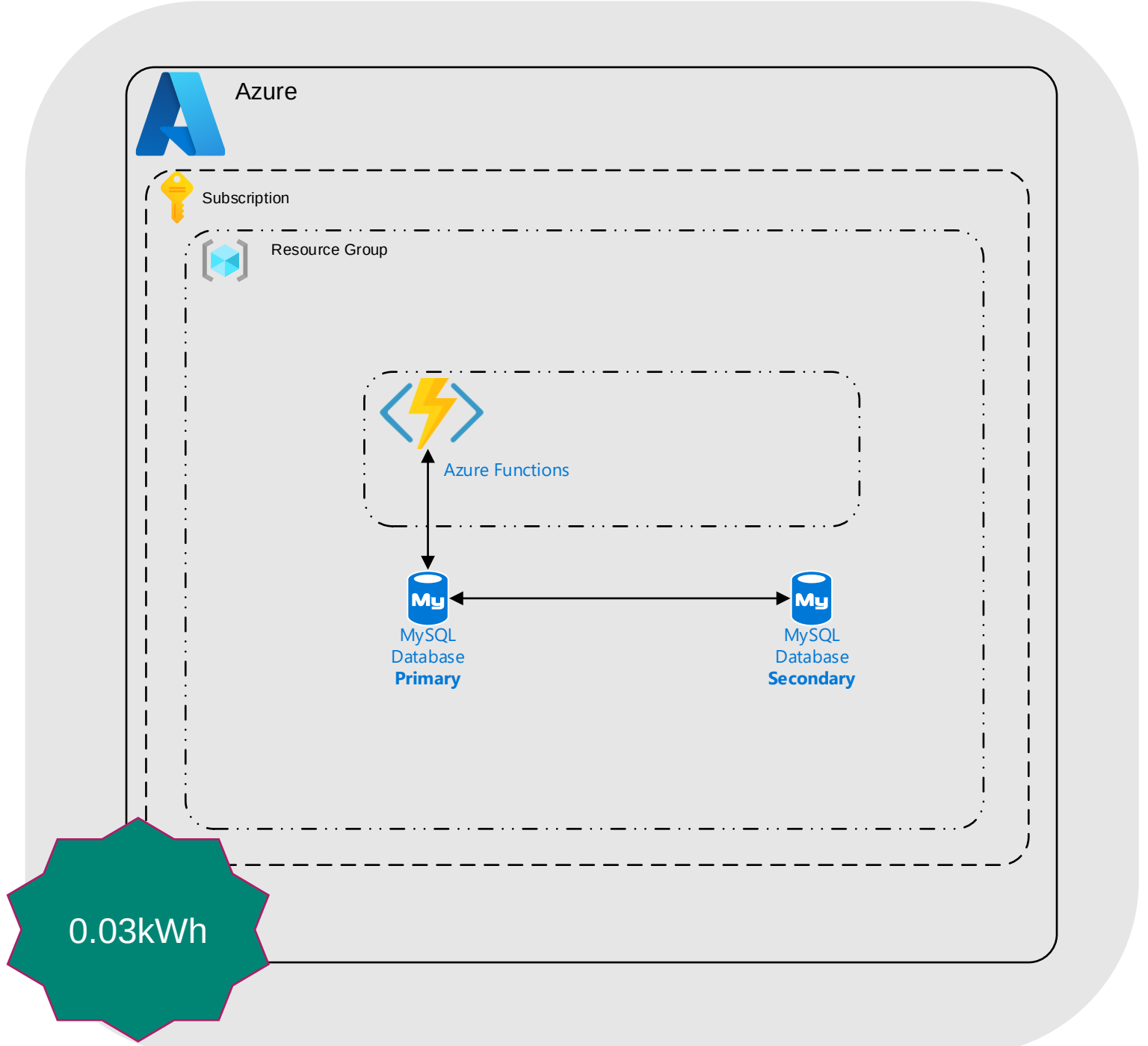
100 000 Executions

Containerise



100 000 Executions

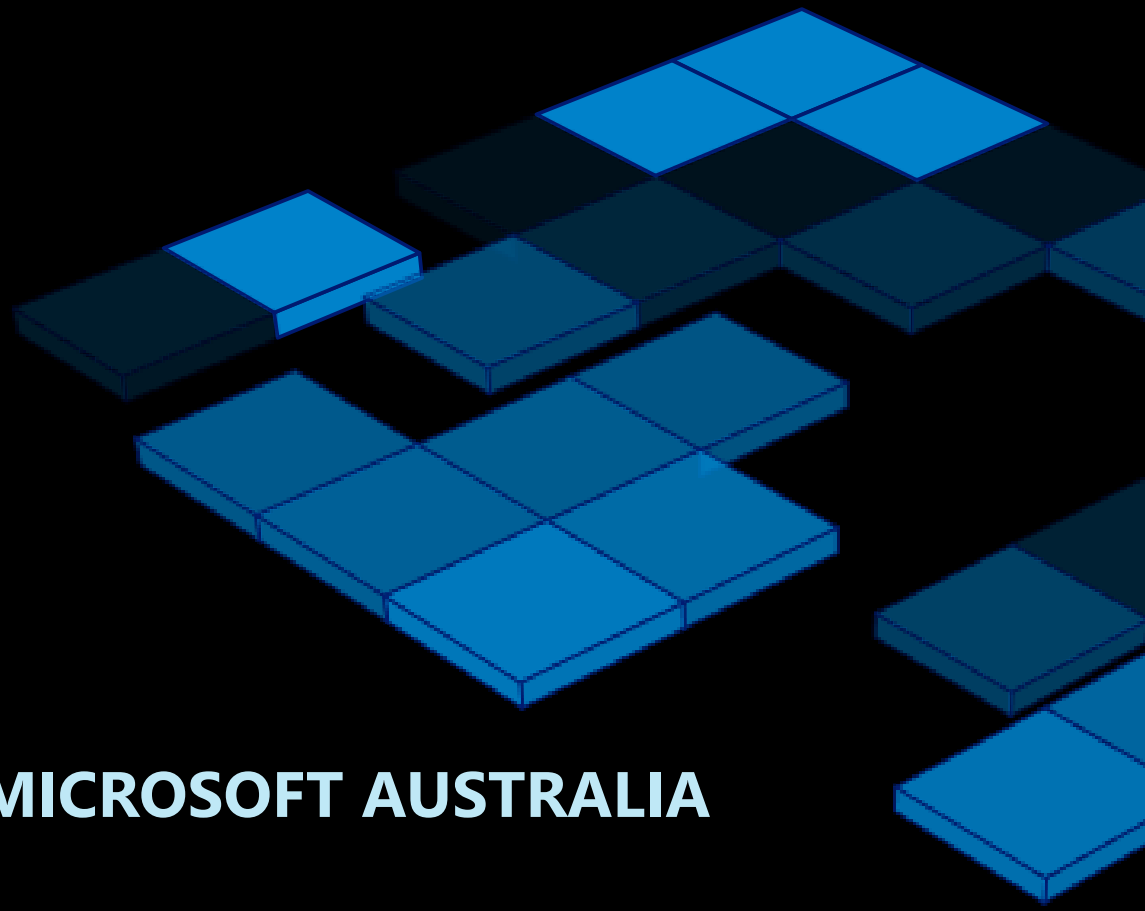
Serverless



100 000 Executions

Time To Show What You Are Made Of

Let’s Be Friends.....



LinkedIn : https://www.linkedin.com/in/shanebaldacchino/
Twitter : sbaldacchino
Web : https://automation.baldacchino.net

SHANE BALDACCHINO | CHIEF ARCHITECT MICROSOFT AUSTRALIA