AWS re:Invent

DECEMBER 2 – 6, 2024 | LAS VEGAS, NV

CMP210

Modernize Apple platform development with AWS and EC2 Mac

Manish Rathaur

Senior Manager, Product Management, Amazon EC2
AWS

Tim Sutton

Senior Mac Mini Rebooter
Block

Amazon EC2 Mac instances

ON-DEMAND APPLE MACOS ENVIRONMENTS FOR THE FIRST TIME ON AWS

POWERED BY APPLE SILICON

Apple chips integrate the CPU, GPU, neural engine, I/O, and so much more onto a single tiny chip

IMPROVED PERFORMANCE

Up to 4x better build performance compared to on premises and up to 60% better price performance on Apple silicon compared to x86 Mac instances

HARNESS THE CLOUD

Provision macOS environments within minutes and only pay for what you use; offload the heavy lifting that comes with managing infrastructure onto AWS

Anyware applitools aspyr Buildkite CIS Center for Internet Security circleci

DeNA Dimps DIGITAL INNOVATOR DNB EastSide Games Electronic Arts FLIPBOARD Goldman Sachs

intuit JAMCITY jamf PAPAYA GAMING The Pokémon Company INTERNATIONAL MPL

NBCUniversal RIOT GAMES ring Star Stable STRABAG twitch Uber Veertu

BLOCK

Agenda

01 iOS Developer Experience at Block

02 2021 crossroads

03 Amazon EC2 Mac – a closer look

04 CI compute architecture

05 Speed bumps along the way

06 Takeaways

07 Q&A

iOS Developer Experience at Block

OUR MISSION

Hands-on support for Square Point of Sale mobile

Continuous Integration (CI) infrastructure

Developer environment: Xcode, CLI tooling, automation

The Bazel build system

iOS and Mac CI infrastructure at large

Cash App, Bitkey, internal Mac apps, 100+ Homebrew bottled eng CLI tools

Bazel Remote Cache and Remote Execution

CI builds mobile and backend services with Bazel…
and developers' laptop builds transparently use cached artifacts from the build

iOS Developer Experience at Block

👏 ❤️ Thanks to everyone who was a part of this journey ❤️ 👏

Jackie Springstead-Chen

Cong Shi

Jerry Marino

Eric Tam

Sunil Venkatraman

Elton Gao

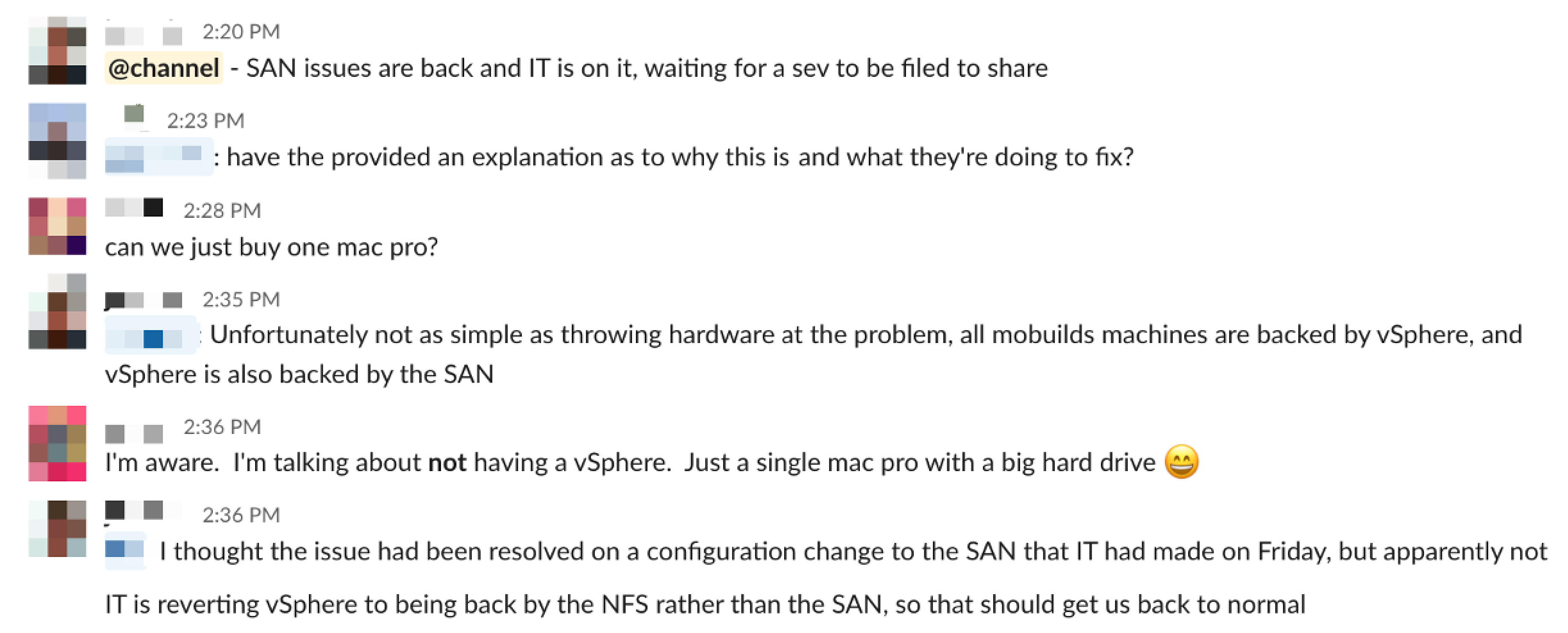
Nick DiZazzo

Bartosz Polaczyk

Justin Martin

Jon Graves

Act I: 2014-2016



[redacted] 2:20 PM
@channel - SAN issues are back and IT is on it, waiting for a sev to be filed to share

[redacted] 2:23 PM
[redacted]: have the provided an explanation as to why this is and what they're doing to fix?

[redacted] 2:28 PM
can we just buy one mac pro?

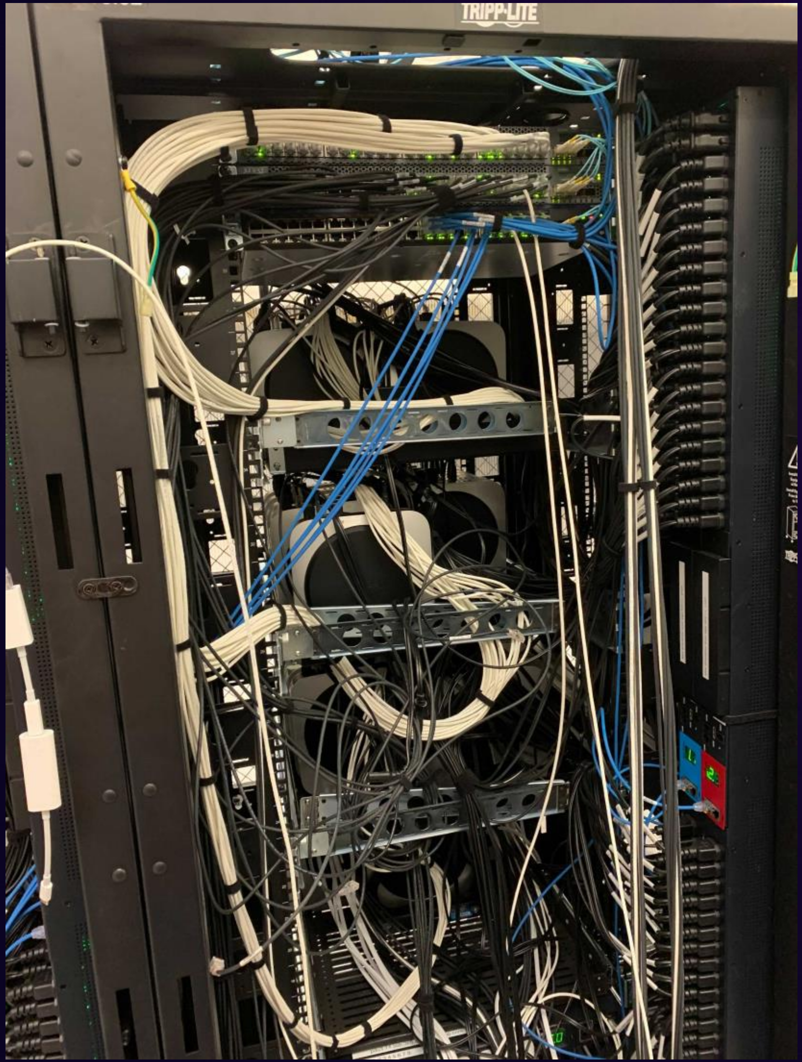
[redacted] 2:35 PM
[redacted]: Unfortunately not as simple as throwing hardware at the problem, all mobuilds machines are backed by vSphere, and vSphere is also backed by the SAN

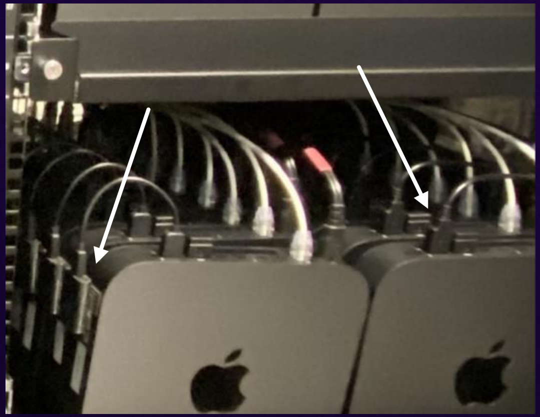
[redacted] 2:36 PM
I'm aware. I'm talking about **not** having a vSphere. Just a single mac pro with a big hard drive 😄

[redacted] 2:36 PM
[redacted] I thought the issue had been resolved on a configuration change to the SAN that IT had made on Friday, but apparently not

IT is reverting vSphere to being back by the NFS rather than the SAN, so that should get us back to normal

Act II: 2017-2020





Act III: 2021-2024

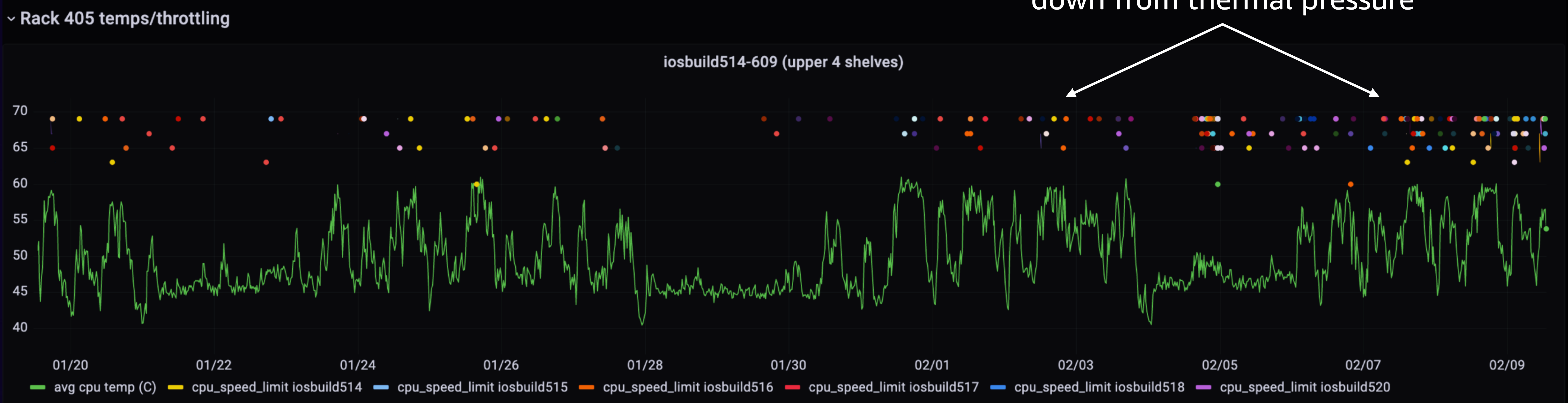
Square and Cash App, by the numbers

- 2 monolithic Git repos, 7M+ lines of Swift and Objective-C
- ~300 contributing iOS developers
- 450 CI machines, 200 remote test execution cluster

- Square-specific
  - 28 hours machine time to run *all* test suites
  - 300 Pull Request builds per day
  - One build fans out to 4-50 machines depending on complexity of code change
  - 27 GB (compressed) Git repo
  - 140,000 targets in the build graph

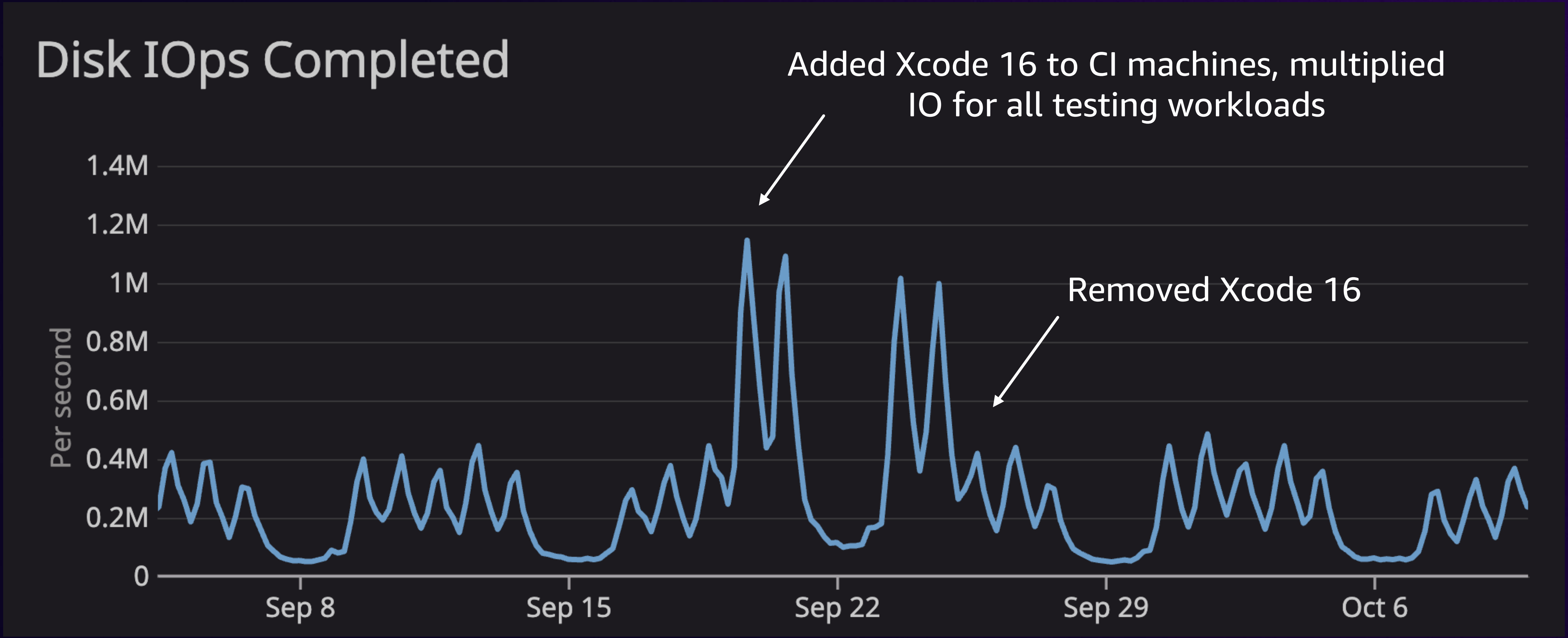
iOS builds-at-scale problems: CPU and thermals

Configured tests to use newer iOS version, CPU clock speeds throttling down from thermal pressure



Rack 405 temps/throttling

iOS builds-at-scale problems: Disk IOPS

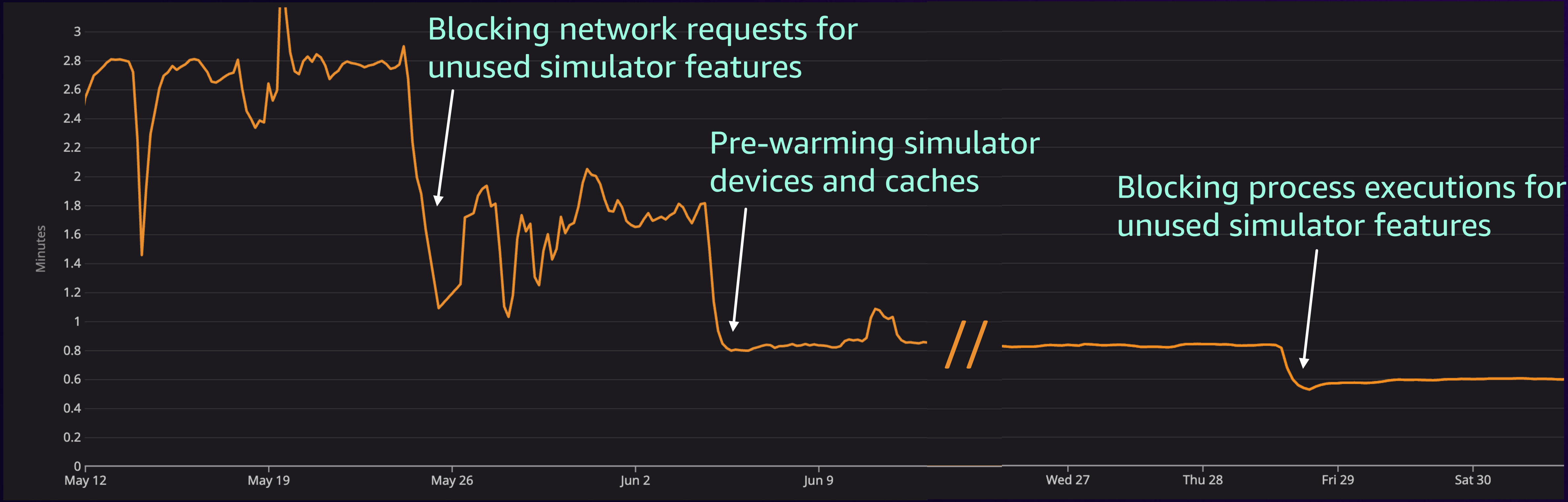


Added Xcode 16 to CI machines, multiplied IO for all testing workloads

Removed Xcode 16

iOS builds-at-scale problems: Caches . . . of everything

iOS Simulator boot performance tuning, May – Nov 2024
From 170 seconds down to 36 seconds



Blocking network requests for unused simulator features

Pre-warming simulator devices and caches

Blocking process executions for unused simulator features

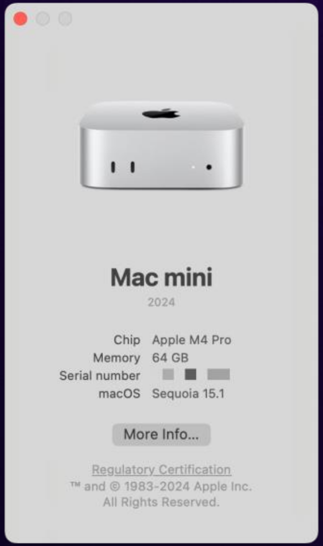
2021 crossroads: Where are we going?

Pain points

PART-TIME INFRASTRUCTURE MANAGEMENT

Toil

Risk



Mac mini

2024

Chip Apple M4 Pro

Memory 64 GB

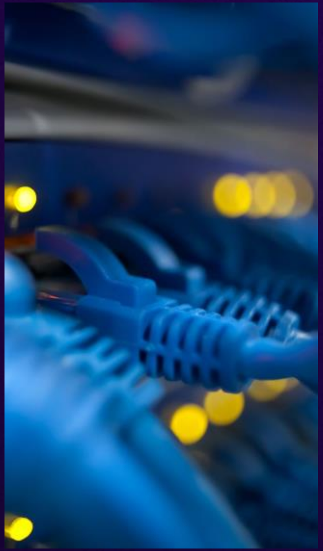
Serial number [redacted]

macOS Sequoia 15.1

Regulatory Certification

™ and © 1983-2024 Apple Inc.
All Rights Reserved.

Inflexibility



Integration antipatterns

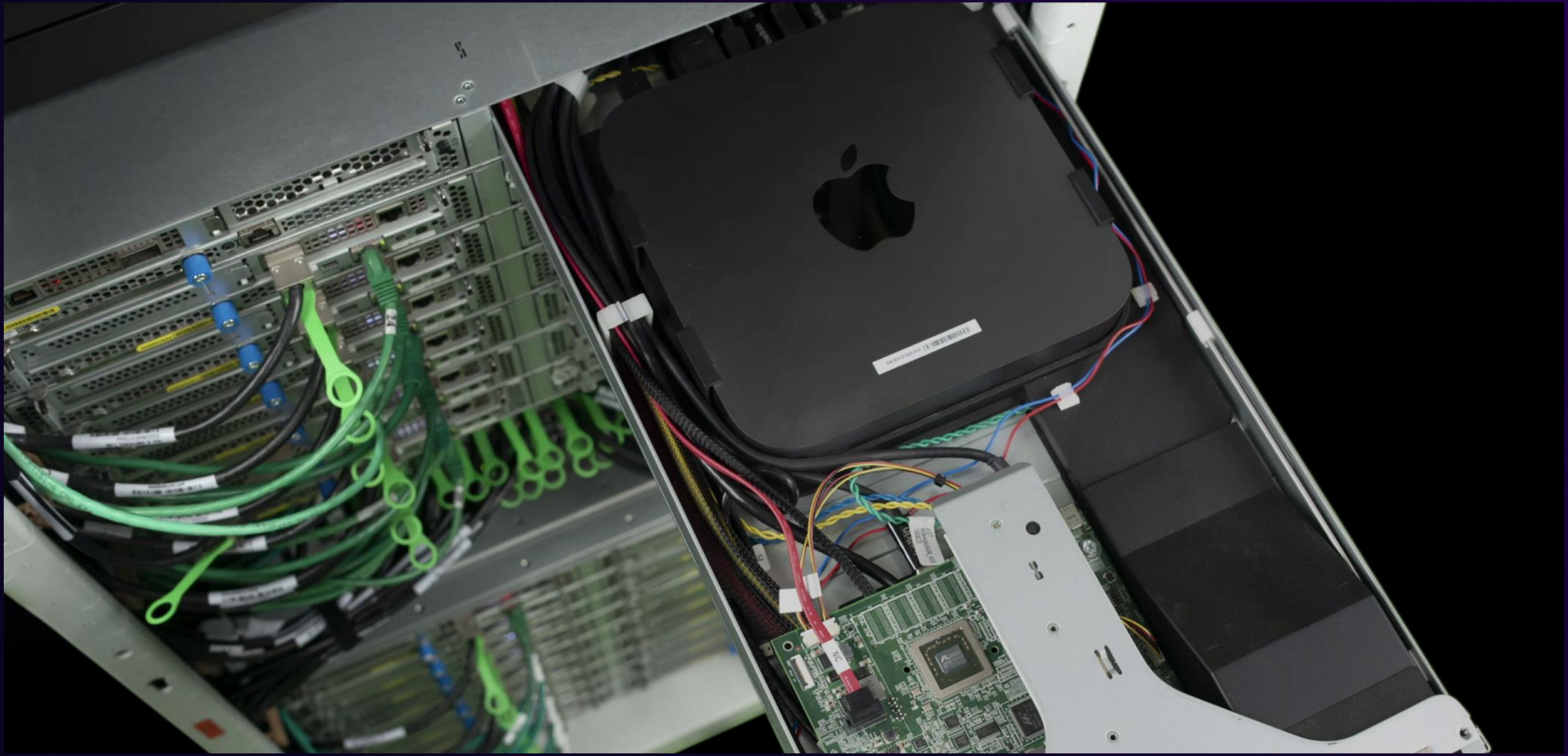
Enter Amazon EC2 Mac

WHAT COULD THIS ENABLE?

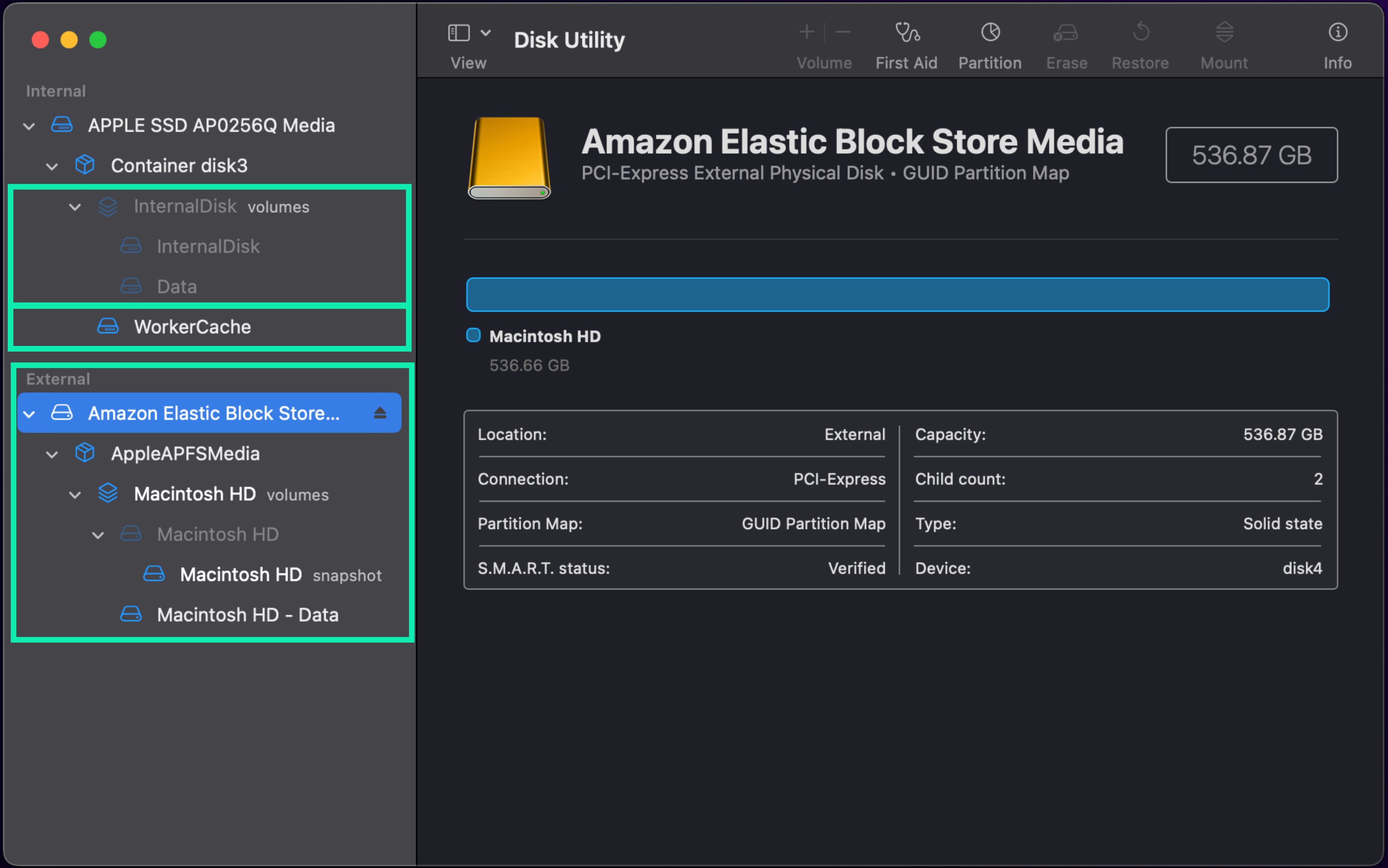
- Bare-metal bootable snapshots
- Eliminate static machine inventory
  - No more “cobwebs” from long-lived installations and hardware
  - Spin up additional capacity and configurations ad hoc without “taking away” from our main production capacity
- Change our hardware configuration or adopt new models as needed
- Integration with other AWS services

Amazon EC2 Mac – a closer look

How Amazon EC2 Mac works



https://github.com/aws-samples/amazon-ec2-mac-getting-started



Amazon Elastic Block Store Media

PCI-Express External Physical Disk • GUID Partition Map

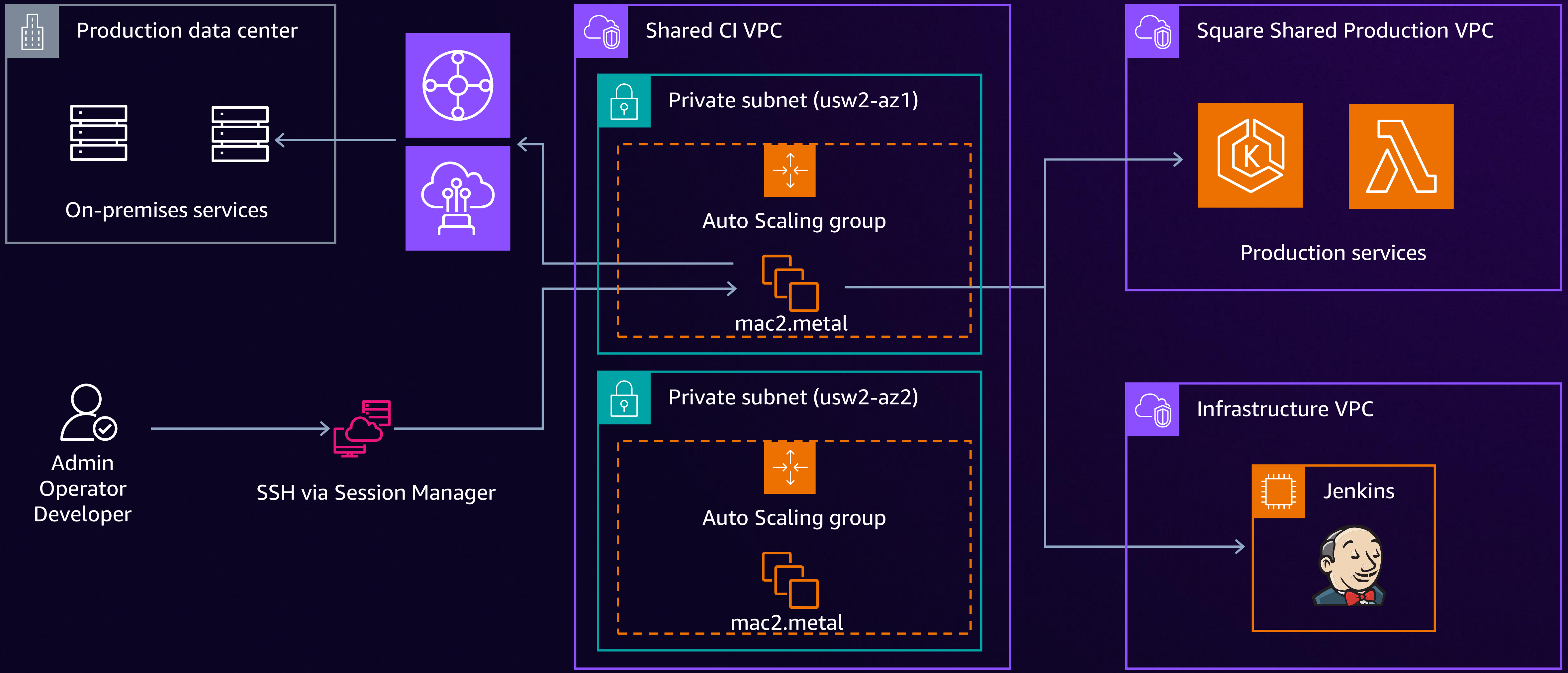
536.87 GB

Macintosh HD
536.66 GB

Location:	External	Capacity:	536.87 GB
Connection:	PCI-Express	Child count:	2
Partition Map:	GUID Partition Map	Type:	Solid state
S.M.A.R.T. status:	Verified	Device:	disk4

CI architecture overview (what we built)

Mac CI machines



Auto Scaling groups as a building block

- **Xcode:** Which versions, iOS simulator device combinations
- **macOS:** Sonoma, Sequoia, future betas
- **Operator environment:** development, staging, or production
- **Architecture:** Apple Silicon or (legacy) Intel
- **Instance types:** mac2, mac2-m2pro
- **Experiments:** OS/user config, testing Apple betas

Auto Scaling groups as a building block

```
sonoma-stable = {
  # 2024-10-16 (macos14.6.1-apple-silicon)

  ami_id                = "ami-02bb786c100a56c5c"
  instance_type         = "mac2.metal"
  jenkins_url           = "https://ci-prod.block.xyz"
  worker_labels         = ["ec2-baremetal"]
  min_num_instances     = 375
  desired_num_instances = 450
  max_num_instances     = 450
  ebs_volume_size       = 750
}
```

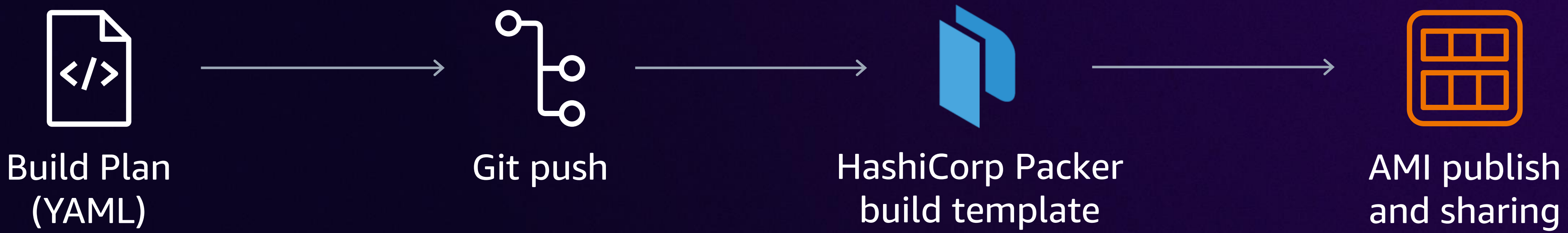
Auto Scaling groups as a building block

```
xcode-16-1 = {
  # 2024-11-04 (Xcode 16.1 + CoreSimulator beta – candidate new prod image)

  ami_id                = "ami-0e13b34ab2ca62738"
  instance_type         = "mac2-m2.metal"
  jenkins_url           = "https://ci-stage.block.xyz"
  worker_labels         = ["ec2-baremetal-staging", "mdx-10384"]
  min_num_instances     = 0
  desired_num_instances = 5
  max_num_instances     = 5
  ebs_volume_size       = 500
  instance_additional_tags = {
    "FeatureWarmupHostDisk" = "true"
    ”AnsibleBranch"         = "tsutton/mdx-10384/test-fix"
  }
}
```

Building AMIs with Packer

ALWAYS BUILDABLE, ALWAYS DEPLOYABLE

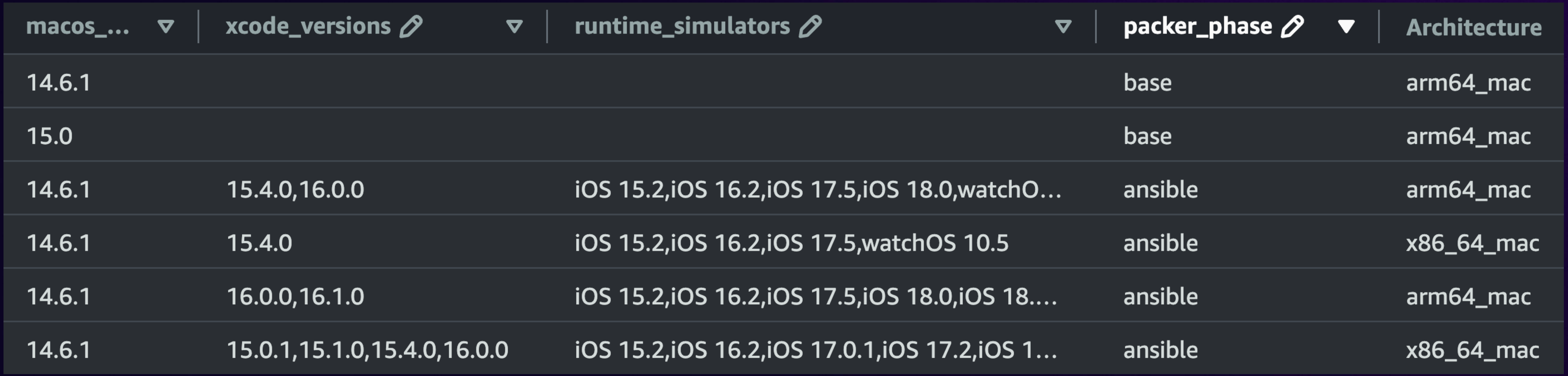


Build Plan (YAML)

Git push

HashiCorp Packer build template

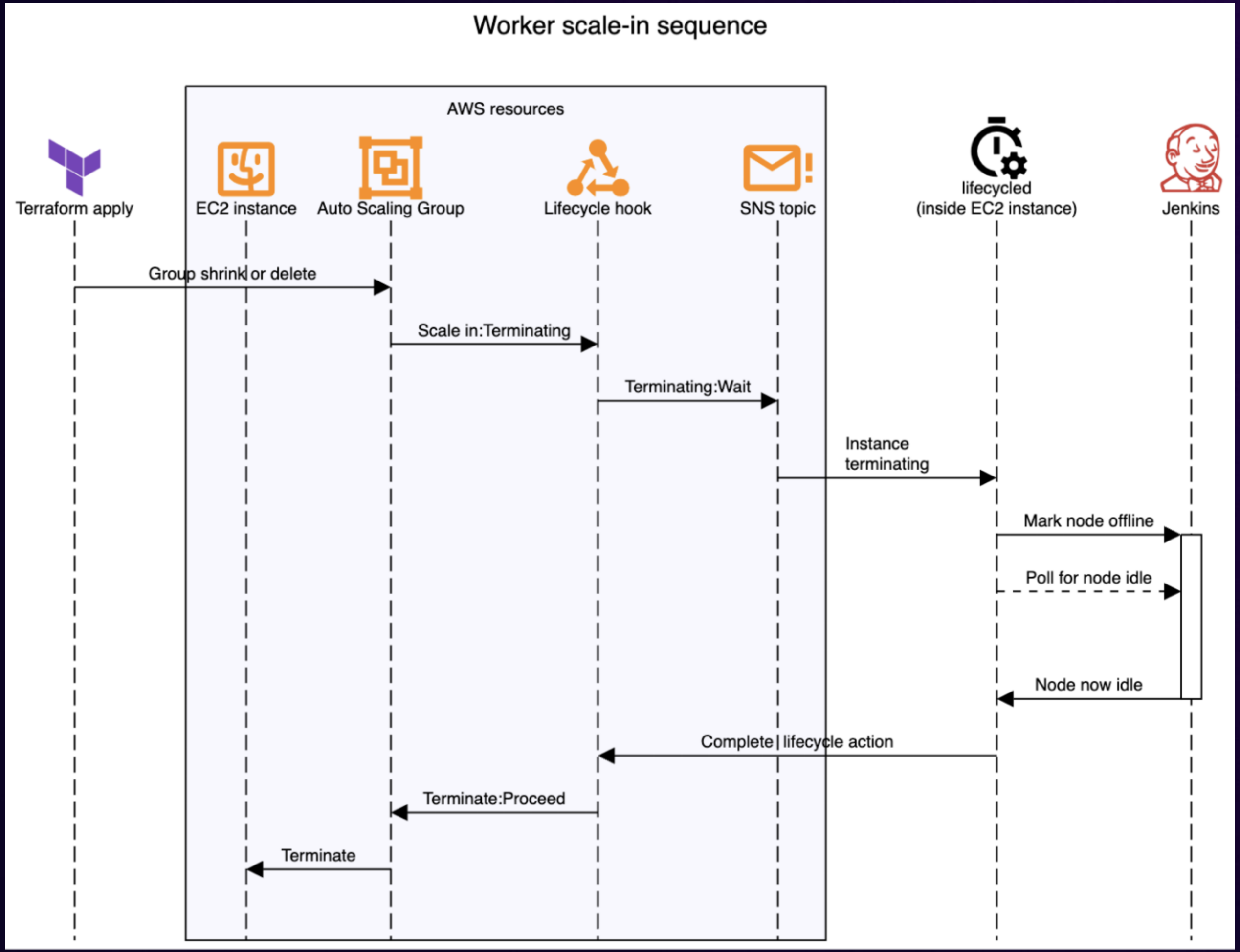
AMI publish and sharing



macos_…	xcode_versions	runtime_simulators	packer_phase	Architecture
14.6.1			base	arm64_mac
15.0			base	arm64_mac
14.6.1	15.4.0,16.0.0	iOS 15.2,iOS 16.2,iOS 17.5,iOS 18.0,watchO…	ansible	arm64_mac
14.6.1	15.4.0	iOS 15.2,iOS 16.2,iOS 17.5,watchOS 10.5	ansible	x86_64_mac
14.6.1	16.0.0,16.1.0	iOS 15.2,iOS 16.2,iOS 17.5,iOS 18.0,iOS 18.…	ansible	arm64_mac
14.6.1	15.0.1,15.1.0,15.4.0,16.0.0	iOS 15.2,iOS 16.2,iOS 17.0.1,iOS 17.2,iOS 1…	ansible	x86_64_mac

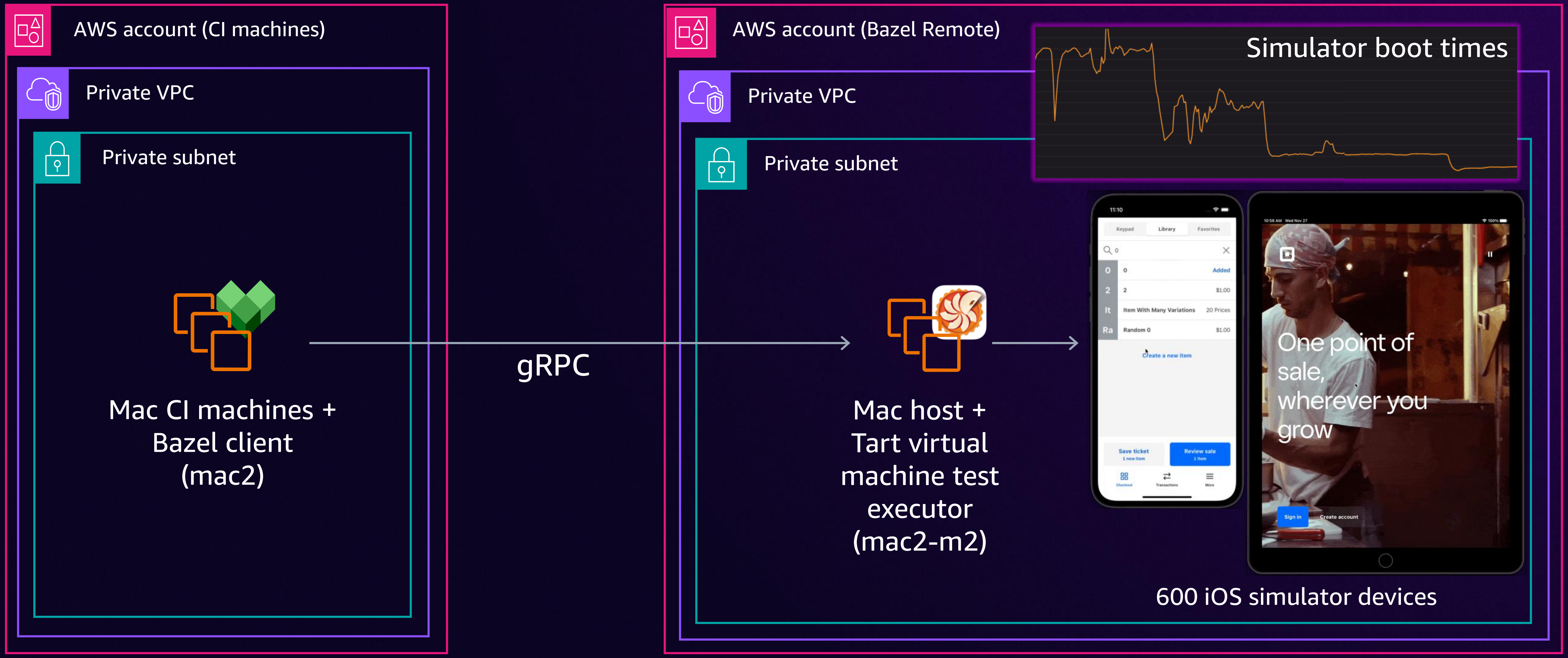
Worker lifecycle with ASG and Lifecycle Hooks

ROLL OUT INSTANCE CHANGES ANY TIME OF DAY



Bazel remote build (and test) execution

BUILDFARM, AMAZON ECS, AND APPLE SILICON VIRTUAL MACHINES



AWS account (CI machines)

Private VPC

Private subnet

Mac CI machines + Bazel client (mac2)

gRPC

AWS account (Bazel Remote)

Private VPC

Private subnet

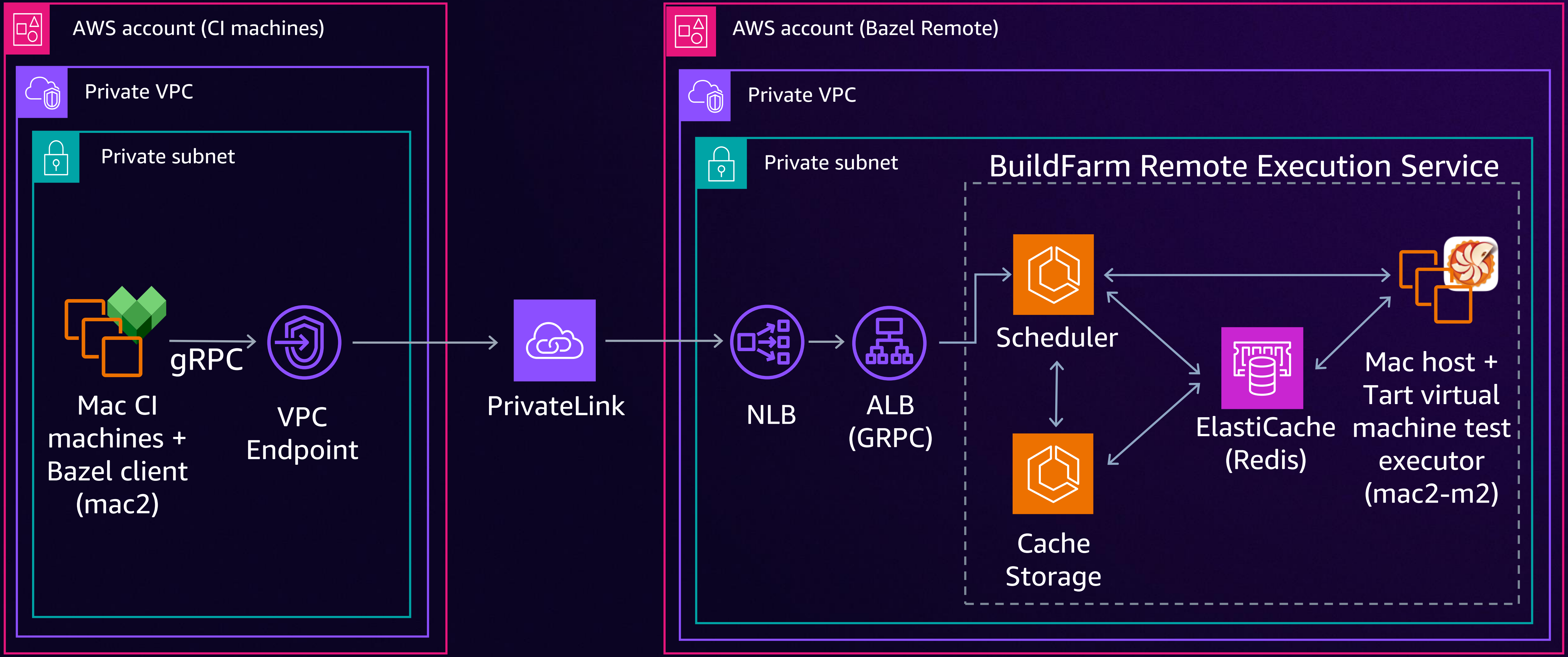
Mac host + Tart virtual machine test executor (mac2-m2)

Simulator boot times

600 iOS simulator devices

Bazel remote build (and test) execution

BUILDFARM, AWS ECS AND APPLE SILICON VIRTUAL MACHINES


AWS account (CI machines)
Private VPC
Private subnet
Mac CI machines + Bazel client (mac2)
gRPC
VPC Endpoint
PrivateLink
AWS account (Bazel Remote)
Private VPC
Private subnet
BuildFarm Remote Execution Service
NLB
ALB (GRPC)
Scheduler
Cache Storage
ElastiCache (Redis)
Mac host + Tart virtual machine test executor (mac2-m2)

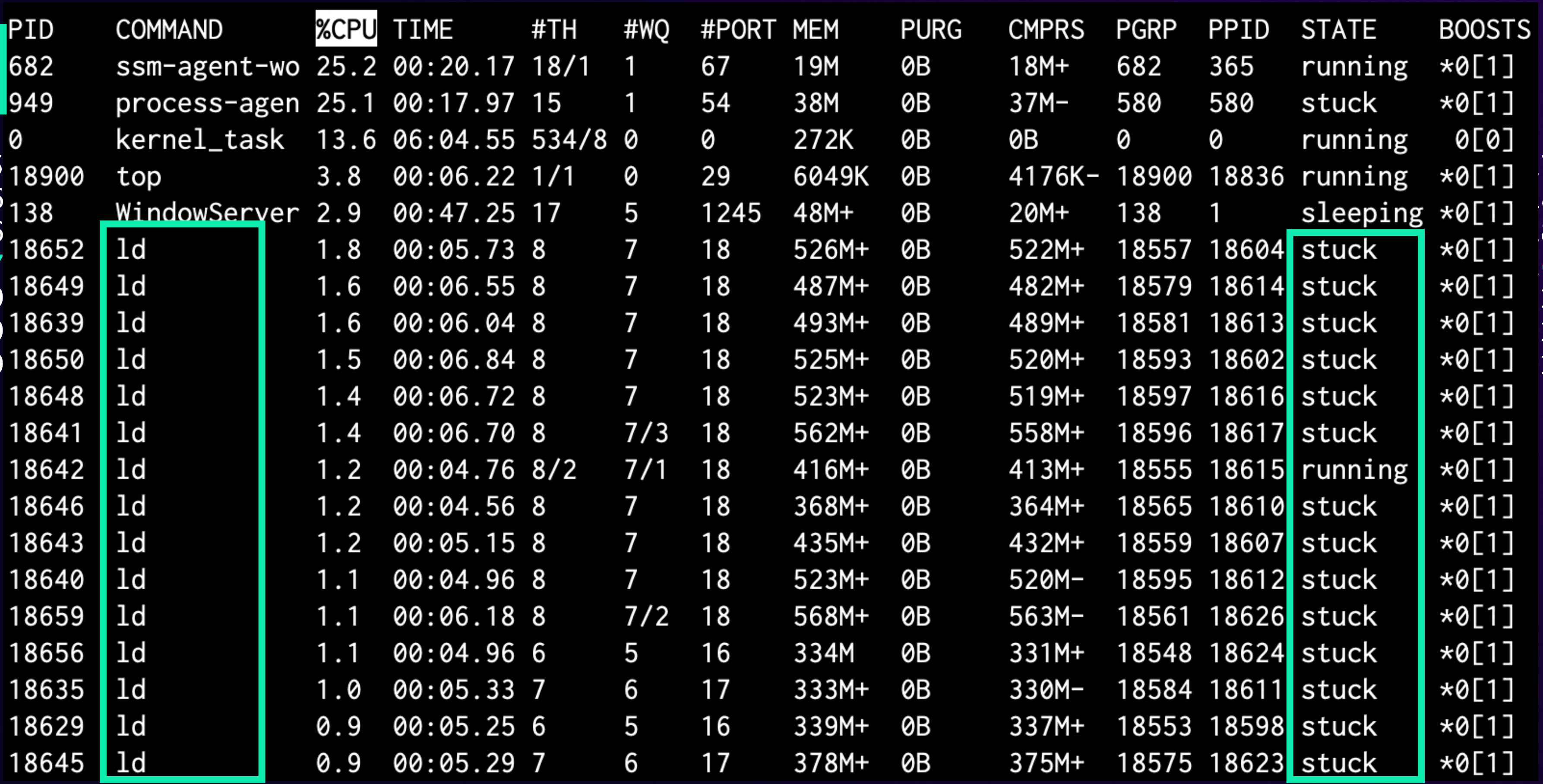
Speed bumps along the way

Understanding Amazon EBS

TIME GAPS AND STUCK PROCESSES

```
[2023-07-24T15:13
[2023-07-24T18:23

[2023-10-18T23:55
[2023-10-18T23:56
[2023-10-18T23:56
[2023-10-18T23:57
[2023-10-18T23:59
[2023-10-18T23:59
[2023-10-18T23:59
```



```
PID    COMMAND      %CPU TIME     #TH    #WQ  #PORT MEM    PURG   CMPRS  PGRP  PPID  STATE    BOOSTS
682    ssm-agent-wo 25.2 00:20.17 18/1   1    67    19M    0B     18M+   682   365   running  *0[1]
949    process-agen 25.1 00:17.97 15     1    54    38M    0B     37M-   580   580   stuck    *0[1]
0      kernel_task  13.6 06:04.55 534/8  0    0     272K   0B     0B     0     0     running   0[0]
18900  top          3.8  00:06.22 1/1    0    29    6049K  0B     4176K- 18900 18836 running  *0[1]
138    WindowServer 2.9  00:47.25 17     5    1245  48M+   0B     20M+   138   1     sleeping *0[1]
18652  ld           1.8  00:05.73 8      7    18    526M+  0B     522M+  18557 18604 stuck    *0[1]
18649  ld           1.6  00:06.55 8      7    18    487M+  0B     482M+  18579 18614 stuck    *0[1]
18639  ld           1.6  00:06.04 8      7    18    493M+  0B     489M+  18581 18613 stuck    *0[1]
18650  ld           1.5  00:06.84 8      7    18    525M+  0B     520M+  18593 18602 stuck    *0[1]
18648  ld           1.4  00:06.72 8      7    18    523M+  0B     519M+  18597 18616 stuck    *0[1]
18641  ld           1.4  00:06.70 8      7/3  18    562M+  0B     558M+  18596 18617 stuck    *0[1]
18642  ld           1.2  00:04.76 8/2    7/1  18    416M+  0B     413M+  18555 18615 running  *0[1]
18646  ld           1.2  00:04.56 8      7    18    368M+  0B     364M+  18565 18610 stuck    *0[1]
18643  ld           1.2  00:05.15 8      7    18    435M+  0B     432M+  18559 18607 stuck    *0[1]
18640  ld           1.1  00:04.96 8      7    18    523M+  0B     520M-  18595 18612 stuck    *0[1]
18659  ld           1.1  00:06.18 8      7/2  18    568M+  0B     563M-  18561 18626 stuck    *0[1]
18656  ld           1.1  00:04.96 6      5    16    334M   0B     331M+  18548 18624 stuck    *0[1]
18635  ld           1.0  00:05.33 7      6    17    333M+  0B     330M-  18584 18611 stuck    *0[1]
18629  ld           0.9  00:05.25 6      5    16    339M+  0B     337M+  18553 18598 stuck    *0[1]
18645  ld           0.9  00:05.29 7      6    17    378M+  0B     375M+  18575 18623 stuck    *0[1]
```

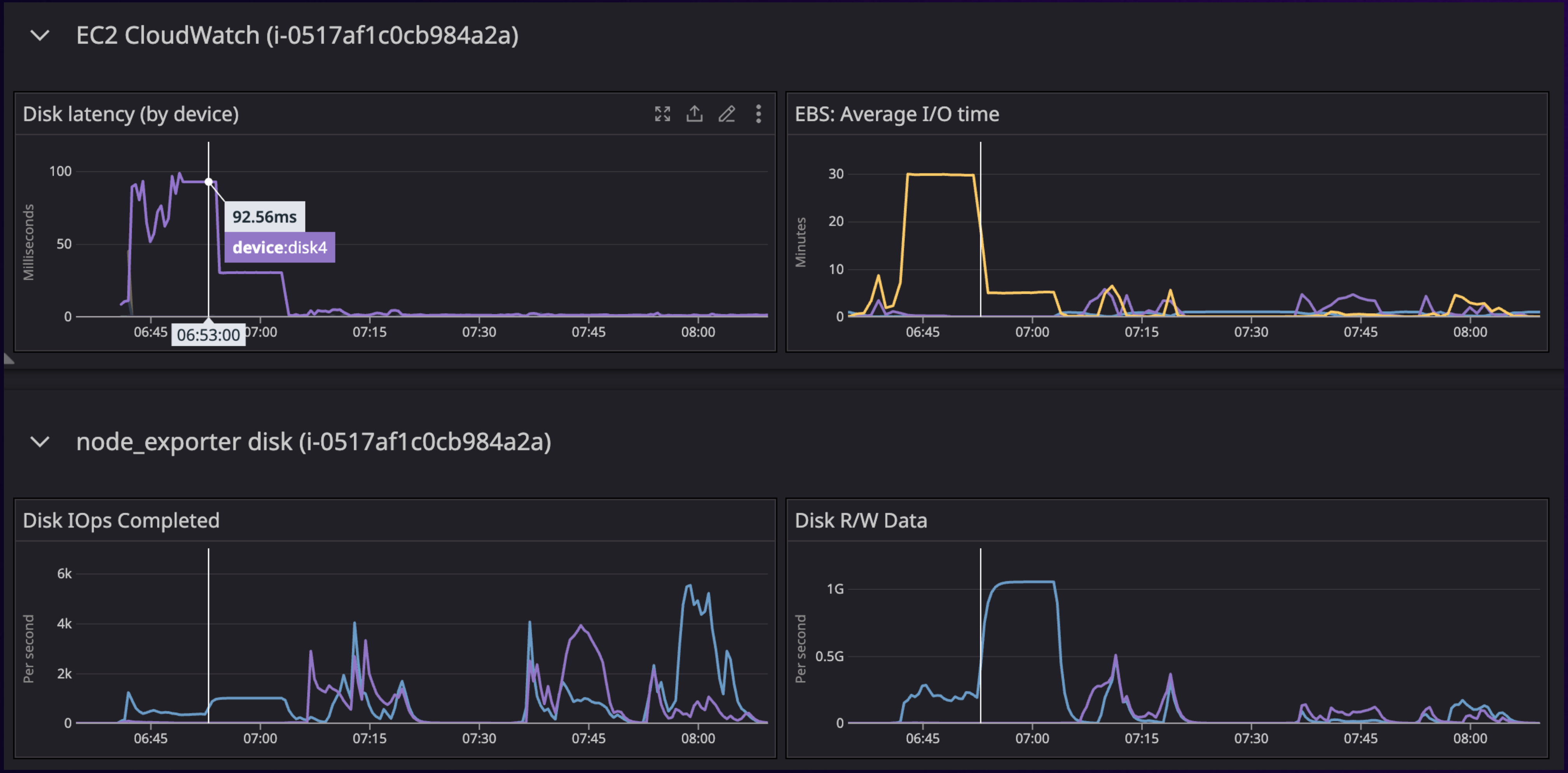
```
7 running)
8 running)
8 running)
ception
18 running)
16 running)
16 running)
```

Understanding Amazon EBS

BLOCKS ARE LAZY-LOADED AS THEY ARE ACCESSED

- “When you create an EBS volume from an EBS snapshot, data from the EBS snapshot is lazy loaded into an EBS volume. If the volume is accessed where the data is not loaded, the application accessing the volume encounters a higher latency than normal while the data gets loaded. This higher latency due to lazy loading could lead to a poor user experience for latency-sensitive workloads.”
  - https://aws.amazon.com/blogs/storage/addressing-i-o-latency-when-restoring-amazon-ebs-volumes-from-ebs-snapshots/
- macOS always expects a low-latency NVMe device, and the boot volume commands a lot of I/O (especially at startup)

Understanding Amazon EBS



Understanding Amazon EBS

RECOMMENDATIONS

- Pre-initialize any EBS volumes backed by snapshots, e.g.,
  - `sudo fio --filename=/dev/rdisk4 --rw=read --bs=1M`
- Test real workloads, not just benchmarks
- Avoid swapping if possible
- New, empty EBS volumes don’t pay this first block read penalty – so consider using those (or the internal SSD) for ephemeral build data
  - “AWS does not manage or support the internal SSD on the Apple hardware”
    - From https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-mac-instances.html
- Find your ideal cost-to-performance IOPS/Mbps settings for a gp3 / io2 volume from real workloads

Network interfaces cached in AMIs

PREVENT ENA INTERFACE CACHING IN MACOS

- Instances launched from our “layered” AMIs would fail to come up on the network
- Difficult for us to debug without an actual screen/keyboard attached
- Do forensics! Shut down the problematic instance, snapshot and attach its volume to another instance, and look at logs
- Looking at `/var/log/amazon` for ENI driver and `ec2-macos-init` logs can give insight

- The ENA interface ID is not deterministic (it’s external, hot-pluggable), so clear NIC caches before capturing a new AMI:
  - `sudo rm -f /Library/Preferences/SystemConfiguration/NetworkInterfaces.plist`
- `StopInstances` on Mac is a physical poweroff, *not* an ACPI shutdown signal, so perform our own OS shutdown, then *wait a short time*, before finally stopping and snapshotting:
  - `sudo shutdown -h now`

Potentially limited host capacity

- Provisioned subnets in only 2 out of 4 Availability Zones in us-west-2
- mac2.metal capacity was scarce when we were needing to scale up
- Expanded subnets from 2 to 4 Availability Zones
- Our forecasting for scale-up timing wasn’t very clear

- Recommendation for larger deployments:
  - Use all Availability Zones possible for a Region, multi-Region if possible
  - Work with your reps if you are planning a big migration! They can arrange for capacity to be delivered to specific AZs for a given date (with lead time)
    - Not all AZs have all Mac instance types; reps can give you this info to save you time

Learnings and takeaways

Retrospective

WHAT WE DID RIGHT

- Kept it as simple as possible by changing as little as possible
- Dev-to-staging-to-prod workflow
- Automated AMI creation
- Planned which workloads to migrate and when
- Monitoring for machines, metrics for build/test performance

Know thyself

KNOW YOUR EXPERTISE, KNOW YOUR PRIORITIES

- We are looking to build on top of an infrastructure platform, not to outsource the build and test environments
- Higher-level Mac CI/test vendor solutions won’t allow us to optimize for what we need
- Knowing our mission:
  - Deliver value with novel solutions to our org’s build and test needs
  - Focus not on *managing* physical infrastructure, but on a *deep understanding* of it
  - Continue building out Mac CI and Bazel remote execution infrastructure that’s tailored for the unique challenges of large iOS codebases

Thank you!

Please complete the session survey in the mobile app

Tim Sutton

github.com/timsutton
tvsutton@mastodon.social
https://macops.ca

Manish Rathaur

mrathaur@amazon.com