
AWS
re:Invent
DECEMBER 2 – 6, 2024 | LAS VEGAS, NV

PRO303

Scaling to new heights: Coinbase migrates to Amazon EKS and scales 10x

Tiberiu Oprisiu
Senior Staff Software Engineer
Coinbase

Dr. Adam Link
Engineering Manager
Coinbase

Rob Martell
Principal Cloud Application Architect
AWS

Your speakers for the session

Tiberiu Oprisiu

Senior Staff Software Engineer
Coinbase

Dr. Adam Link

Engineering Manager
Coinbase

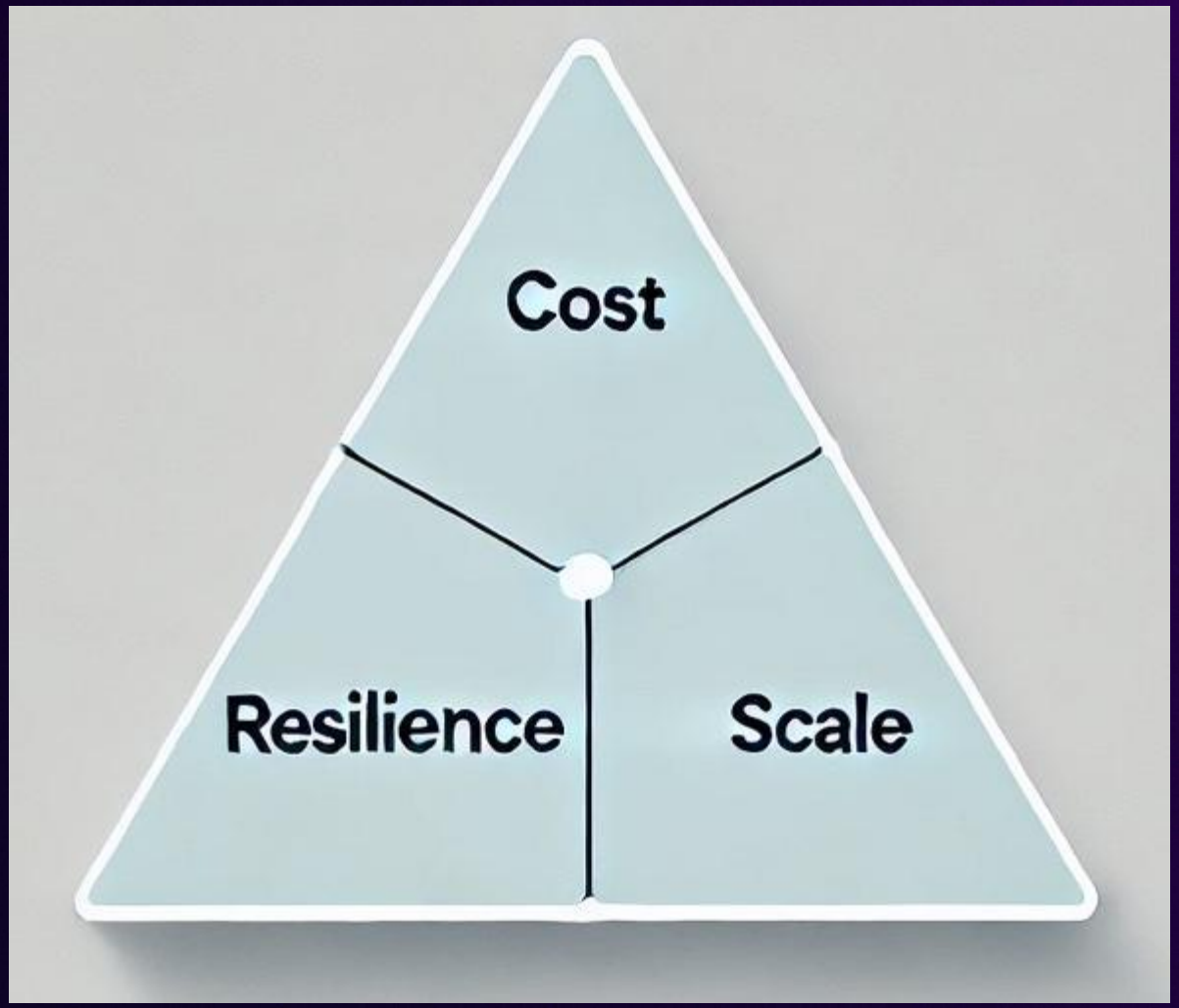
Rob Martell

Principal Cloud Application Architect
AWS

10x scale program

The program objectives – Coinbase and AWS Professional Services

- Reduce costs of compute footprint
- Improve scaling for unpredictable loads
- Modernize compute designs

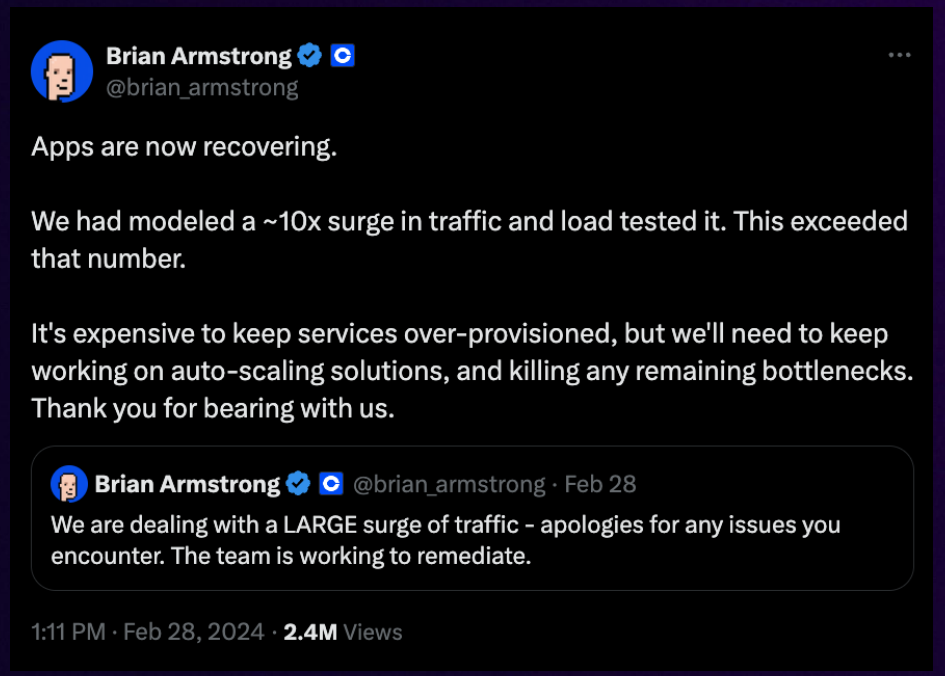


10x roadblocks

- How to make Amazon EC2 clusters more reliable
- How to react to extremely high scaling rate (tps)
- When to drain for scale-down
- When to use static stable Pods in Amazon EKS
- EKS design restructures and what to move to EKS
- Are there better tools for scaling?

Coinbase

- Founded May 2012
- First and only publicly traded, audited crypto exchange
- Crypto market volatility causes sudden traffic influxes
- Ensure scalability and reliability during bull cycles
- Cost optimizations during bear cycles



Coinbase 2022

- Coming out of a strong bull cycle with large crypto adoption
- The whole industry did a “reset” to adapt to the new normal
- Coinbase had to reset system usage and costs

Bitcoin price

Time

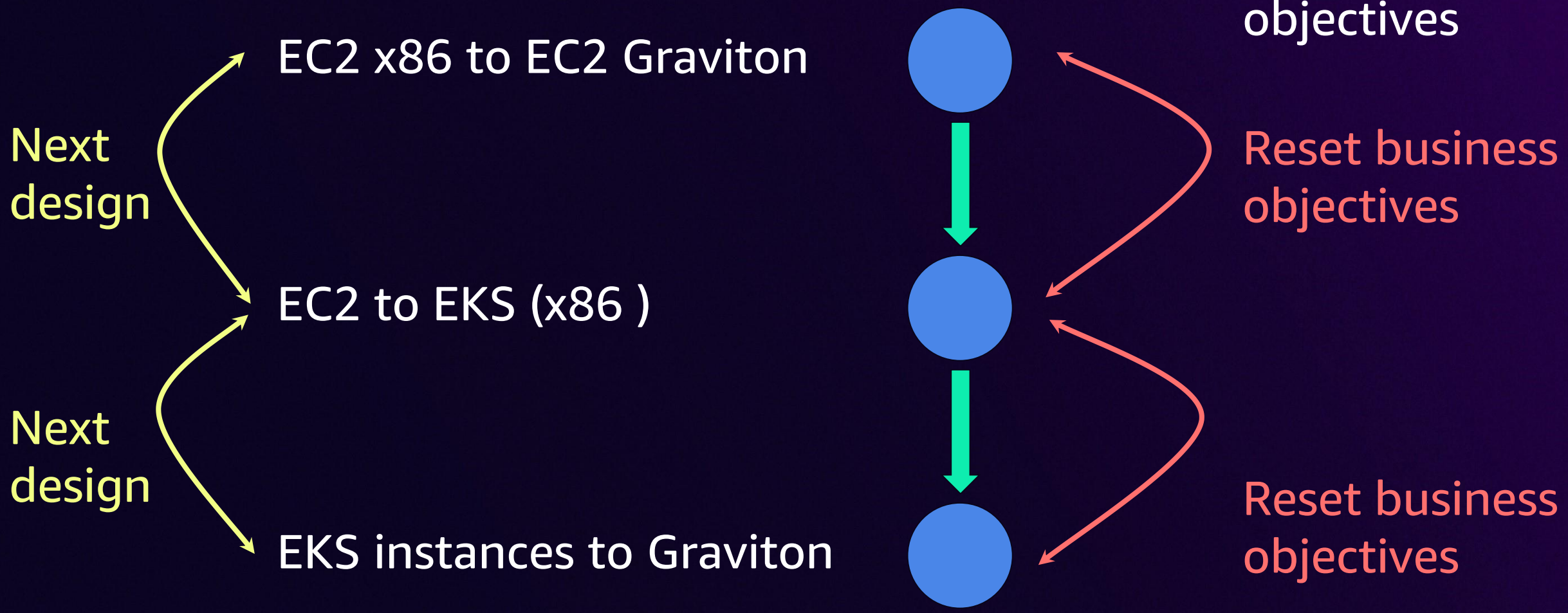
CCoE role in Coinbase

- Who delivers on this program for Coinbase
  - Cloud Center of Excellence (CCoE) ensures Coinbase uses the cloud optimally
- 3 core pillars
  - Cloud architecture excellence
    - Are we building it the right way?
  - Cloud usage excellence
    - Are we using it cost effectively and completely?
  - Cloud lifecycle management
    - Are we keeping up with the latest?
- Cloud optimization is a function of the expense of the cloud architecture divided by the rates paid for that architecture

Coinbase’s strategic objectives – All phases

- Scale for traffic surges without overprovisioning
- Cost optimize without sacrificing resiliency
- Maximize system use without crashing during scaling
- Execute swiftly without increasing headcount

10x program path at Coinbase



10x program path – Phase 1

EC2 x86 to EC2 Graviton

EC2 to EKS (X86)

EKS instances to Graviton

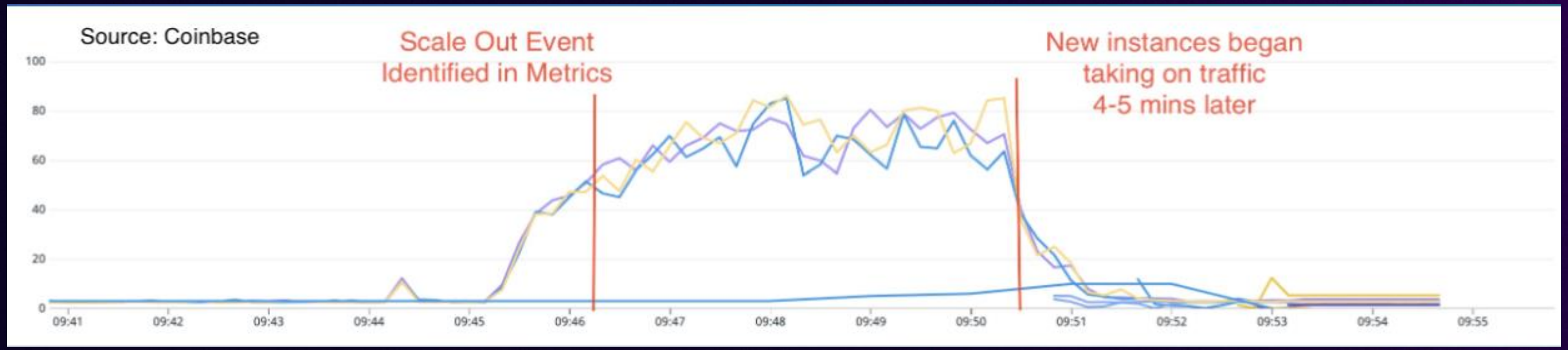
Phase 1

Improving Auto Scaling groups

Analyzed our scaling process holistically to find our bottlenecks

Not scaling fast enough

Improving Auto Scaling groups



Observability correct?

Updated our infra to support warm pools, step scaling, granular metrics

Improving Auto Scaling groups

Result

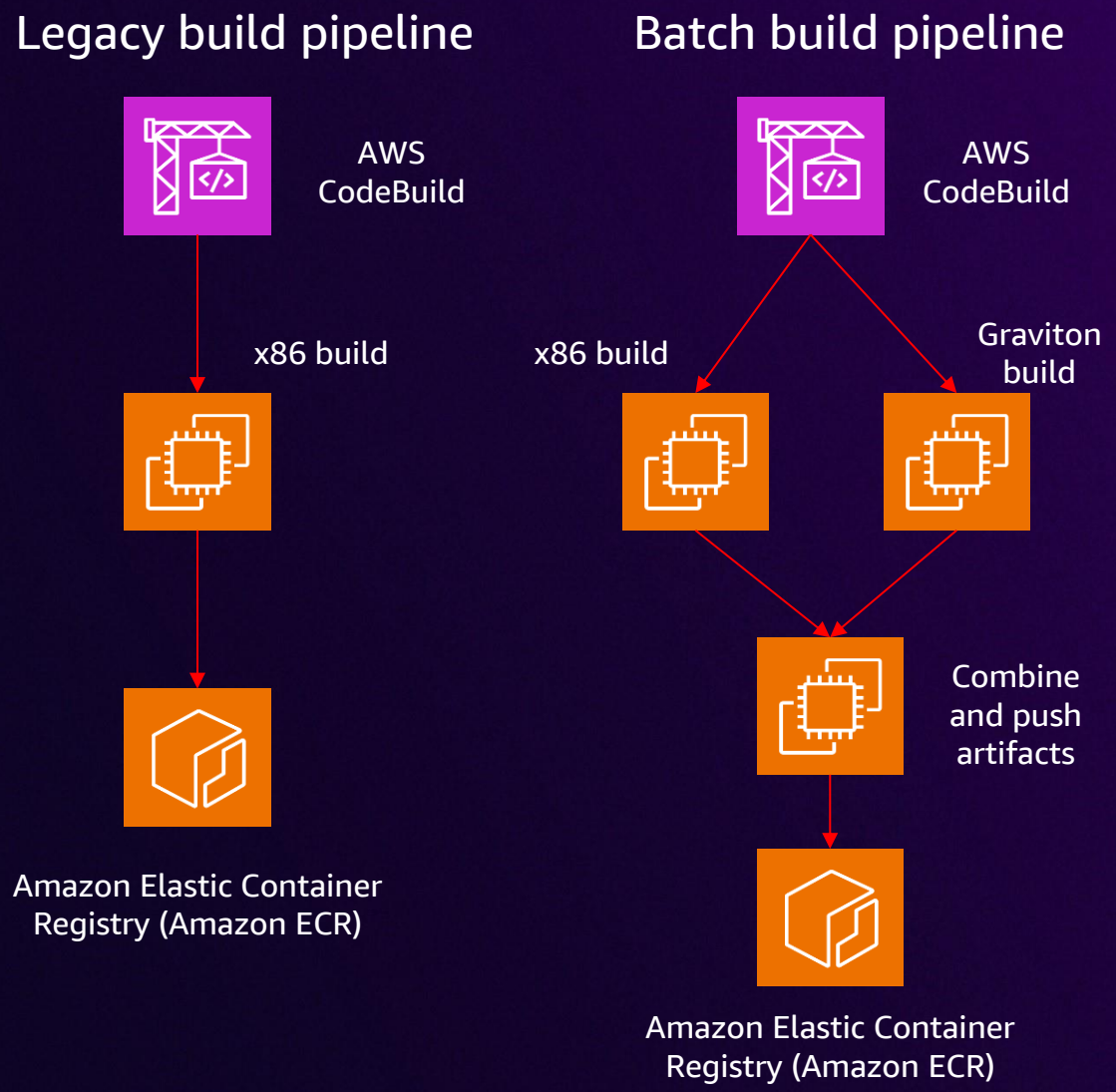
- Enabled sub-min scaling for our services
- Improved scalability, reliability, cost efficiency

Phase 1: EC2 x86 to EC2 Graviton

- Delivered: Reduced EC2 spend up to 20% by switching from x86 to Graviton using a fan-out execution approach
- Problem: How do we fan out into many teams quickly
- Solution: Crafted SOW with AWS Professional Services; selected team members based on skills needed for each project
- Action: Reached out to our TAM
  - Allocated budget based on expected cost efficiencies from architectural changes

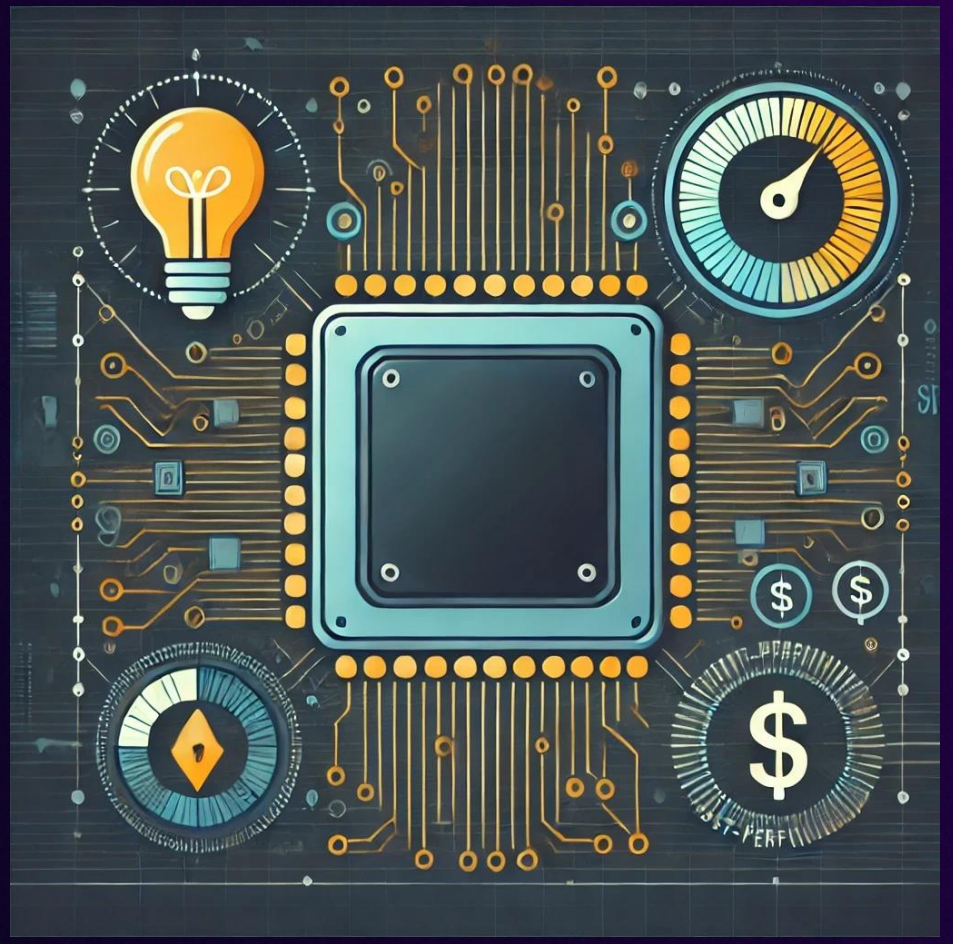
Updating build pipeline

- Graviton vs. x86
  - 20% more cost-effective
  - Better performance
  - Reduced carbon footprint
- Implementing dual architecture
  - 2x build time due to emulated builds
  - Needed to improve customer experience before rollout
  - Batch builds architecture



Graviton migration: Learnings

- Customer experience matters
- Instance availability is important
  - Couldn't use multi-instance ASGs due to warm pools feature
- Golang vs. Ruby performance
- Cloud Center of Excellence (CCoE) is building flexible workforce best practices for Coinbase



EC2 to Graviton – Partially completed

Efficiency

- Higher sustained utilization
- Dual compile, same time to compile

Cost

- Instance type savings

Resilience

- Improved latency
- Increased instance variety

↓ Resources

20% reduction in spend

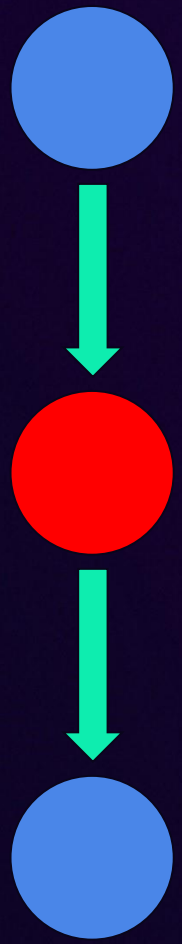
Happy customers and regulators

10x program path – Phase 2

EC2 x86 to EC2 Graviton

EC2 to EKS (x86)

EKS Instances to Graviton



Phase 2

Phase 2: EC2 to EKS

- Delivered: Compute cost optimization by 50% or more
- Problem: Cost to deliver, including incidents, kept low
- Solution: Larger project/more teams
- Action: 20 teams involved for EKS
  - Weekly program check-ins
  - Weekly senior management to highlight wins
  - Team assignments
  - Automated reports (workforce, savings)
  - Next week objectives

Delivering on ROI

- Financial payback
  - Identified higher ROI on these migrations
  - Bin packing of services
  - Faster scaling
  - Better integration and management with Amazon EKS
    - Leave the undifferentiated heavy lifting to AWS services
- Delivered: 12 months to migrate 3,500 service configurations

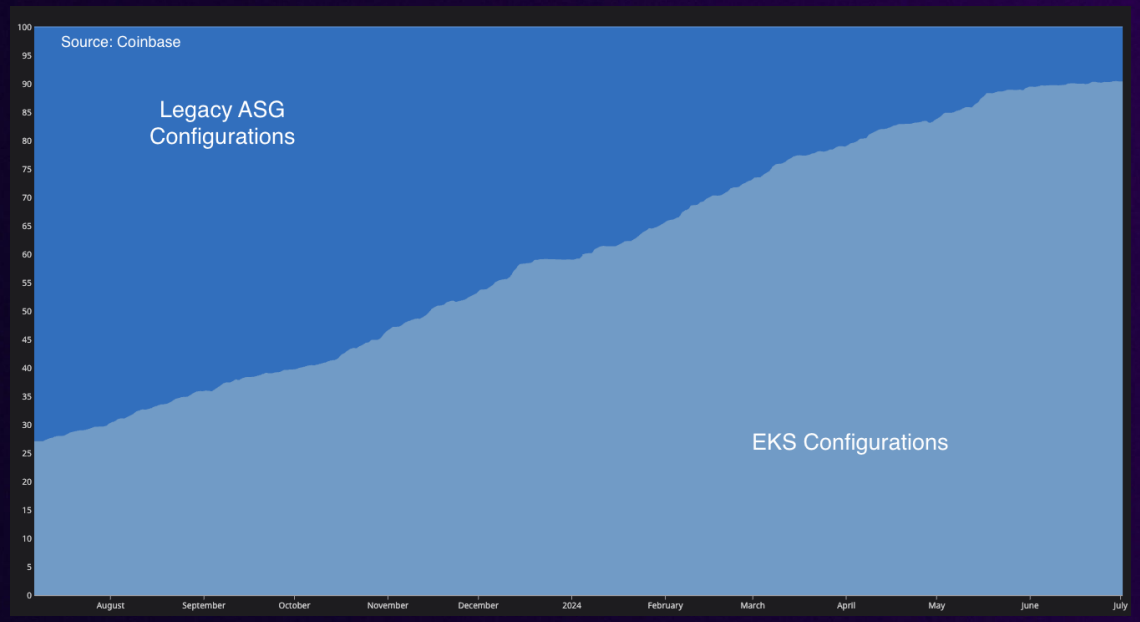
EKS: Preparing EKS migrations

- Add tooling
  - Enable automation (Kubetools)
  - Load test EKS safely
- Begin migrations with CCoE engineer
  - Gather initial learnings
  - Create guides to support all team members
- Assemble target list of in-scope services

EKS: Learnings from K8s migration

- De-risk infra-level prerequisites
  - CLB to ALB migration
- Use complex migrations as teaching moments
- Daily sessions to address blockers as a team

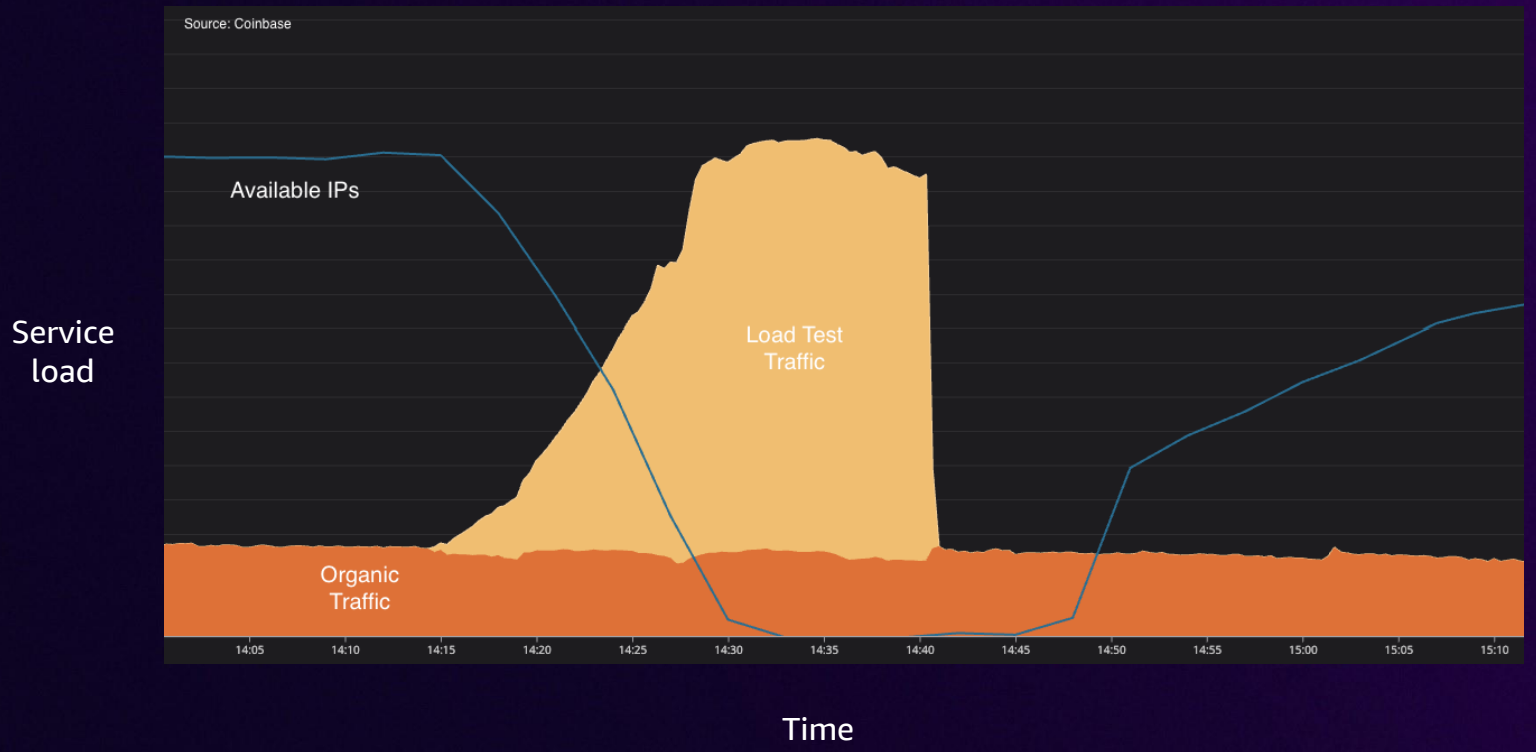
EKS migration progress



EKS: Learnings from K8s migration

- Constant load testing
  - 10x peak
- Scale created IP issues

IP usage during load tests



EKS: End state Phase 2

Efficiency

- Bin packing
- Granular resource settings

68% ↓ resources

Scalability

- Rapid scale-out
- Shared over-allocation

50% faster scaling

Centralization

- Infra management
- Managed Kubernetes offerings

↓ Operational burden

10x program path – Phase 3

EC2 x86 to EC2 Graviton

EC2 to EKS (x86)

EKS instances to Graviton

Phase 3

Phase 3: EKS instances to Graviton

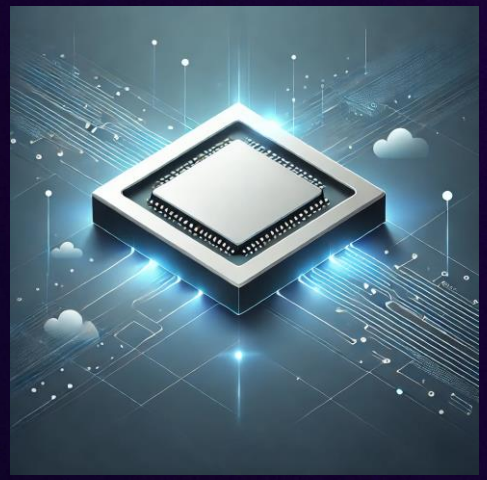
- Delivered: Efficient operations on EKS
- Problem: Our multi-architecture work was losing ground
- Solution: Merge multi-architecture and EKS projects
- Action: Restaff to have Graviton and EKS skills and new SOW

Graviton EKS migrations

- Update guides for Graviton on EKS
- Assemble target list of services to handle
  - Focus on Golang services
  - EKS cluster-based sprints
- Create tracking mechanism
- Celebrate milestones

Amazon Elastic Kubernetes Service (Amazon EKS)

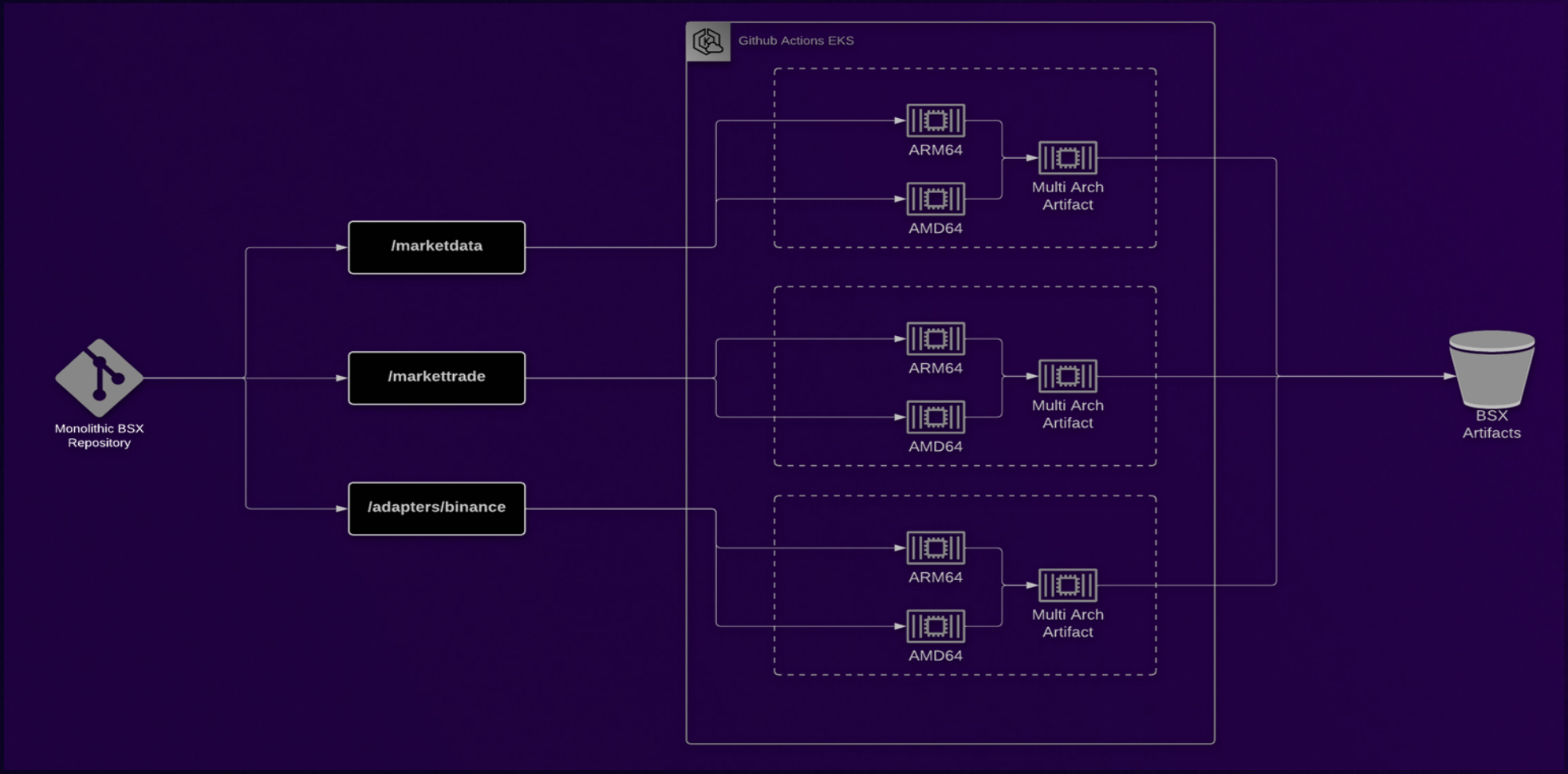
+



Graviton EKS migrations: Learnings

- Using Graviton on EKS was simpler than initial EKS migration
- Cluster Autoscaler was a blocker
  - Cluster Autoscaler vs. Karpenter
  - Worked within limitations to increase Graviton usage

Graviton EKS migrations: Learnings

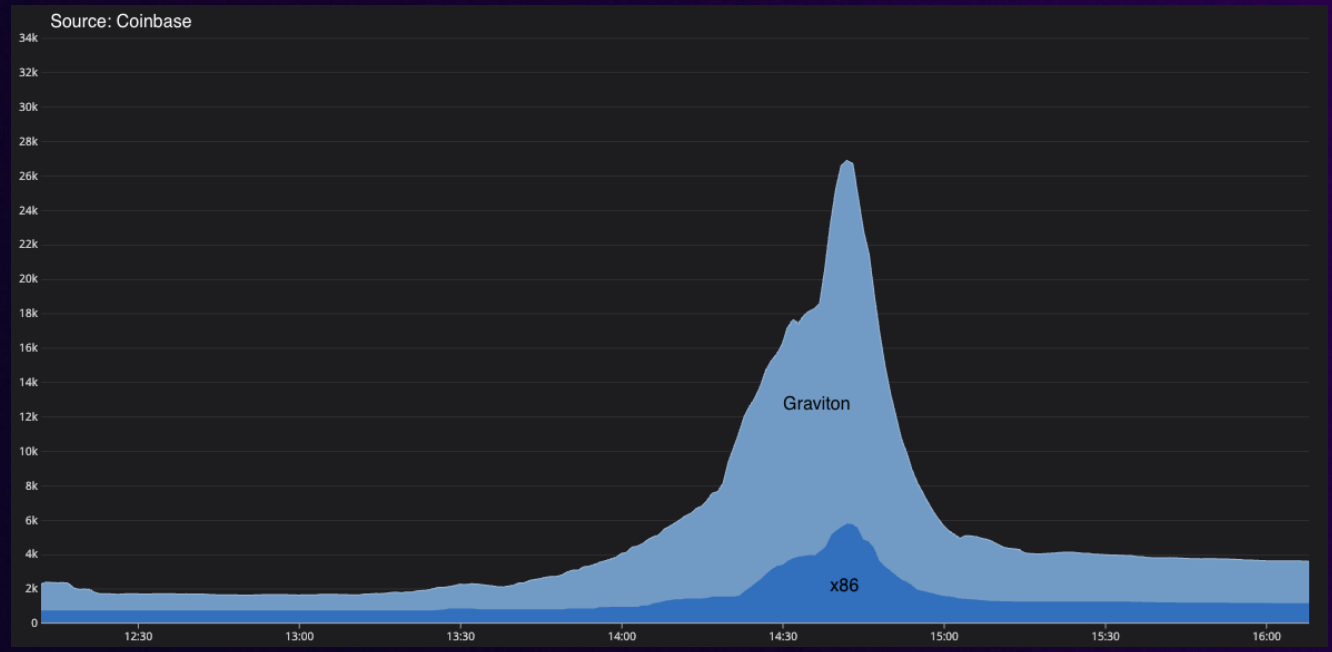


Re-architected build pipeline for 400+ artifact repos

- Enabled parallel builds

EKS scaling strategy with Graviton

- Burst into Graviton but fall back to x86
  - Cost-effective scaling
  - Instance availability is better now



EKS to Graviton – Ongoing

Efficiency

- 20% savings on instances

10% ↓ overall savings

Scalability

- Higher sustained utilization

Increased buffer capacity

Resilience

- Increased instance availability

↓ Operational issues

Learnings from 10x program

- Return on investment:
  - Focus on high-ROI programs
    - Effort + Costs savings + Customer impact -> Successful program
- Program journey 3 phases
  - Provided experience in each step for high-risk/high-volume workloads
- Define clear, single-threaded leader
  - Assists with blockers, product team coordination, and cultural onboarding
- Production deployment
  - Ensure tight PR control with multiple approval layers for safety

10x program delivers

- 10x migration program result
  - Migrated to EKS in 12 months
  - Faster and more reliable scaling
  - Cost-optimization objectives met with three phases
  - Proves ROI-based program funding can deliver
- CCoE became flexible workforce experts
  - Uniquely differentiated our team internally

coinbase

and

AWS Professional Services

Thank you!

Please complete the session survey in the mobile app