AWS re:Invent

CMP331-R1

# Save up to 90% and run production workloads on Spot Instances

**Chad Schmutzer**
Principal Developer Advocate
Amazon Web Services

**John Weber**
Senior Manager
Adobe

**Devang Sampat**
Engineering Manager
Hulu LLC

AWS re:Invent

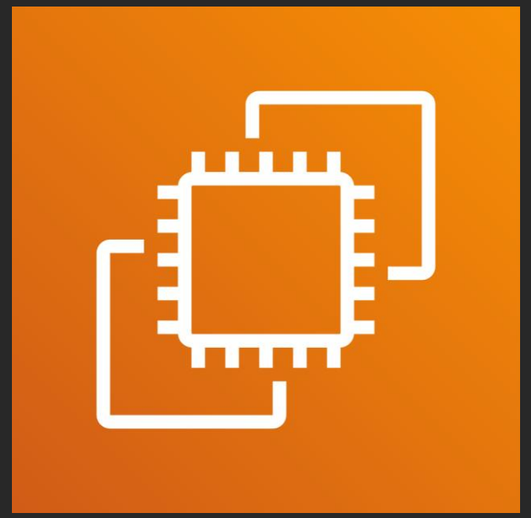


aws

# Agenda

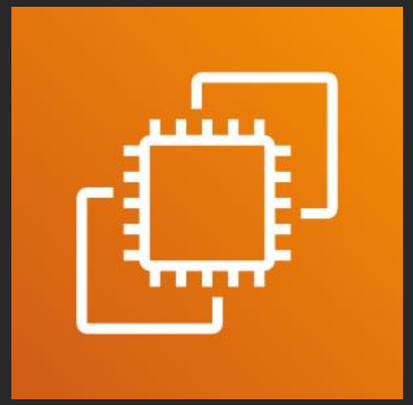
- EC2 Spot Instances—Key concepts to use Spot successfully
- EC2 Spot Instances—What's new in 2019
- Hulu—Using Spot at scale
- Adobe—Using Spot in an enterprise

# At first there was Amazon Elastic Compute Cloud (Amazon EC2)

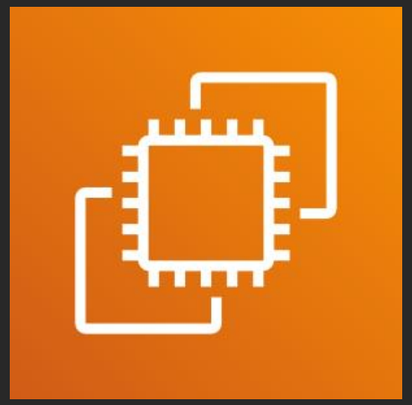


m1.small

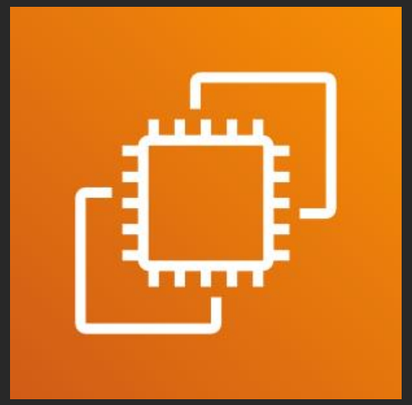
# Then we added some new instance types



m1.small

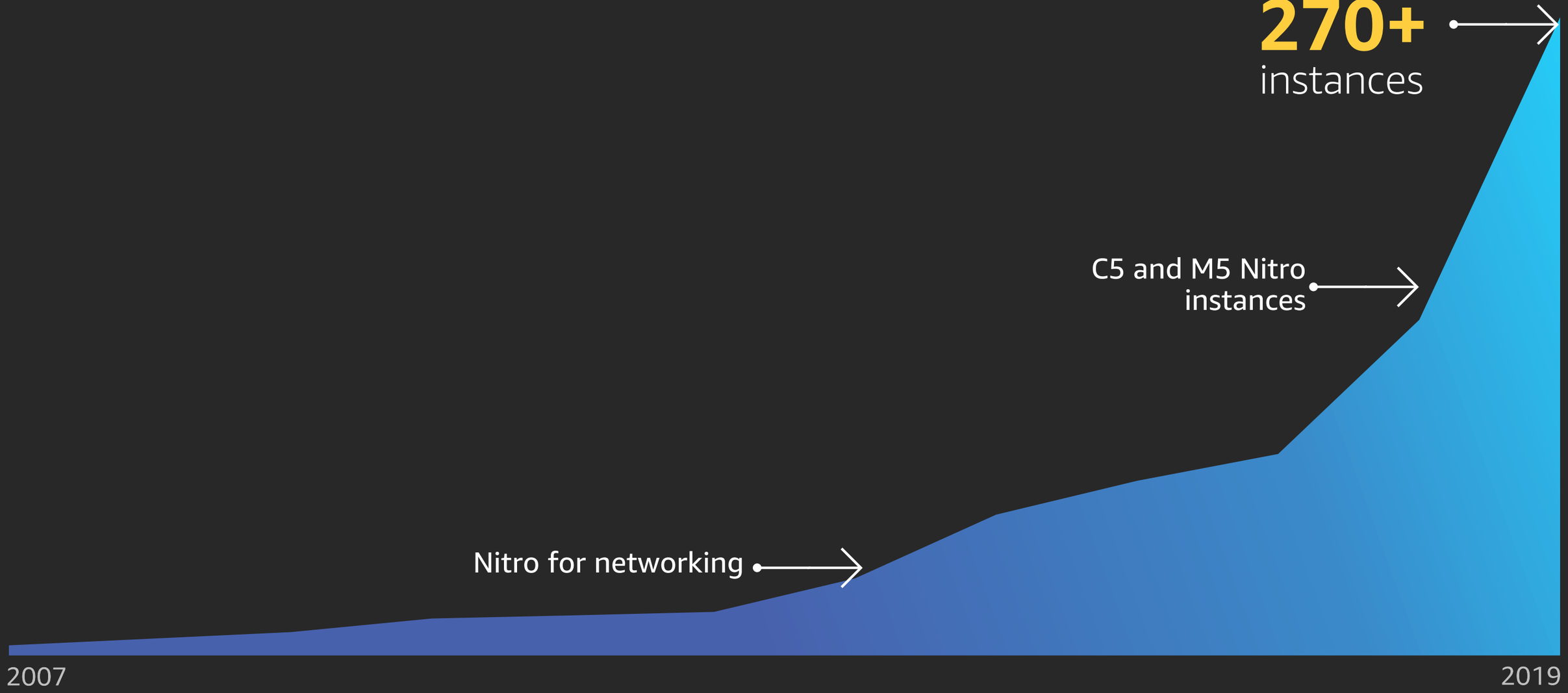


m1.large



m1.xlarge

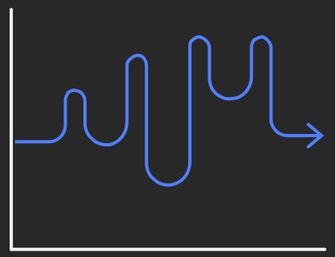
# Then we added a lot more instance types



# Amazon EC2 purchase options

## On-Demand

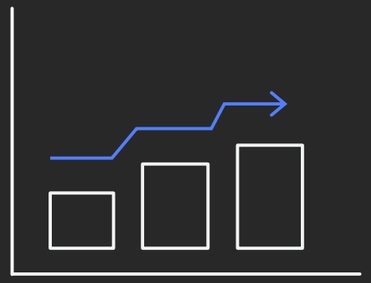
Pay-for-compute capacity **by the second** with no long-term commitments



Spiky workloads, to define needs

## Savings Plans & Reserved Instances

Make a commitment and receive a **significant discount** off compute



Committed & steady-state usage

## Spot Instances

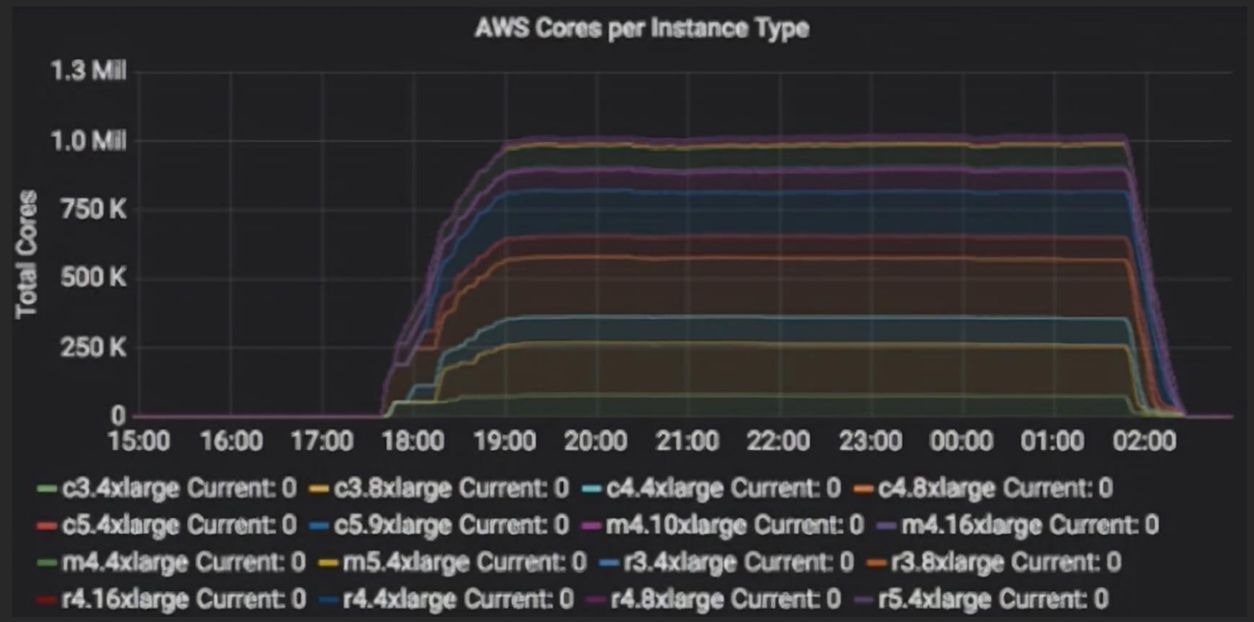
Spare EC2 capacity at **savings of up to 90%** off On-Demand prices

Fault-tolerant, flexible, stateless workloads

# Accessing spare capacity at scale

## Western Digital

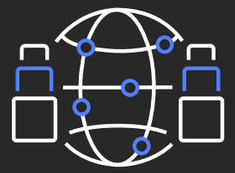
Over 2.3 million simulation jobs on a single HPC cluster of 1 million vCPUs built using Amazon EC2 Spot Instances. Time to results: 20 Days → 8 hours



# How much spare capacity is there?

*“On average, every week, AWS customers are using more compute capacity on Amazon EC2 Spot Instances than customers were running across all of Amazon EC2 in 2014."*

# What do I need to know about Spot Instances?



## Spot is spare capacity

Same infrastructure as On-Demand

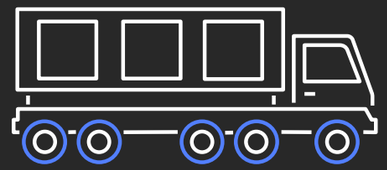
70–90% off

## Spot pricing

Smooth, infrequent changes, no spikes, more predictable

## Interruptions

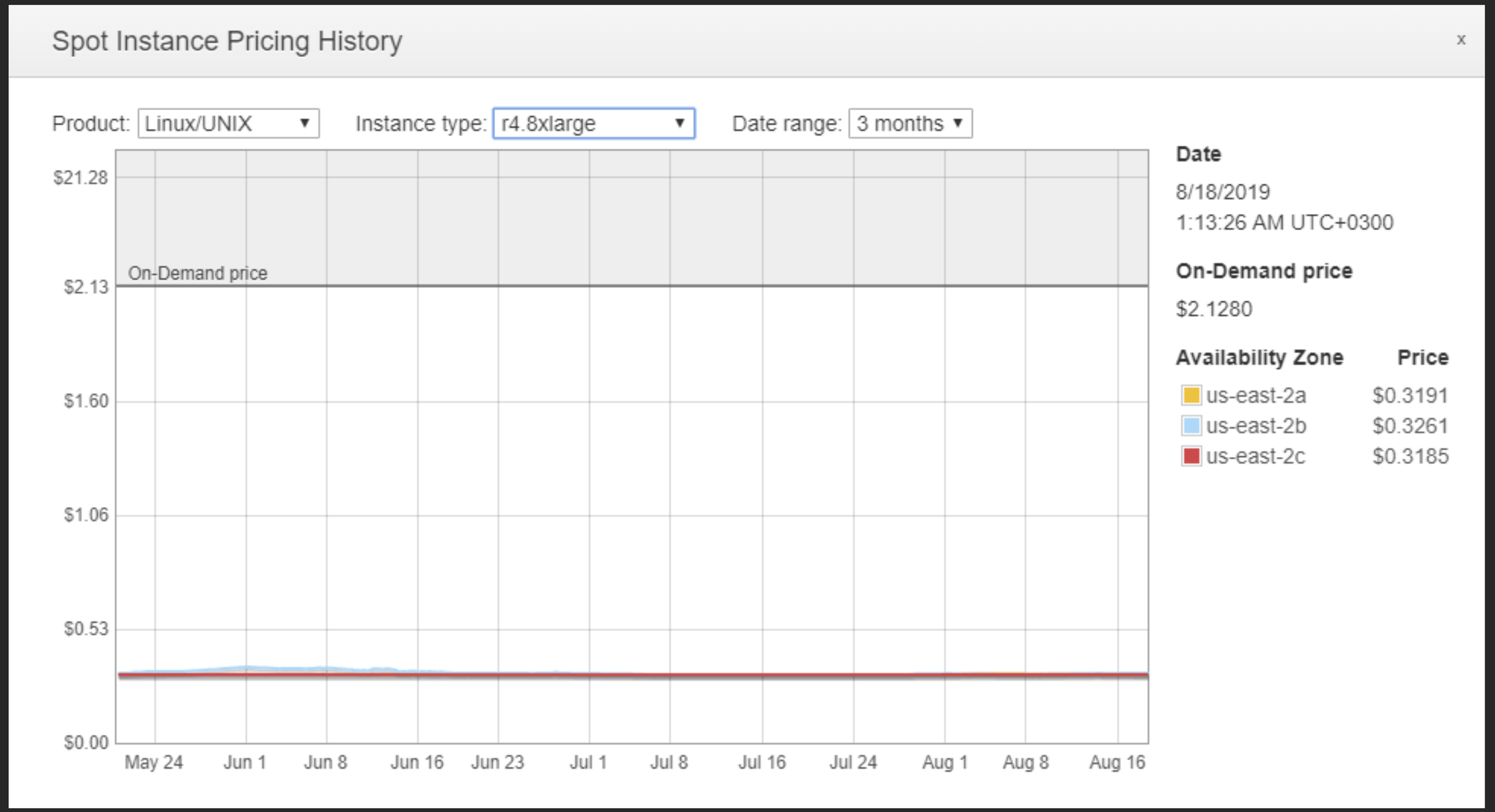
Only happen when On-Demand needs capacity (no bidding)



## Be flexible

Choose multiple instance types, sizes, and Availability Zones

# Pricing model

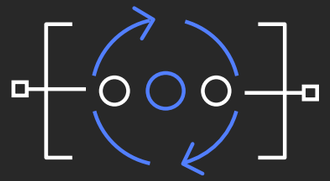


# Spot interruptions

## Minimal interruptions

Less than 5% of all running Spot Instances were interrupted in the last 3 months

The work you are doing to make your applications fault-tolerant also benefits Spot



Spot is optimized for stateless, fault-tolerant, or flexible workloads

# EC2 Spot pools—instance type flexibility

| C4 | 1a | 1b | 1c | On-Demand |
|---|---|---|---|---|
| 8XL | $0.28 | $0.27 | $0.29 | $1.76 |
| 4XL | $0.21 | $0.19 | $0.16 | $0.88 |
| 2XL | $0.08 | $0.07 | $0.08 | $0.44 |
| XL | $0.04 | $0.05 | $0.04 | $0.22 |
| L | $0.01 | $0.01 | $0.02 | $0.11 |

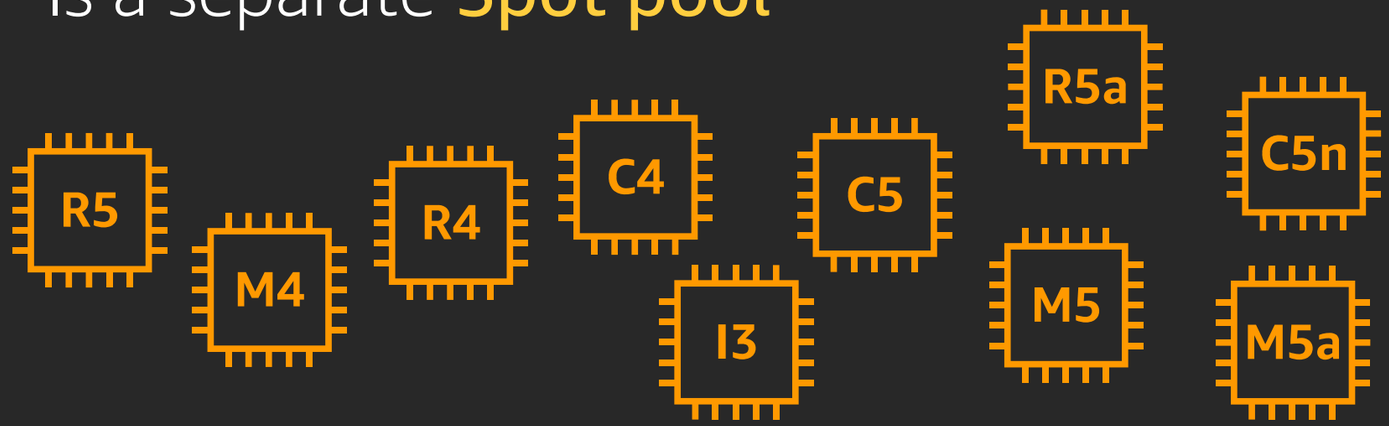
Each instance family

Each instance size

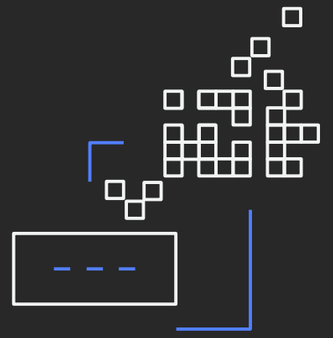
Each Availability Zone

In every Region

Is a separate **Spot pool**

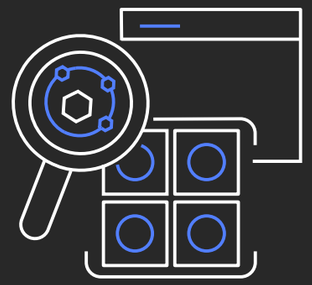


# Getting started with Spot



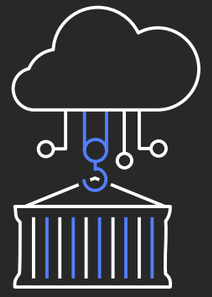
Big data

CI/CD



Web services

High performance computing



Or any containerized workload

Spot is ideal for:

- ☑ Fault-tolerant
- ☑ Flexible
- ☑ Loosely coupled
- ☑ Stateless workloads

# What’s new for Spot in 2019

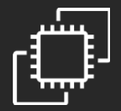
AWS re:Invent



aws

# AWS and third-party integrations make this easy and efficient

Amazon EC2 Auto Scaling



EC2 Fleet


Amazon Elastic Container Service (Amazon ECS)


Amazon Elastic Container Service for Kubernetes (Amazon EKS)


Amazon SageMaker

AWS Thinkbox

Amazon EMR

AWS CloudFormation


AWS Batch


AWS Elastic Beanstalk

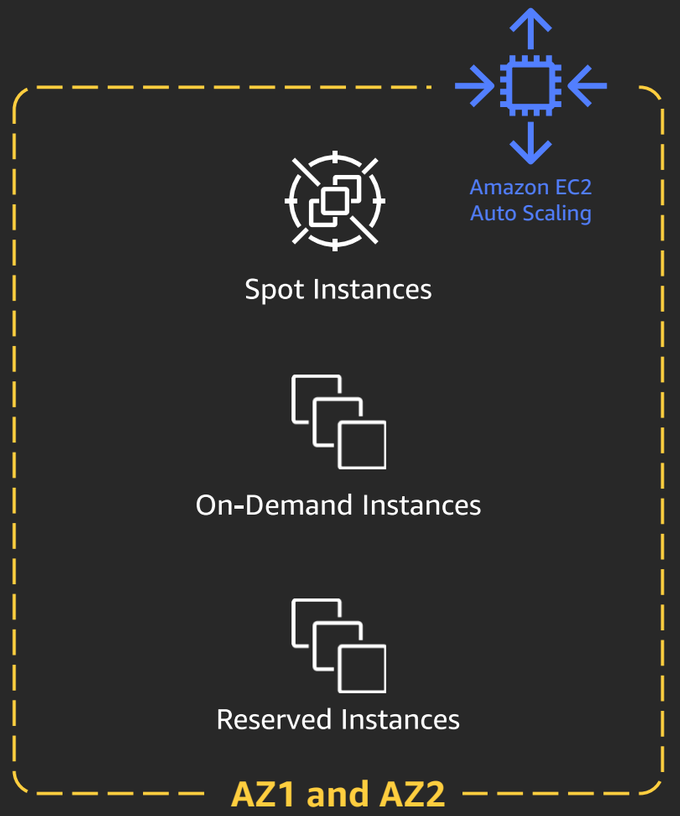




# Save up to 90% using EC2 Auto Scaling

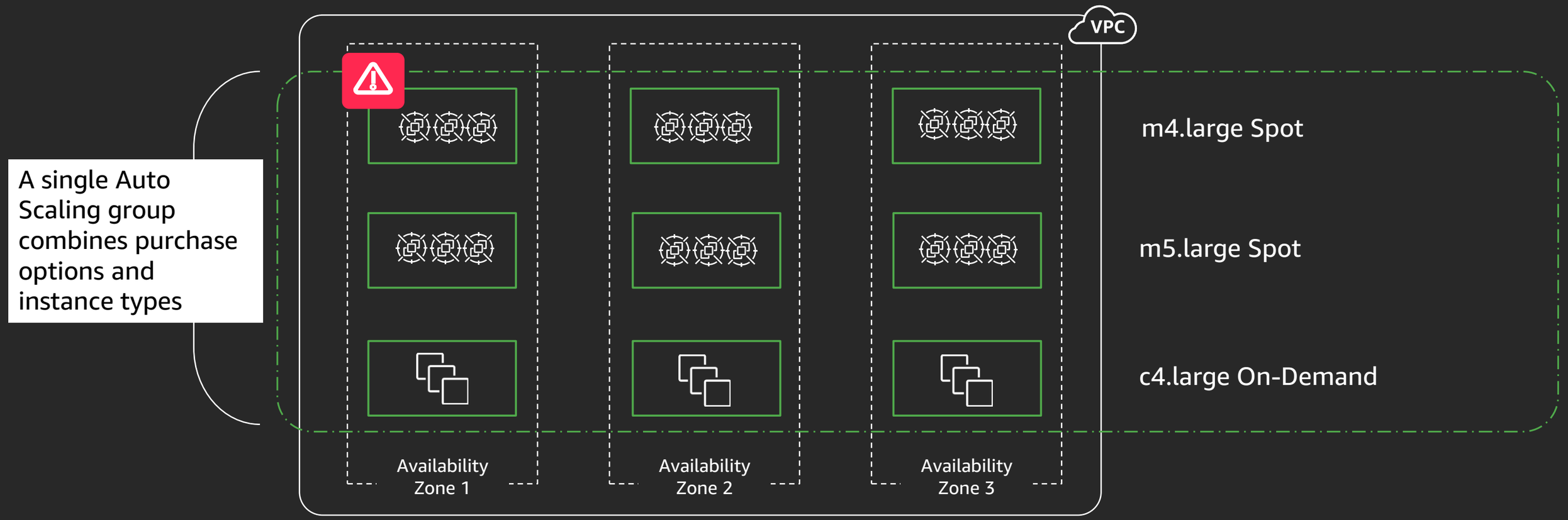
**Automatically scale instances across instance families and purchase models in a single Auto Scaling group**

- Capacity optimized NEW!
- Lowest cost
- Prioritized list




**Reduce cost** **Optimize performance** **Eliminate operational overhead**

# EC2 Auto Scaling – Mixed Instances Policy



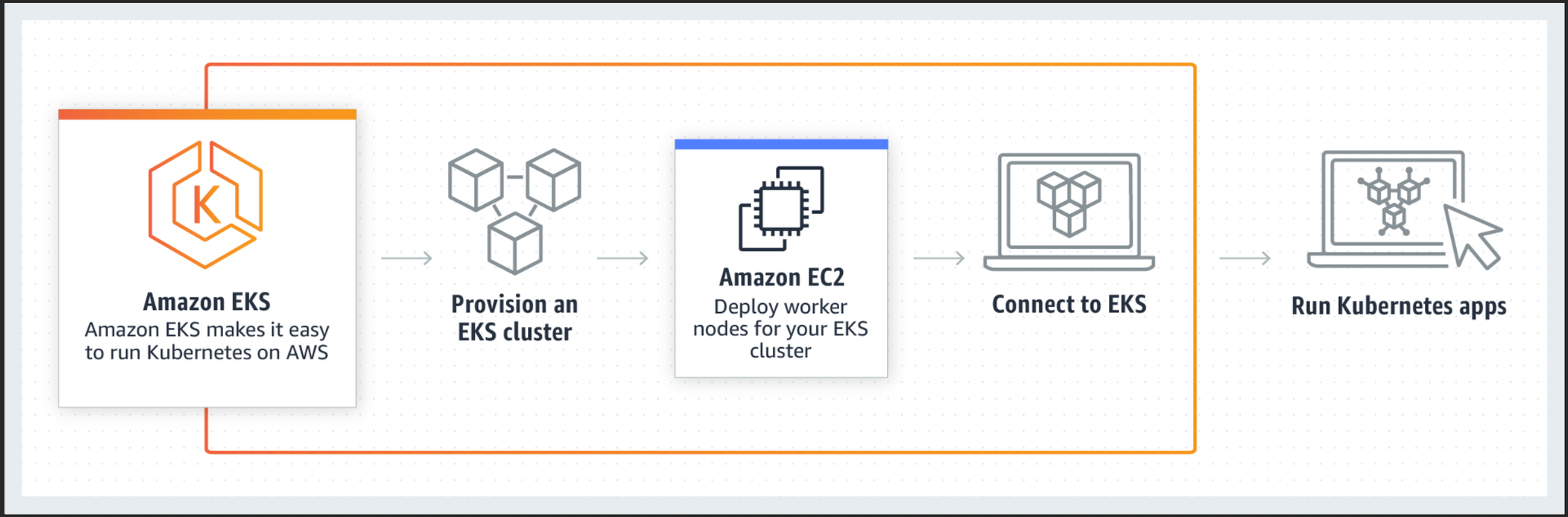
# Amazon ECS

Automated draining
for Spot Instances running ECS services

Fully manages Spot clusters

https://amzn.to/2Dzo9HV

# Amazon EKS

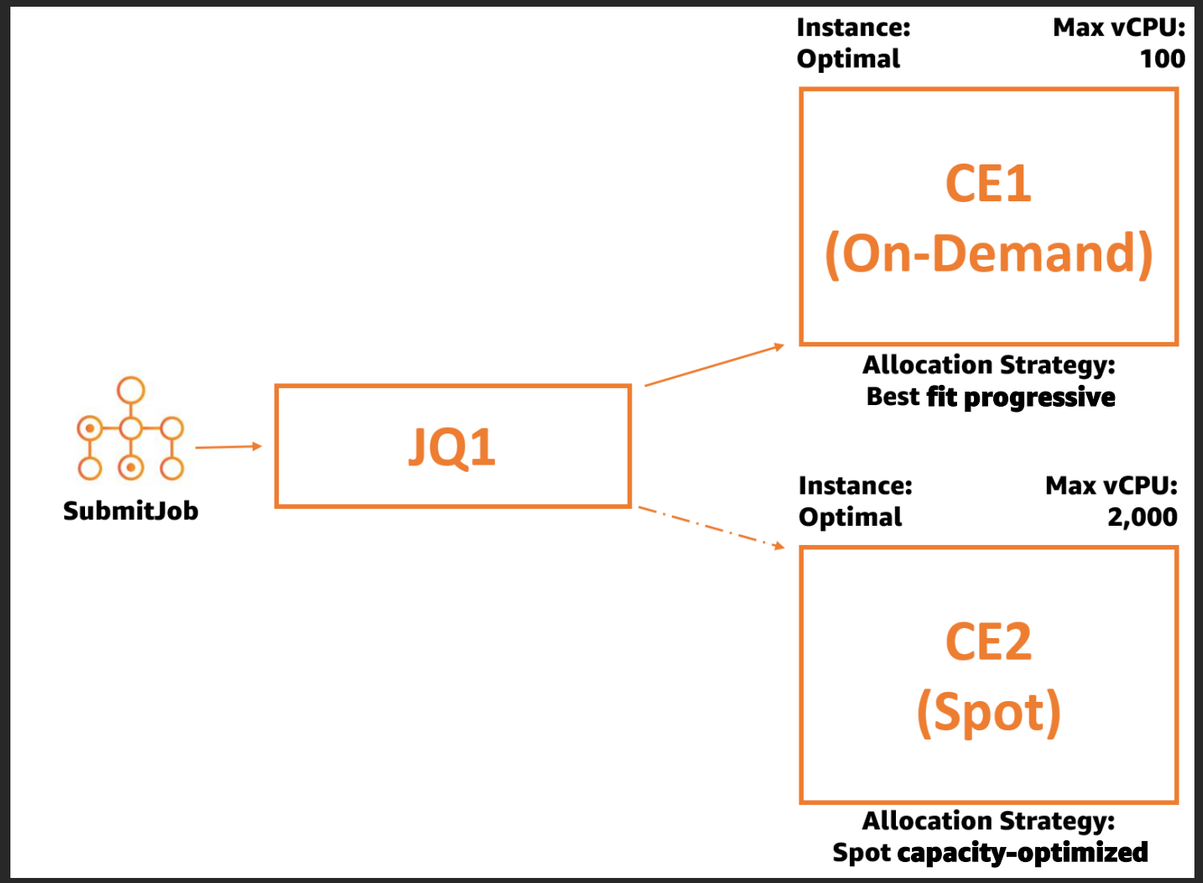



AWS node termination handler supports automated draining for Spot Instance nodes on Kubernetes

https://github.com/aws/aws-node-termination-handler

# AWS Batch

1. Fully managed
2. Cost-optimized resource provisioning
3. Allocation strategies
    1. Spot capacity-optimized strategy
    2. Best fit progressive




https://amzn.to/33Bn7ph

# Savings at scale

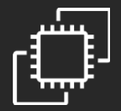
**NextRoll**

"Over 2019 alone, Batch and Spot market usage has saved us **$4,000,000**. That saving comes from comparing our model to the usage of EC2 on-demand instances to run our jobs."

http://tech.nextroll.com/blog/dev/2019/11/19/aws-batch-at-nextroll.html

# AWS and third-party integrations make this easy and efficient

Amazon EC2 Auto Scaling



EC2 Fleet


Amazon Elastic Container Service (Amazon ECS)


Amazon Elastic Container Service for Kubernetes (Amazon EKS)


Amazon SageMaker

AWS Thinkbox

Amazon EMR

AWS CloudFormation


AWS Batch


AWS Elastic Beanstalk





# Spot in production at Hulu

# What is Hulu?

# Workloads at Hulu

## Video on demand (VOD)

- As long as playback starts within seconds, latency is not a concern
- Caching requires you to think about how to keep a long tail of content ready
- Load profile follows predictable peaks and troughs
  - Most people get home from work and watch during 5–9PM

## Live

- Extremely latency sensitive (your neighbors cheer before you see the action)
- Local affiliates means we can have 50–100 different streams for the same live event
  - Geographic distribution of viewers matters
- Load profile can be spiky especially for sports
  - You can double viewership within seconds to low minutes if there’s a touchdown

**The unpredictability and spiky nature of Live TV provides the biggest challenges from a scaling perspective**

# The problem

# Requirements

## Problem

- Business need: Can we build a system to simulate a spike of XX million viewers? (Load testing framework)
- Engineering definition: Can we build a system to deploy, bootstrap, and execute X million instances of a code artifact within Y minutes (Batch job processing)
- As long as your workload can follow certain assumptions, the system is reusable for other applications

## Assumptions

- Developers want to deploy an artifact written in an arbitrary technology
- Each instance is stateless AFTER it is provisioned
- We should be cost efficient
- We need to be able to stop a workload within seconds

# The solution

# Workflow

V1: Pull an image directly off Amazon Elastic Container Registry (Amazon ECR)

- Max ~75K vCPUs

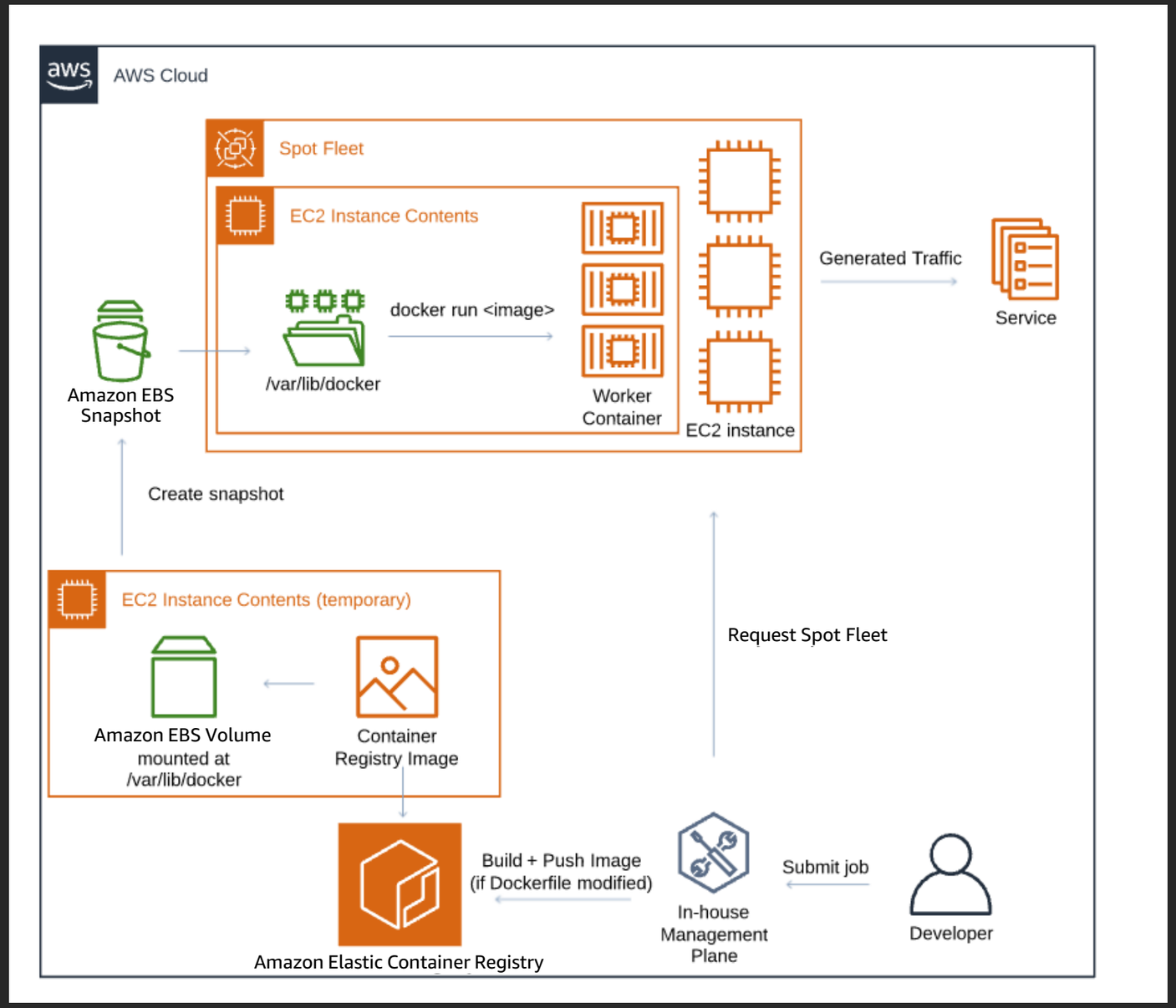
V2: Add an Amazon Elastic Block Store (Amazon EBS) factory to cache the image

- Max ~200K vCPUs

Now we encountered different Amazon Elastic Compute Cloud (Amazon EC2) issues that stopped us from reaching desired capacity

- Running out of IPs
- Unable to find sufficient hardware
- Account limits

What was going on?



# Version 3

## Running out of IPs

- Our networking design was flawed
  - Using 2 Regions (US-East-1 and US-West-2) and requested 1 Spot Fleet in each Region
  - Each Region had 1 VPC and 3 AZs
  - Each AZ had a /20
  - The entire IP space was a/8 that was shared with normal application workloads AND our data center
    - Getting a larger contiguous subnet was challenging
- Solutions:
  - Got our own/12

## Unable to find sufficient hardware

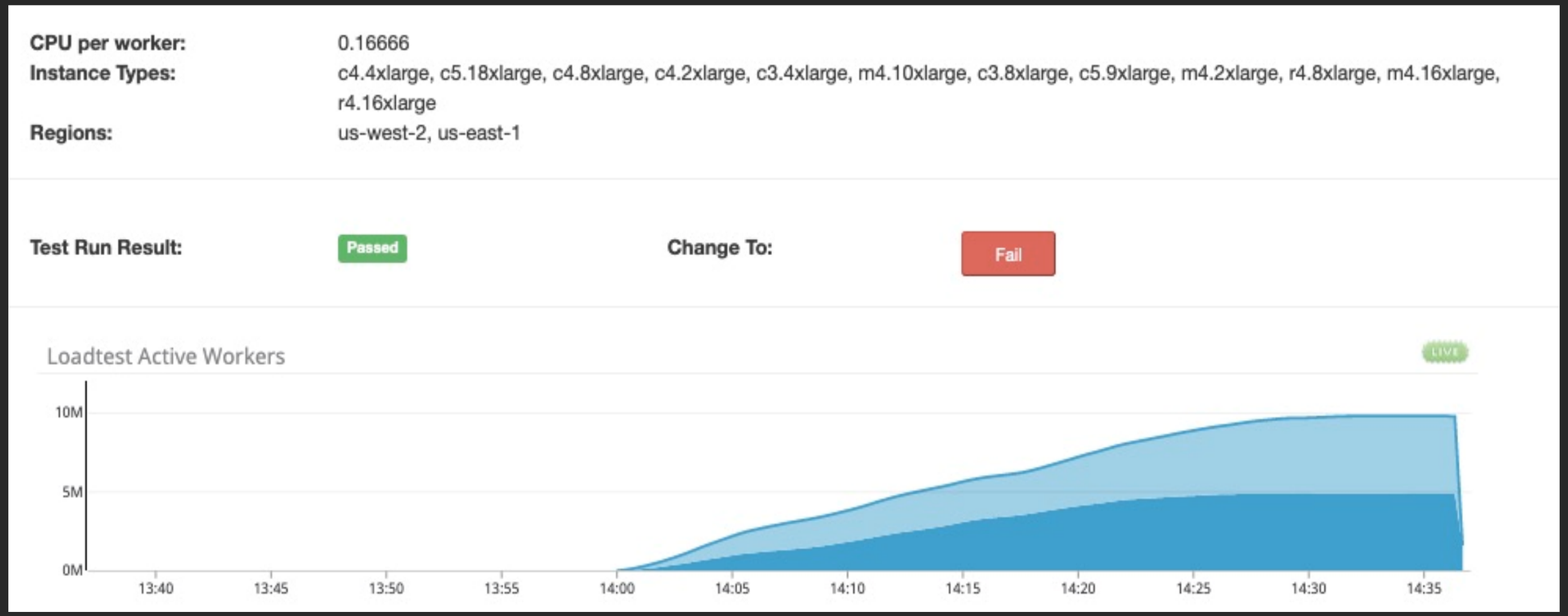
- We had limited ourselves to use only a few instance types to minimize the number of IPs
- The more variety of instance types, the more available capacity you can tap
- Switched to using the new capacity-optimized Spot Fleet allocation strategy
  - Issues with fractional weights
  - Reduced our complexity because target capacity is capped at 3000 units, so we had to hack it using weighted capacity as a scale factor
  - The potential for higher cost was worth it if the provisioning was more reliable

## Account limits

- We had a single AWS account for all of Hulu. Every time we bumped a limit, another team would end up using our capacity for something else

# Great success

- 1,666,660 vCPUs engine
- Dual Region intake up to 87,648 instances
- 0 to ~10M containers in 30 minutes



# Spot in production at Adobe

# Adobe has changed over time

Born before modern Windows even existed
(anyone remember PostScript?)

Grew up selling perpetual licenses for boxed products

Successful service exploration in 2008, decision to move to the cloud

One of the first major companies to move to subscriptions

Thousands of developers had to adapt quickly from shrink-wrap to SaaS

# Out with the old…and in with the new

Data centers
Stateful
Monoliths
Perpetual

Cloud
Stateless
Microservices
Volatile

# Enterprise challenges

Dozens of services

Hundreds of teams

Pros

- Creativity is in our DNA
- Agility and innovation

Cons

- Rigid silos and patterns
- Many teams to work with

"Inconsistently Inconsistent"

# Spot is easy

Anything stateless

Anything autoscaling

Low priority work

Low cost acceleration

# Spot is hard

How does one deal with state?

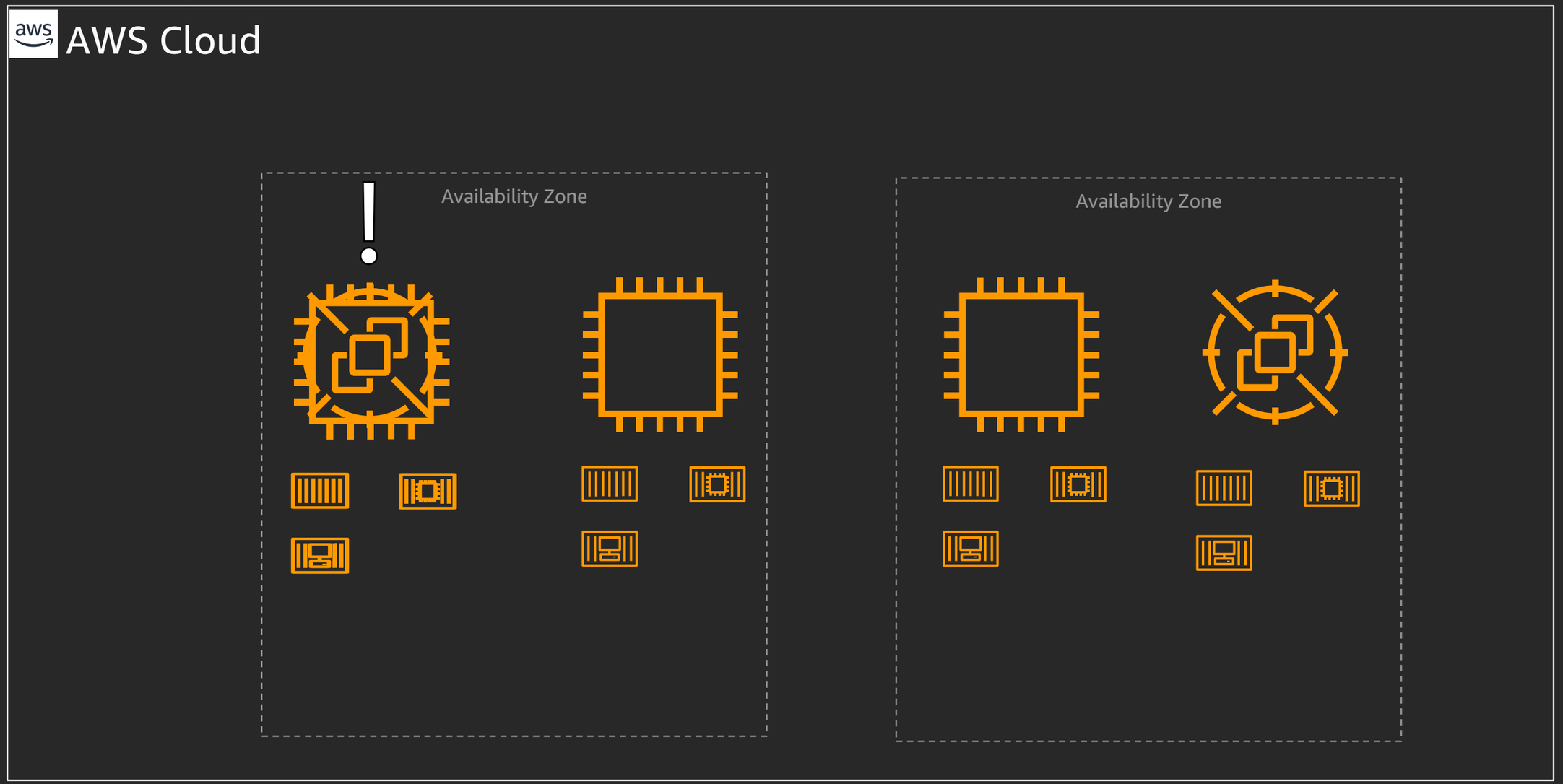
- Perpetual instances
- Hibernate
- Amazon Elastic File System (Amazon EFS)

# Platforms for the win

## Today

- Ethos – Adobe's multi-tenant environment that leverages Spot across environments
- "Give and take"
- v3 Cs:
  - Containerization
  - CI/CD
  - Clusterization

# Signaling



# What’s next?

## Tomorrow

- K8s
- Serverless

## In-between

- Continued work with additional internal services to find fits

# Thank you!

AWS re:Invent



aws

# Please complete the session survey in the mobile app.