Firecracker: secure and fast microVMs for serverless computing

Arun Gupta, Radu Weiss

2019-06-18

What is Firecracker?

Open source virtualization technology (microVM)

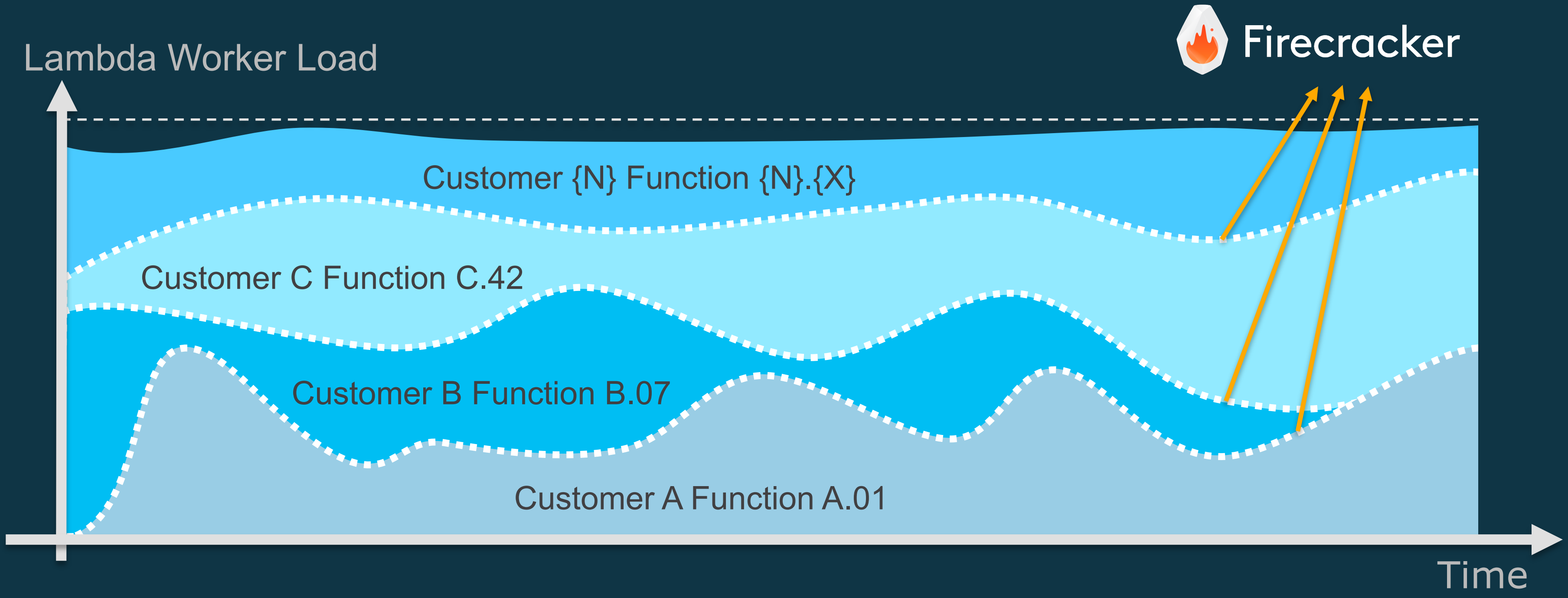
Security and isolation of traditional VMs

Speed and density of containers

Low resource overhead

Developed at Amazon

What kind of runtimes do we need for serverless?



Benefits of Firecracker

Security ↑

Startup time ↓

Utilization ↑

Benefits of Firecracker

Security from the ground up

KVM-based virtualization

Speed by design

<125ms to launch 150 microVMs per second/host

Scale and efficiency

<5MB memory footprint per microVM

Firecracker design principles

Multitenant

Any vCPU and memory combination

Oversubscription permissible

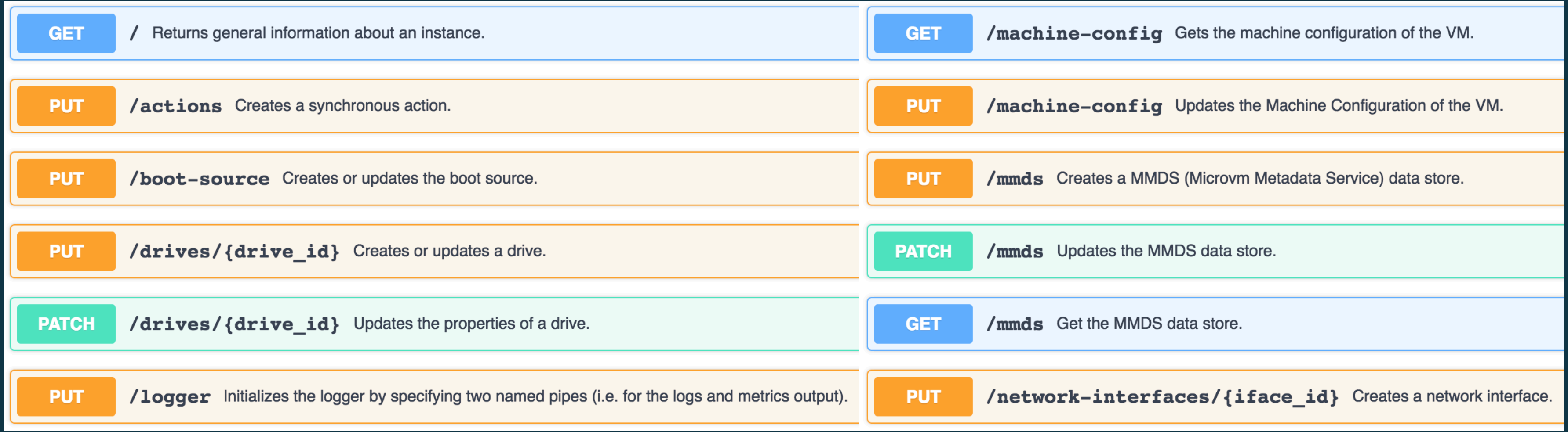
Mutation rate: 100+ microVMs/host/sec

Limited only by hardware resources

Host-facing REST API

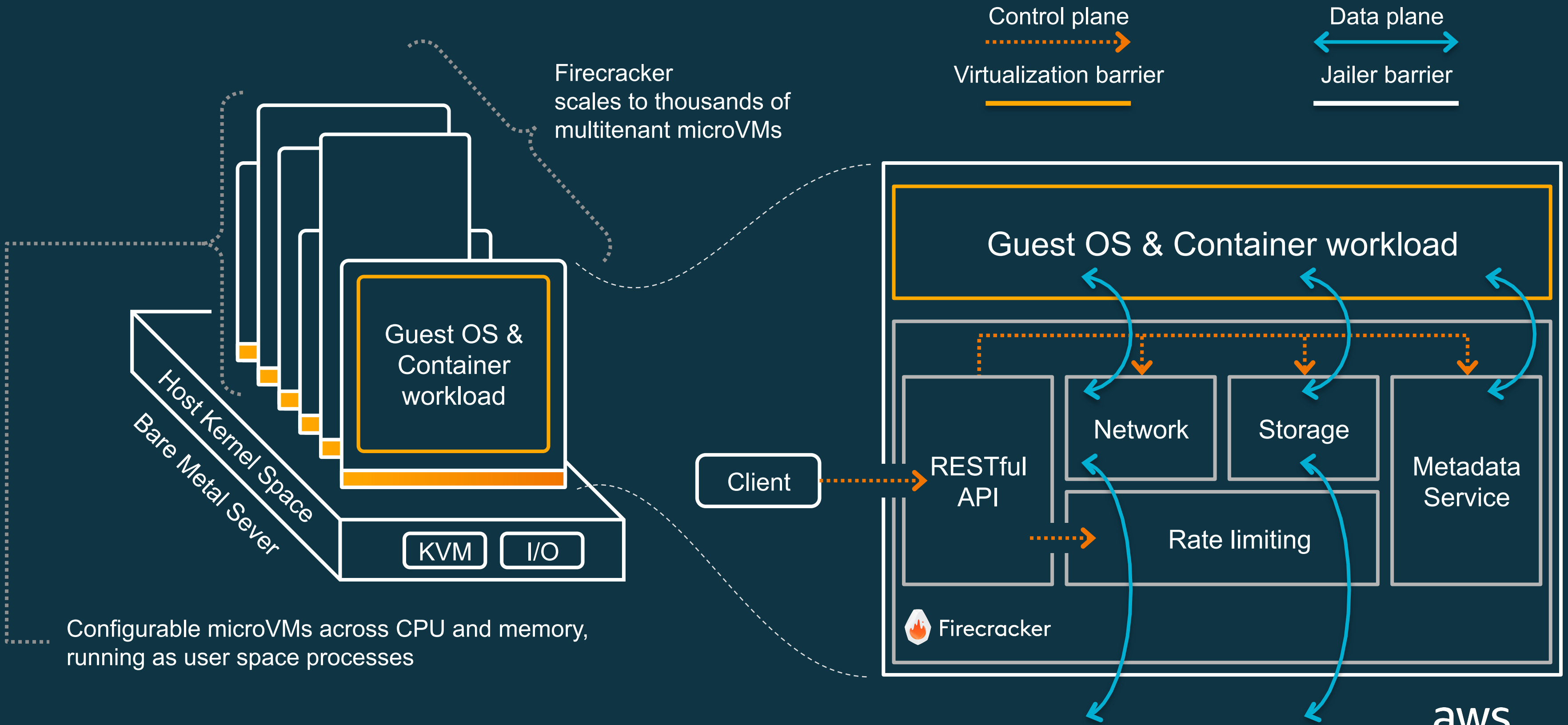
Minimalist guest machine/device model

Host-facing REST API



GET	`/`	Returns general information about an instance.
PUT	`/actions`	Creates a synchronous action.
PUT	`/boot-source`	Creates or updates the boot source.
PUT	`/drives/{drive_id}`	Creates or updates a drive.
PATCH	`/drives/{drive_id}`	Updates the properties of a drive.
PUT	`/logger`	Initializes the logger by specifying two named pipes (i.e. for the logs and metrics output).
GET	`/machine-config`	Gets the machine configuration of the VM.
PUT	`/machine-config`	Updates the Machine Configuration of the VM.
PUT	`/mmds`	Creates a MMDS (Microvm Metadata Service) data store.
PATCH	`/mmds`	Updates the MMDS data store.
GET	`/mmds`	Get the MMDS data store.
PUT	`/network-interfaces/{iface_id}`	Creates a network interface.

Firecracker architecture



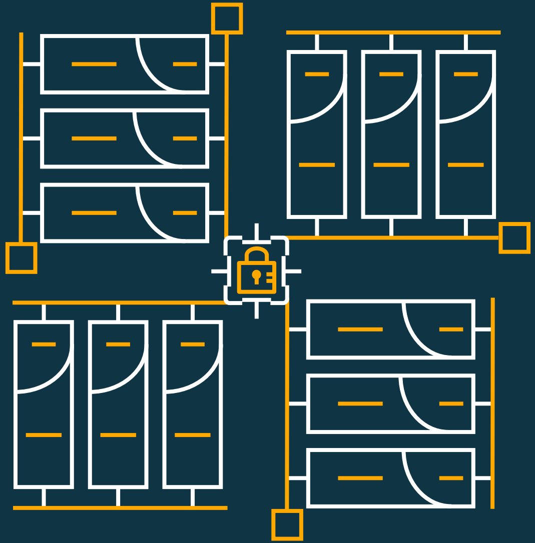
Firecracker
scales to thousands of
multitenant microVMs

Configurable microVMs across CPU and memory,
running as user space processes

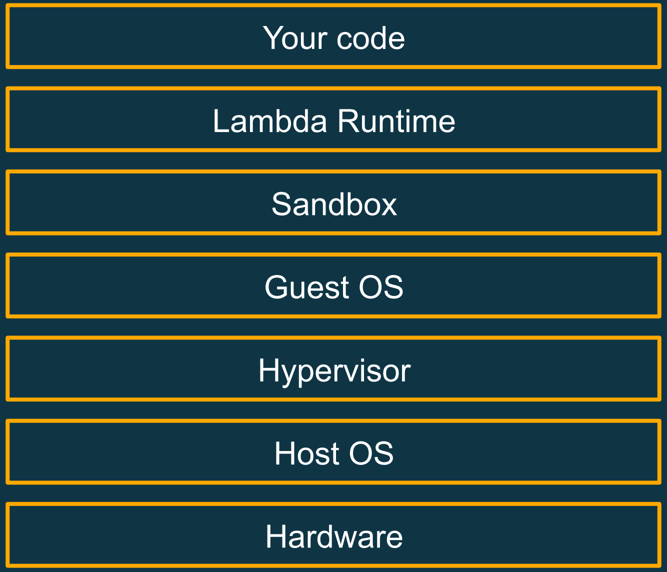
AWS Lambda

Lambda worker

Provisions a secure environment for customer code execution



Lambda worker architecture



Your code

Lambda Runtime

Sandbox

Guest OS

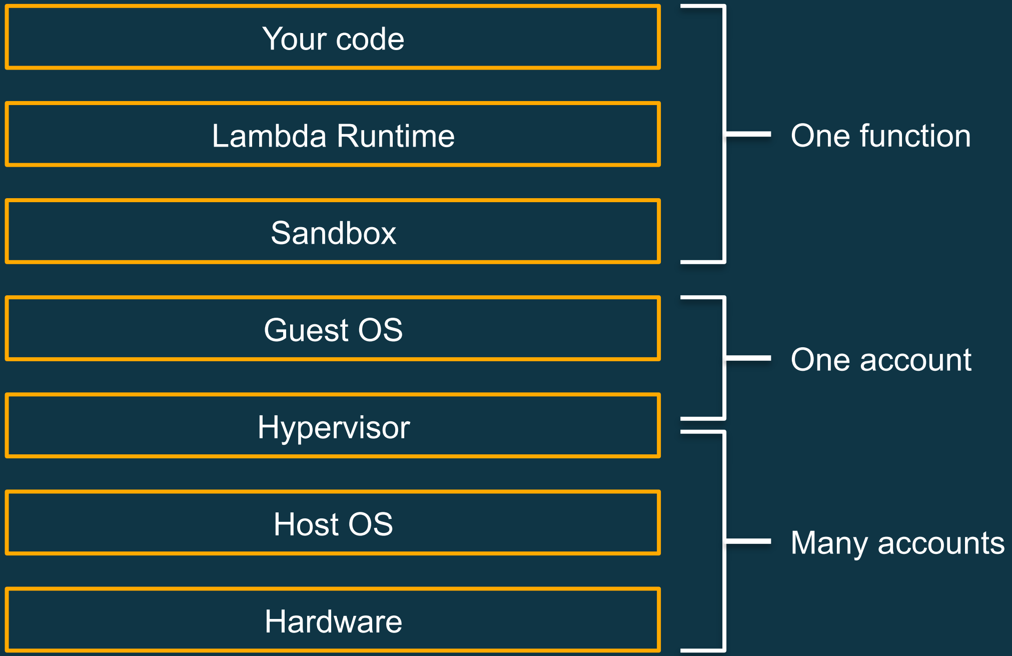
Hypervisor

Host OS

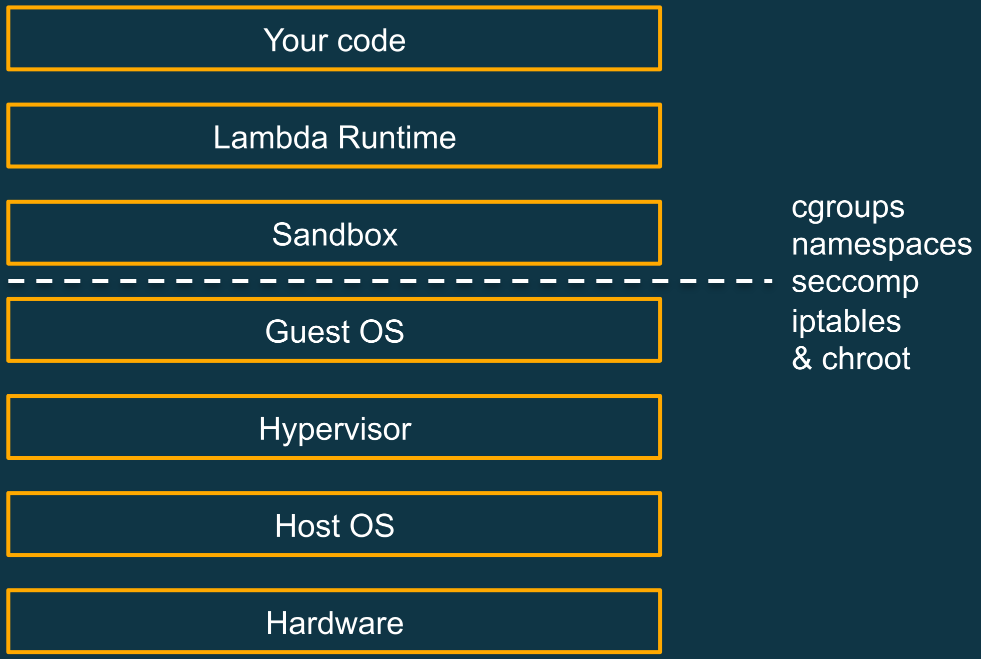
Hardware

Lambda isolation

Keeping workloads safe and separate



Lambda security boundaries



Your code

Lambda Runtime

Sandbox

Guest OS

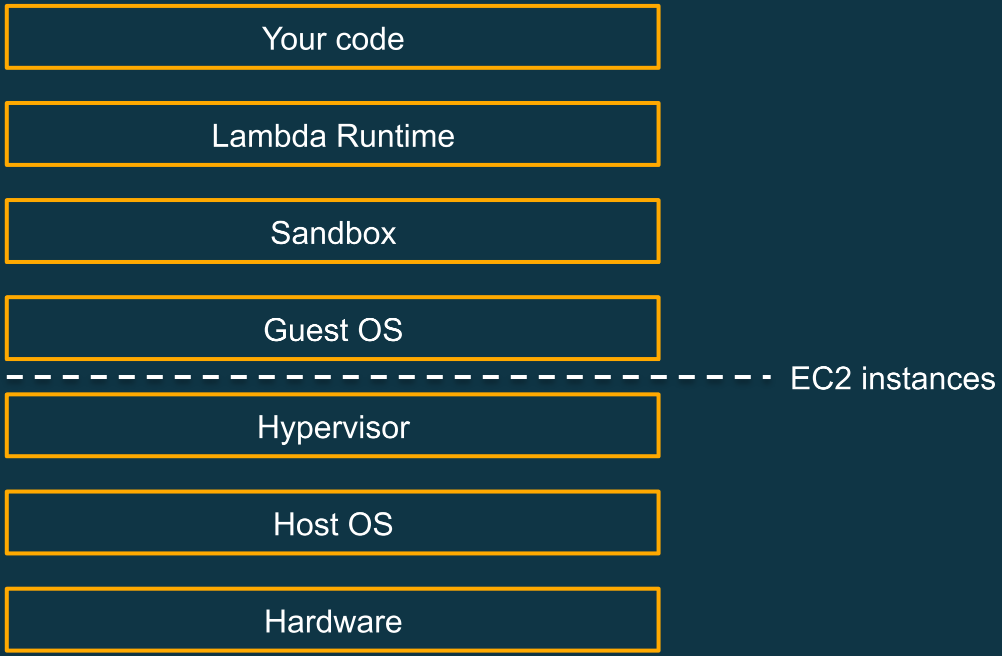
Hypervisor

Host OS

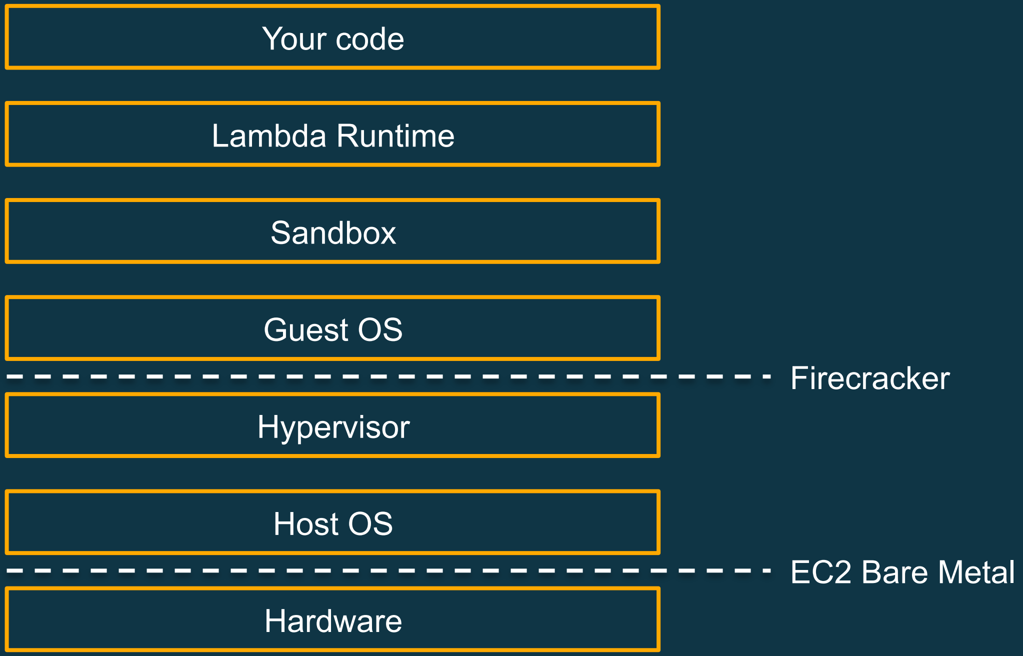
Hardware

cgroups
namespaces
seccomp
iptables
& chroot

Lambda isolation using EC2



Lambda isolation using Firecracker



Your code

Lambda Runtime

Sandbox

Guest OS

Firecracker

Hypervisor

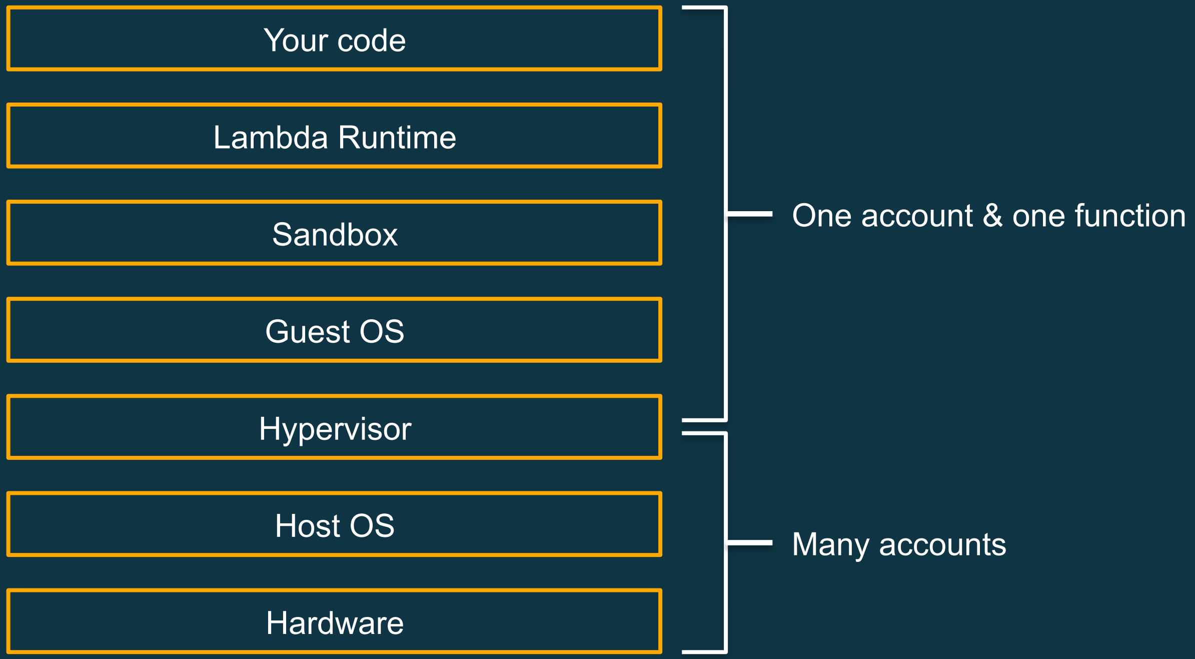
Host OS

EC2 Bare Metal

Hardware

Lambda isolation using Firecracker

Keeping workloads safe and separate



Your code

Lambda Runtime

Sandbox

Guest OS

Hypervisor

Host OS

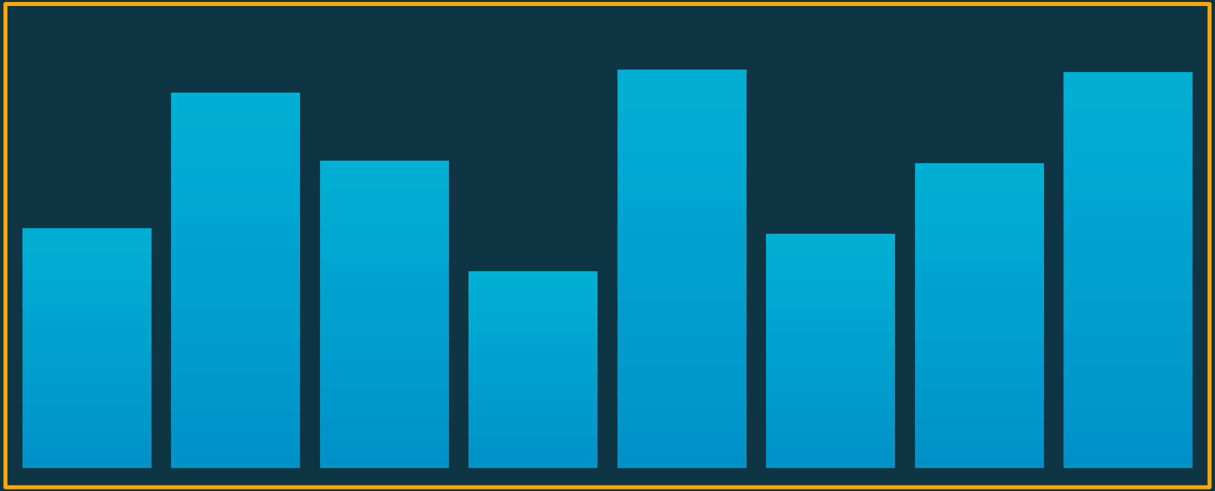
Hardware

One account & one function

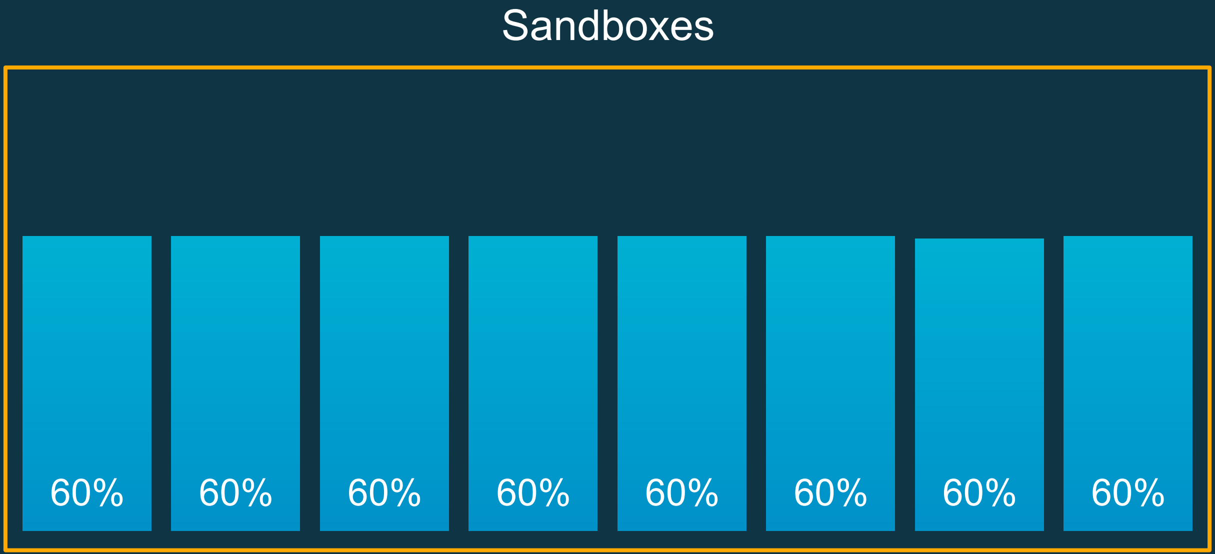
Many accounts

Available Sandboxes for a function

Sandboxes

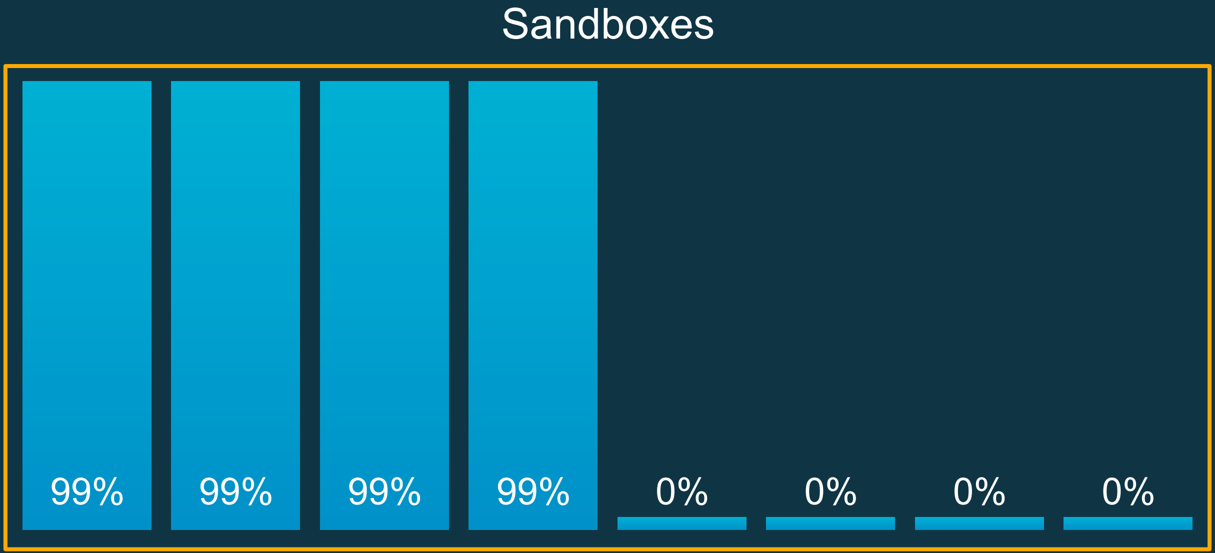


Load balancing before AWS Lambda



Load balancing with Lambda

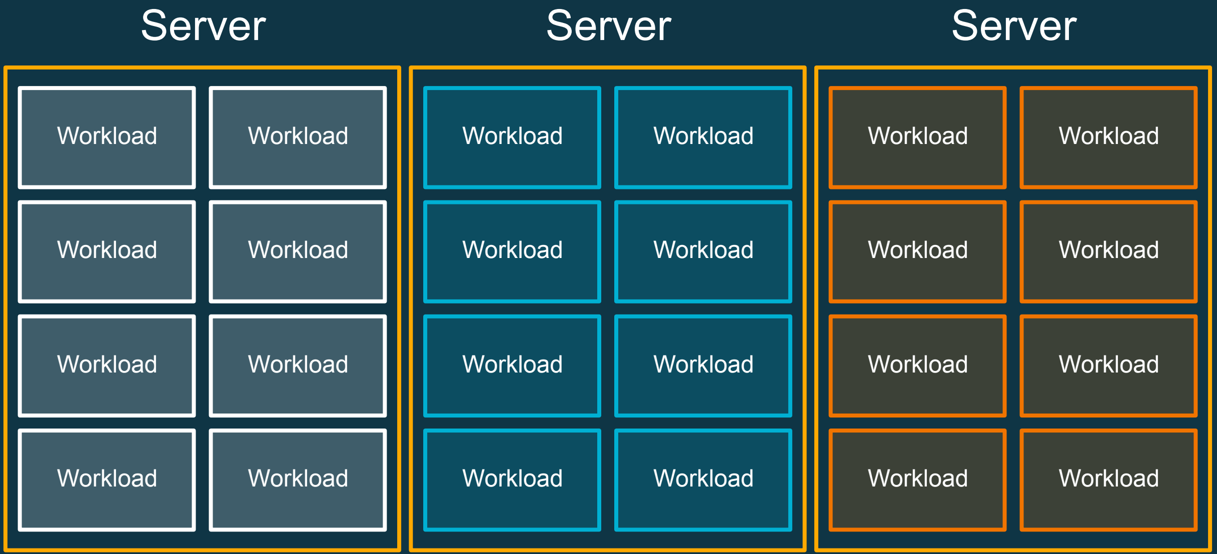
Concentrate the load



Cache locality

Ability to instantly scale

Allocate Workloads: pack server with one workload



Server

Workload Workload
Workload Workload
Workload Workload
Workload Workload

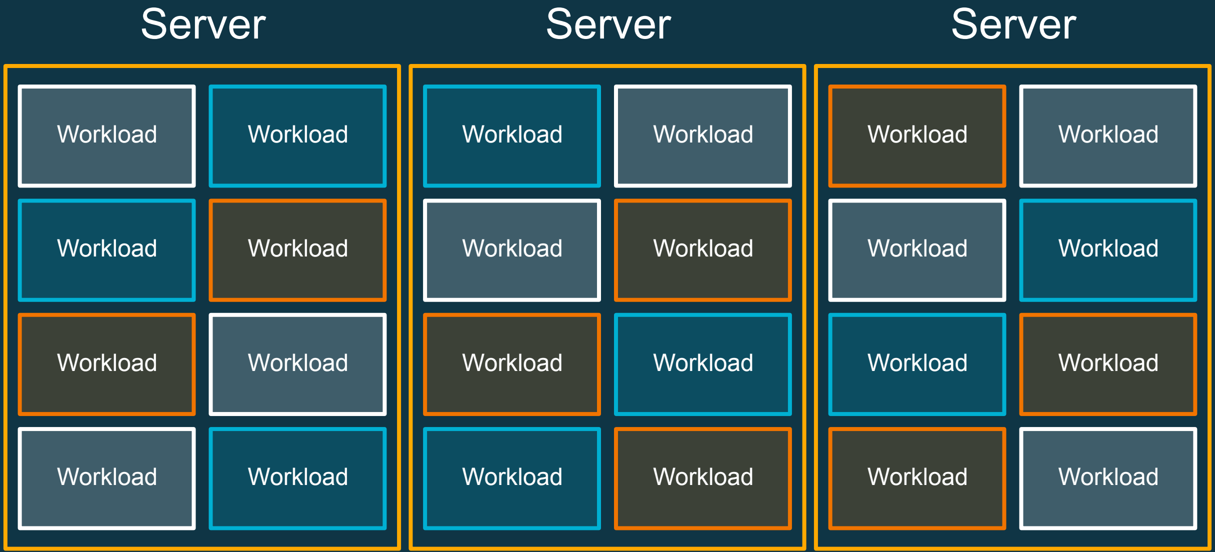
Server

Workload Workload
Workload Workload
Workload Workload
Workload Workload

Server

Workload Workload
Workload Workload
Workload Workload
Workload Workload

More efficient: pack server with many workloads



Server

Workload Workload
Workload Workload
Workload Workload
Workload Workload

Server

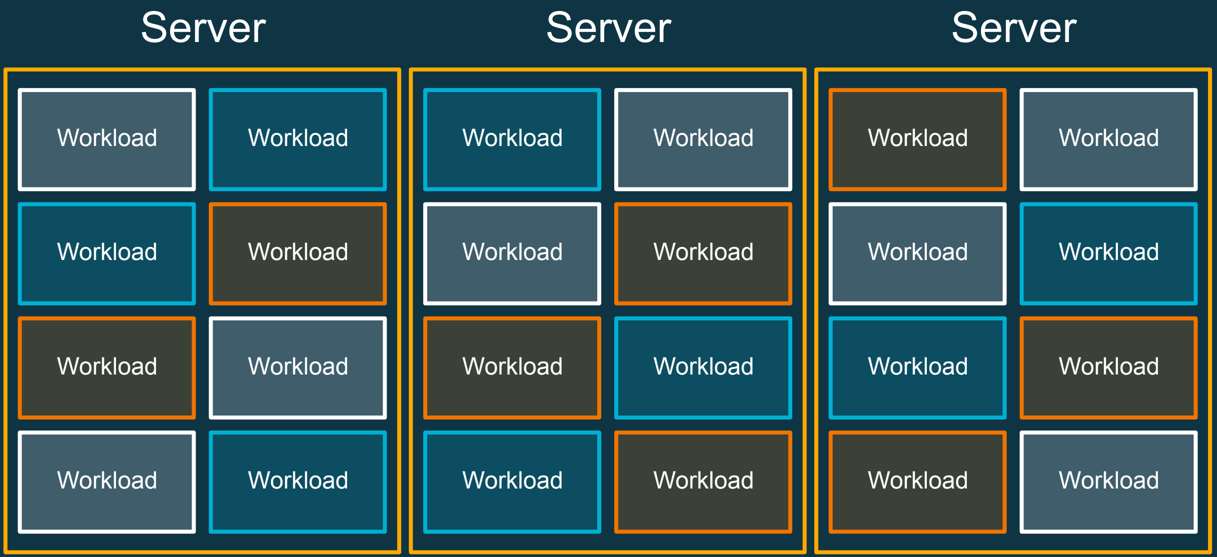
Workload Workload
Workload Workload
Workload Workload
Workload Workload

Server

Workload Workload
Workload Workload
Workload Workload
Workload Workload

Take advantage
of statistical multiplexing

Most efficient: placement optimization



Server		Server		Server	
Workload	Workload	Workload	Workload	Workload	Workload
Workload	Workload	Workload	Workload	Workload	Workload
Workload	Workload	Workload	Workload	Workload	Workload
Workload	Workload	Workload	Workload	Workload	Workload

Pick workloads that pack well together

Minimize contention

AWS Container Services landscape

Management

Deployment, scheduling, scaling & management of containerized applications

Amazon Elastic Container Service

Amazon Elastic Container Service for Kubernetes

Hosting

Where the containers run

Amazon EC2

AWS Fargate

Image Registry

Container image repository

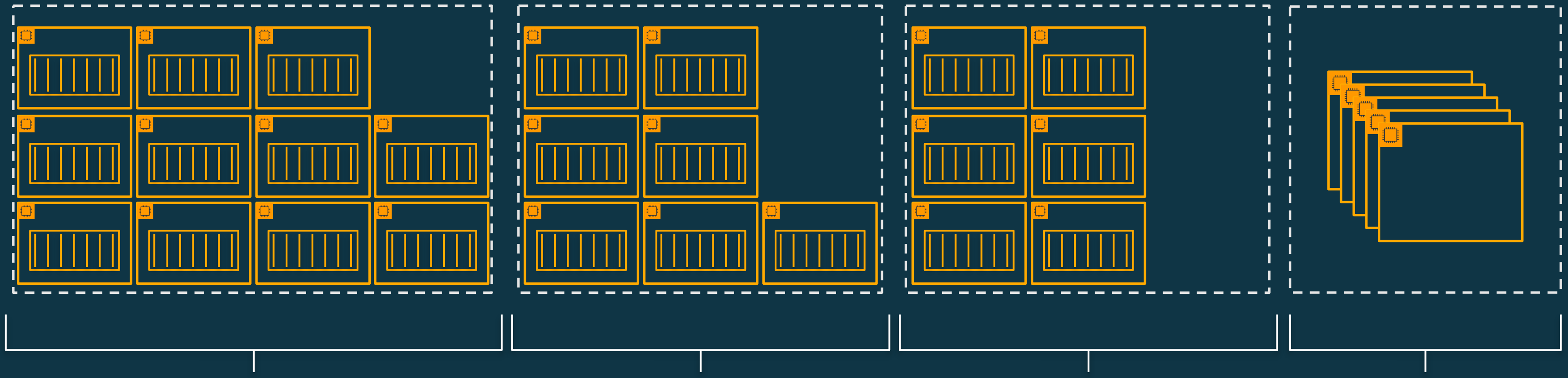
Amazon Elastic Container Registry

AWS Fargate

Fargate configurations

CPU (vCPU)	Memory Values (GB)
0.25	0.5, 1, 2
0.5	Min 1GB, max 4GB, in 1GB increments
1	Min 2GB, max 8GB, in 1GB increments
2	Min 4GB, max 16GB, in 1GB increments
4	Min 8GB, max 30GB, in 1GB increments

Fargate EC2 resource usage: with warm pool



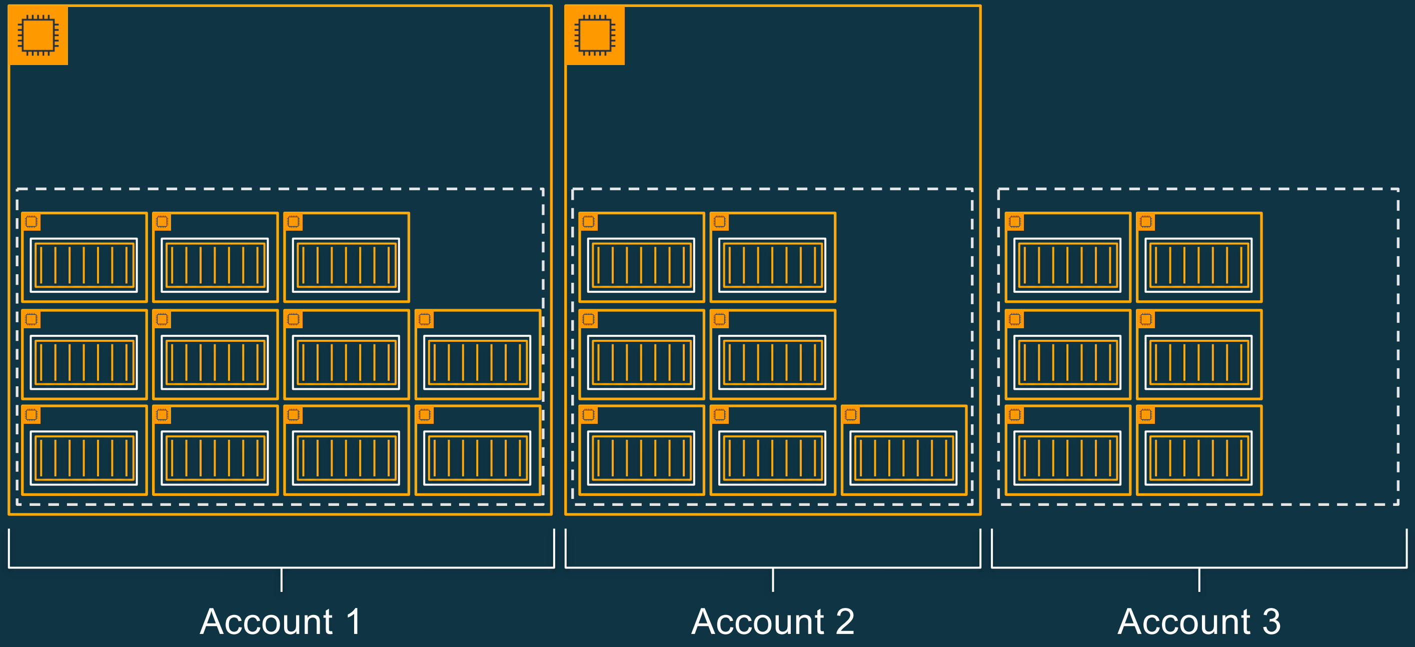
Account 1

Account 2

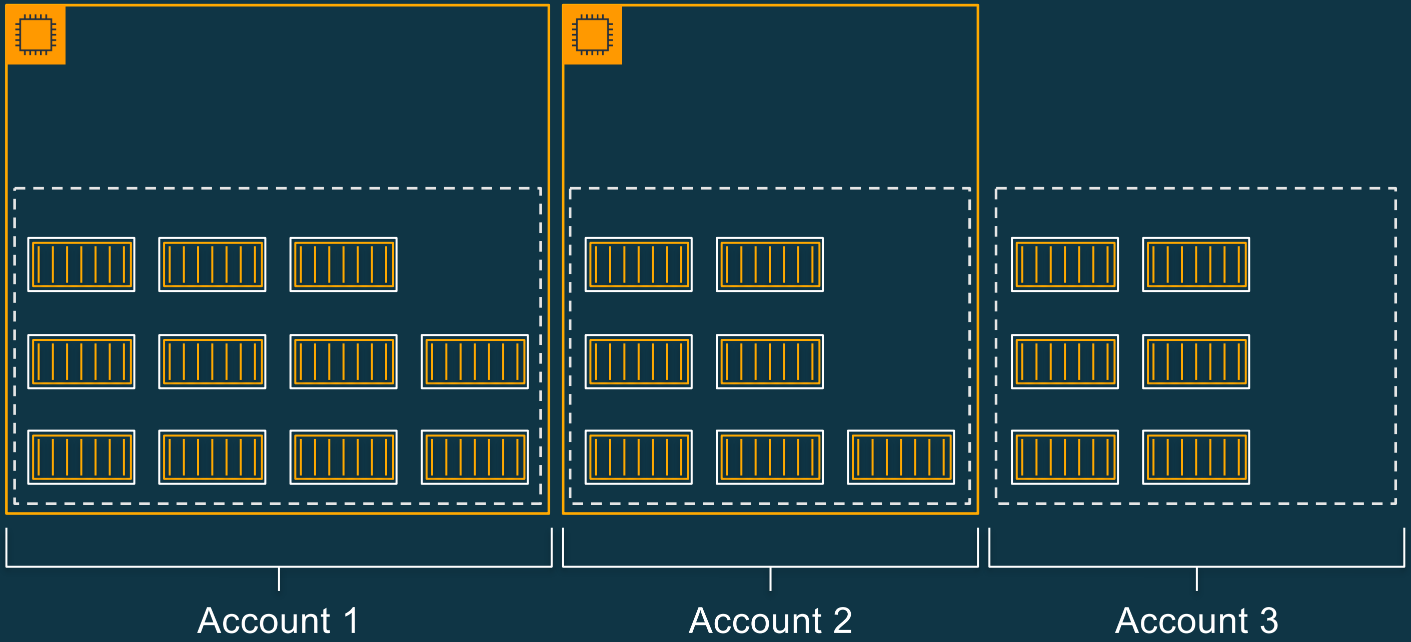
Account 3

Service Account

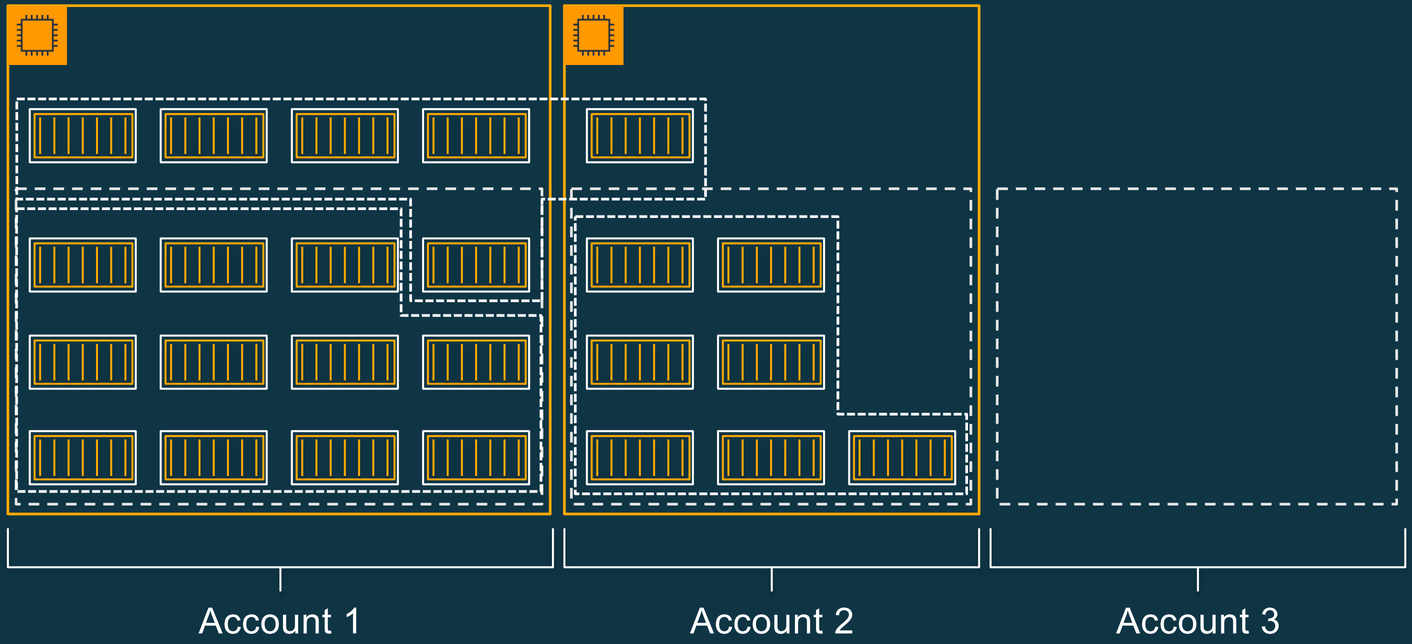
Fargate EC2 resource usage: with Firecracker



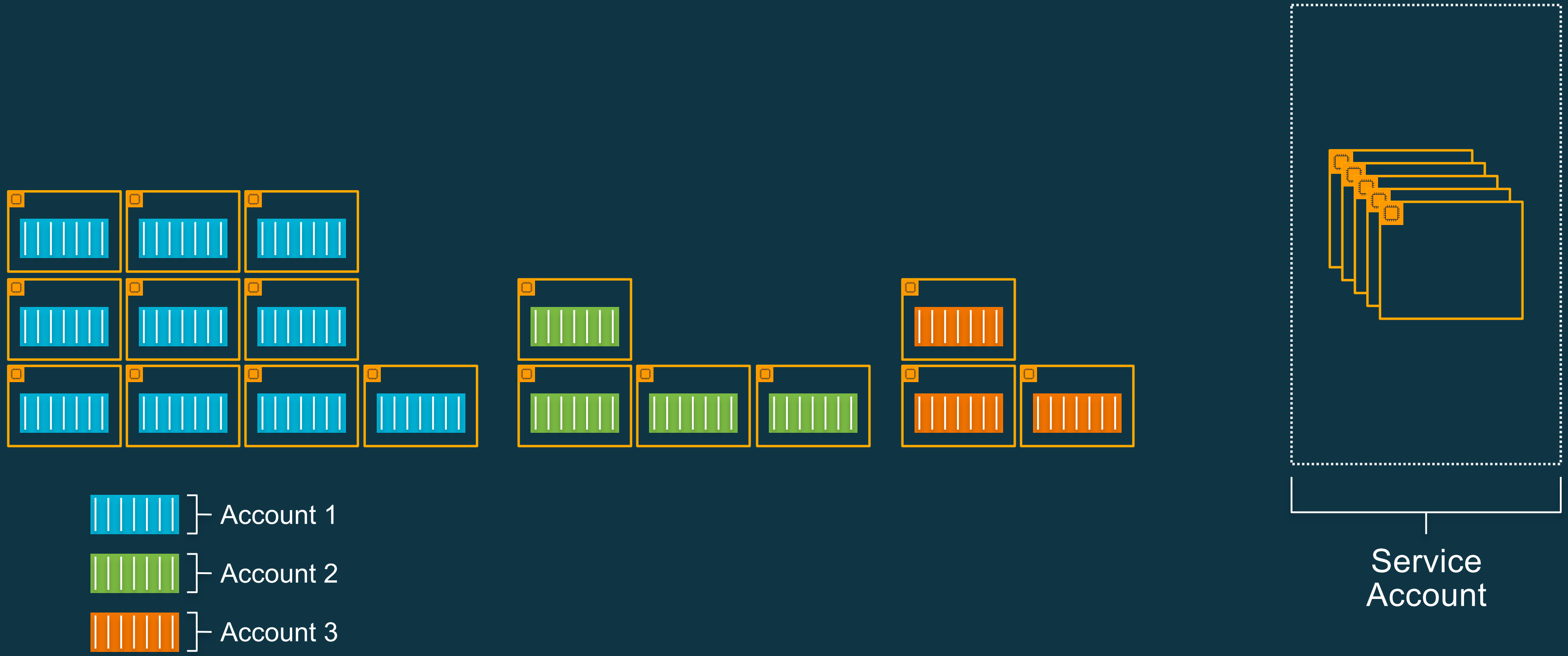
Fargate EC2 resource usage: with Firecracker



Fargate EC2 resource usage: with Firecracker



Fargate EC2 resource usage: with warm pool



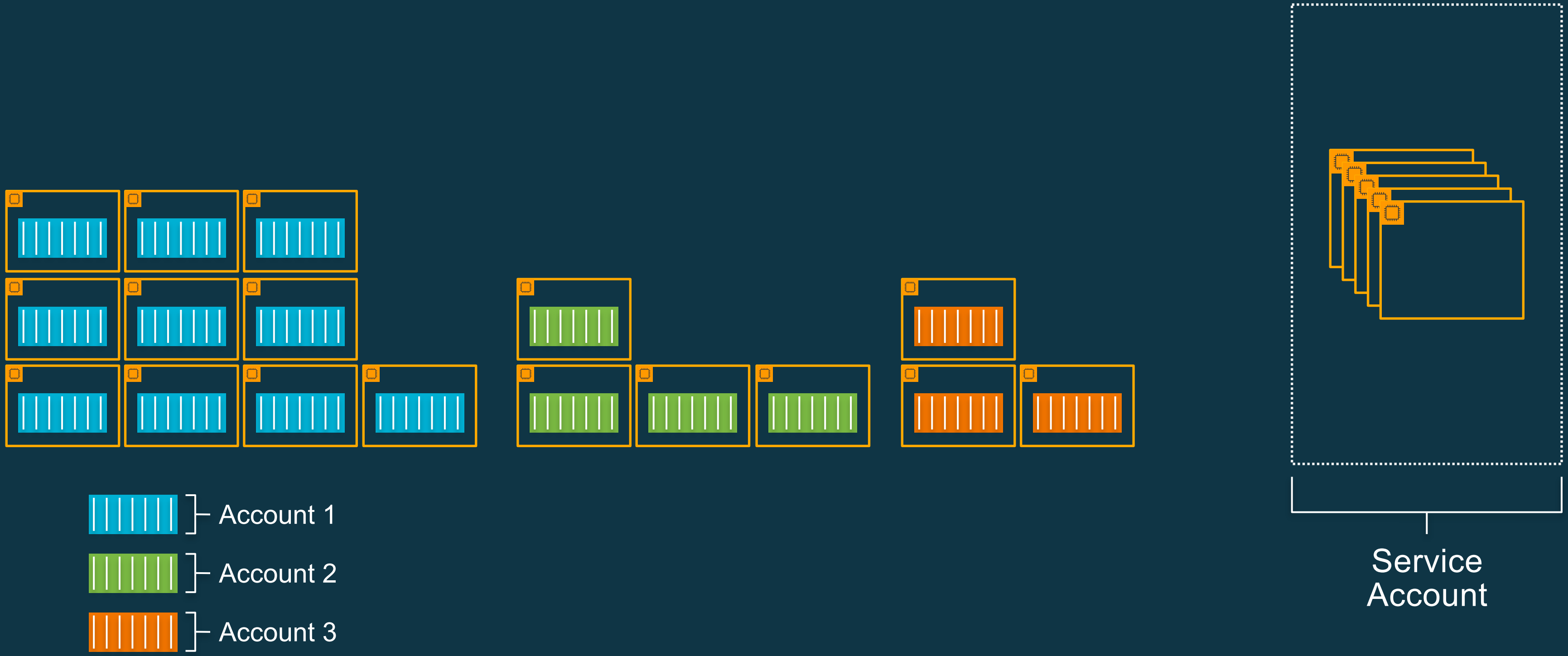
Account 1

Account 2

Account 3

Service Account

Fargate EC2 resource usage: with Firecracker



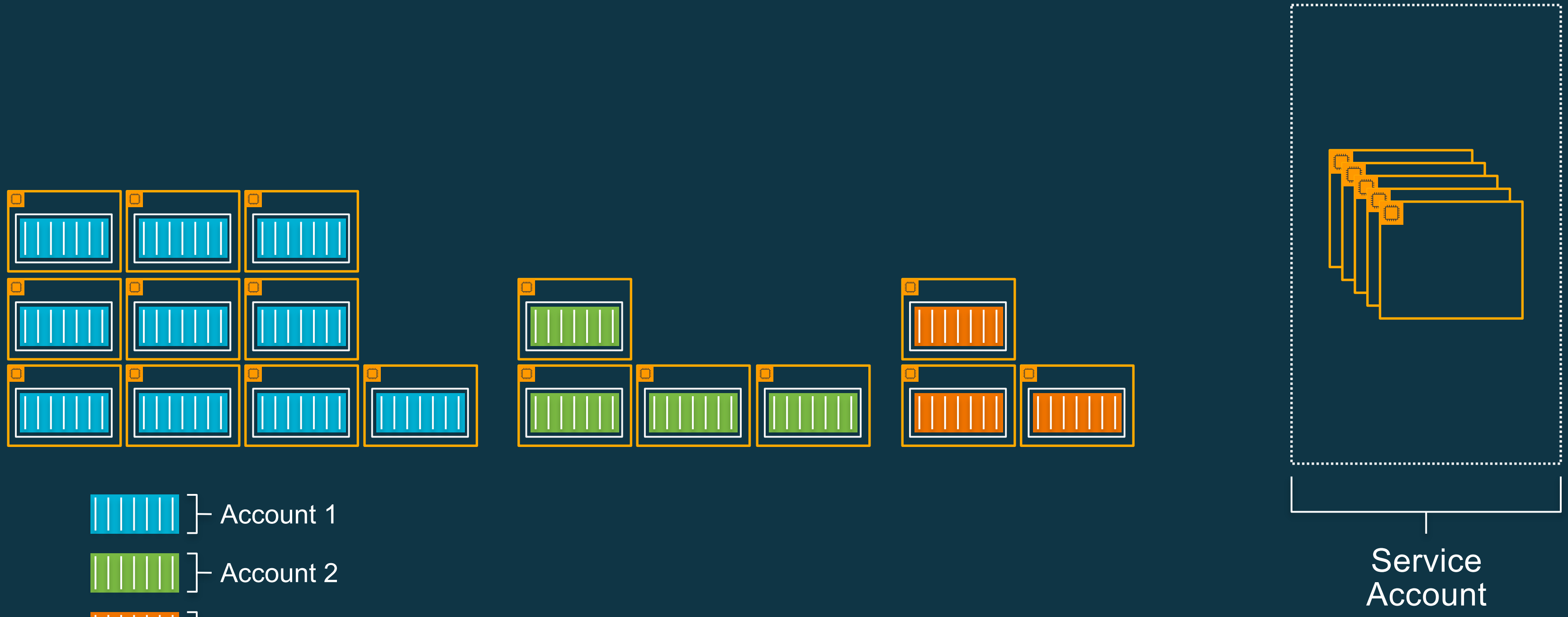
Account 1

Account 2

Account 3

Service Account

Fargate EC2 resource usage: with Firecracker



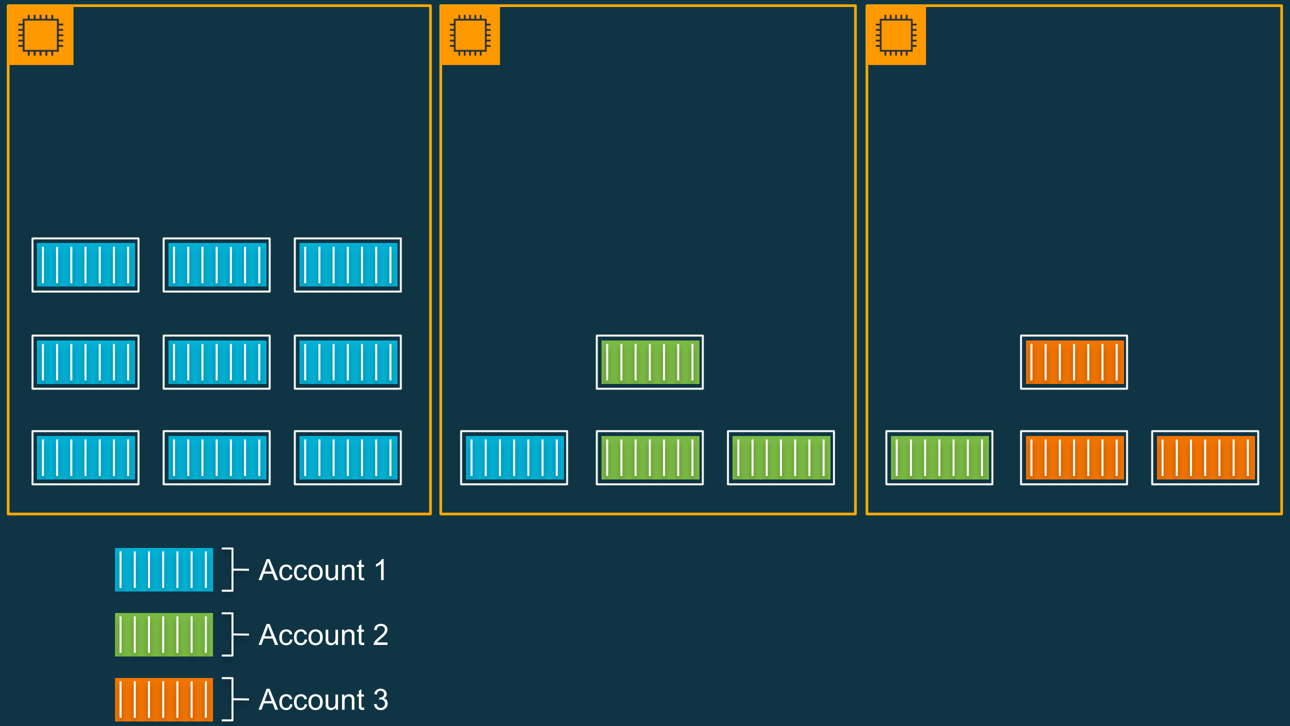
Account 1

Account 2

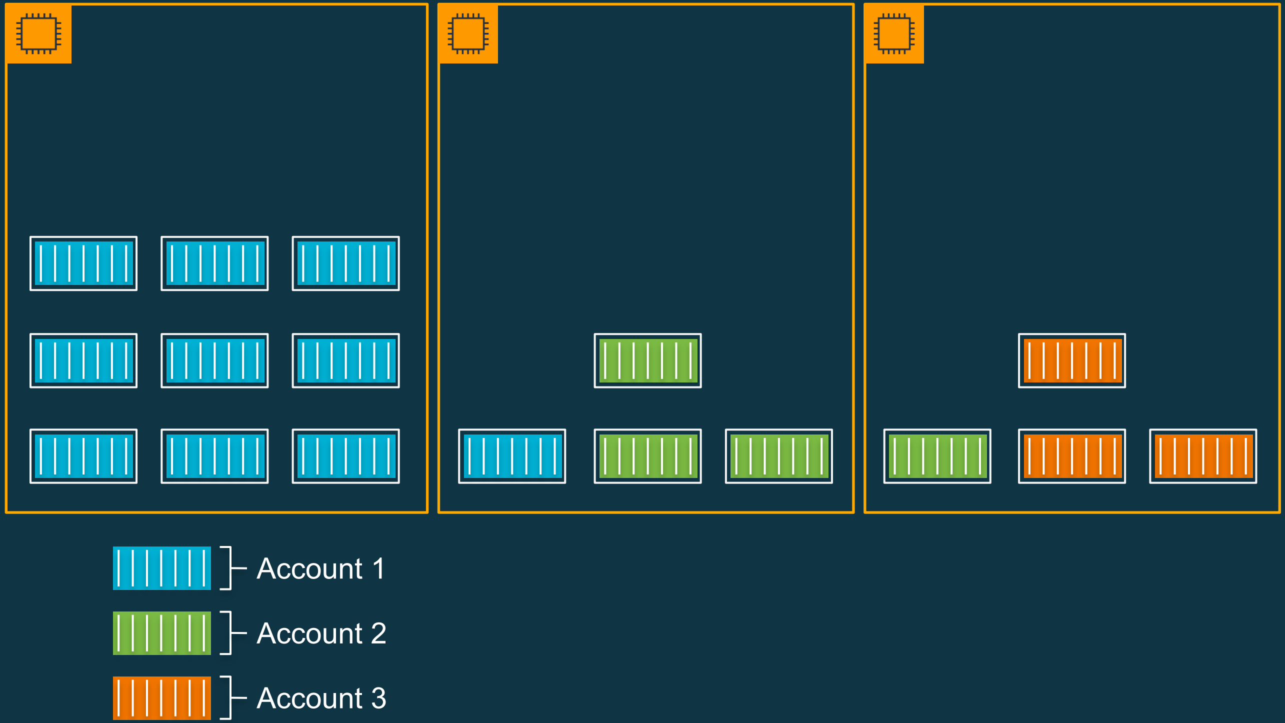
Account 3

Service Account

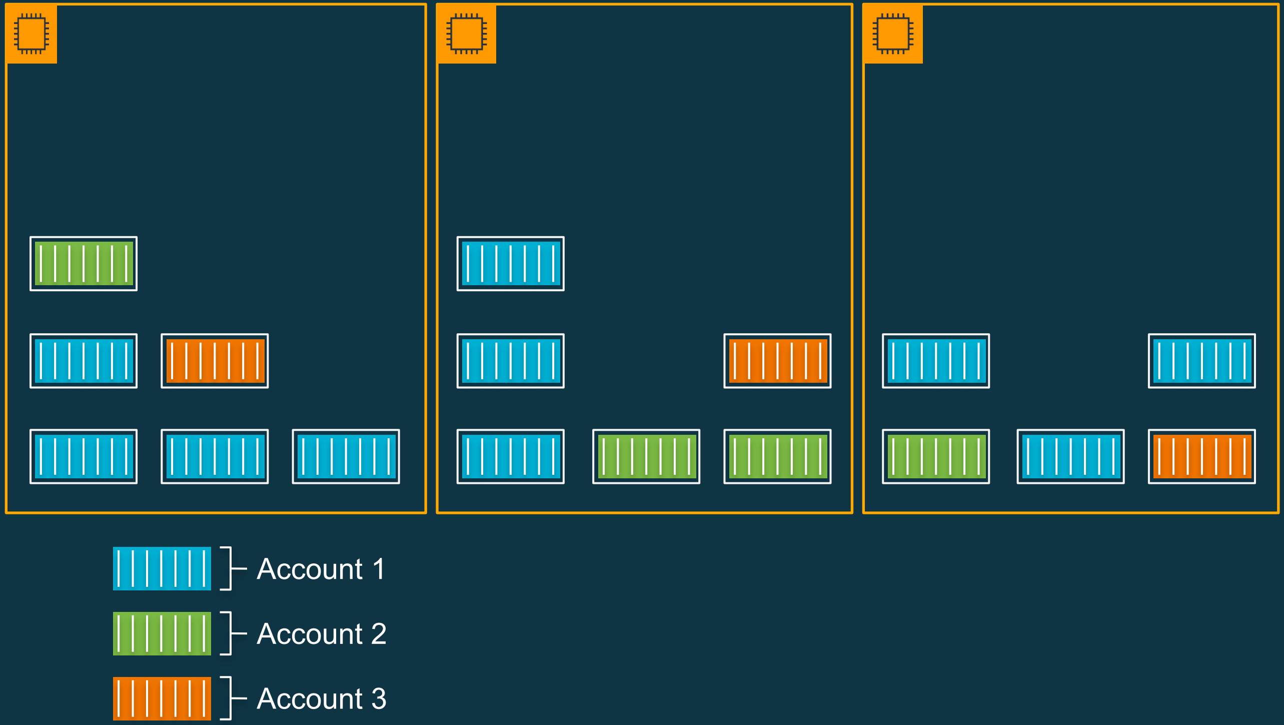
Fargate EC2 resource usage: with Firecracker



Fargate EC2 resource usage: with Firecracker



Fargate EC2 resource usage: with Firecracker



AWS Fargate price reduction

vCPU	GB Memory	Effective Price Cut
2	12	-47.00%
2	13	-47.90%
2	14	-48.60%
2	15	-49.30%
2	16	-50.00%
4	8	-35.00%
4	9	-36.20%
4	10	-37.30%
4	11	-38.30%

20% per vCPU per second

65% per GB per second

35%–50% cumulative

https://aws.amazon.com/blogs/compute/aws-fargate-price-reduction-up-to-50/

firecracker-containerd

Containerd to manage containers as Firecracker microVMs

Multi-tenant hosts

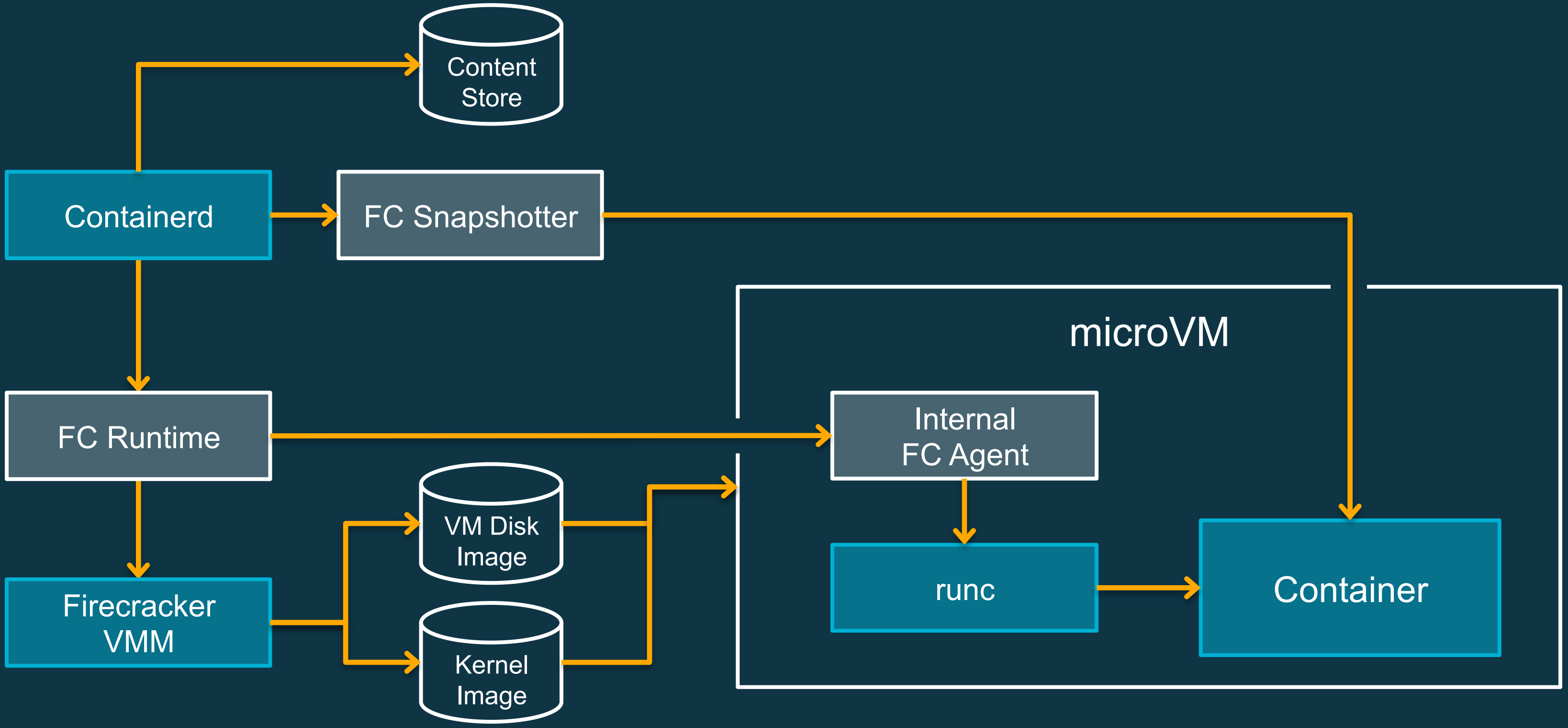
OCI image format

Work with popular orchestration frameworks

- Kubernetes and Amazon ECS

Define a future: light as container, secure as VM

Firecracker and containerd architecture



Firecracker as an open source project

88 contributions from 32 open source community contributors (~22%)

Dozens of community bug reports, 19 feature requests, RFC feedback

Talks at 12 industry conferences across 2019

rust-vmm, working with other industry players to build VMM crates.

Firecracker integration with open source projects

Kata Containers

UniK

OSv

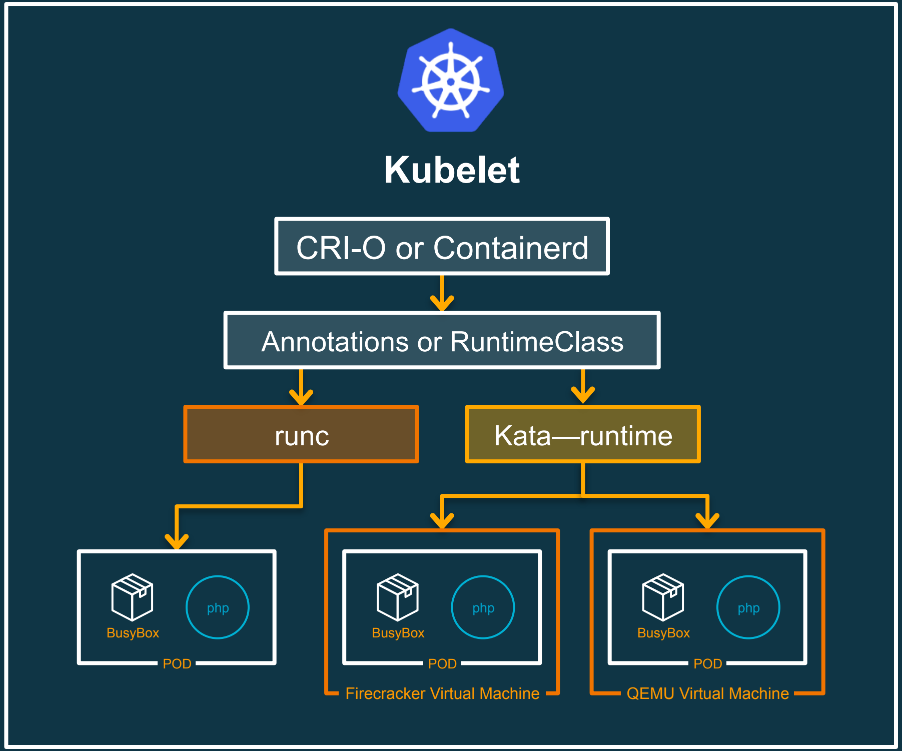
Firecracker and Kata Containers

Build lightweight virtual machine that seamlessly plugs into containers

Kata Containers supports multiple hypervisors

- Default QEMU
- Preliminary Firecracker support

Firecracker and Kata Containers



```
spec:
 template:
   spec:
     runtimeClassName: kata-fc
```

Firecracker and QEMU

Quick EMUlator

QEMU is a Type-2 hypervisor

Firecracker is cloud-native alternative to QEMU

Minimal device model

Firecracker: cloud-native alternative to QEMU

QEMU	Firecracker
Type-2 hypervisor	Type-2 hypervisor
Full-system emulation, including peripherals	Minimal device model: `virtio-net`, `virtio-block`, serial console, 1-button keyboard
Several instruction sets	Intel, AMD and Arm support in Alpha stage.
KVM- and Xen-hosting	KVM-hosting
Guest OS: Linux, Solaris, Windows, DOS and BSD	Guest OS: Linux
??	125ms startup, 5MB footprint

Who should use Firecracker directly?

Teams building compute services

Teams integrating Firecracker with container stacks

Developers & security engineers that want to contribute

Getting started with Firecracker

Firecracker on AWS bare metal

Firecracker on other clouds with bare metal (e.g., Packet)

Firecracker on GCP nested-virt

Firecracker on Azure nested-virt

Firecracker on your dev machine (physical/nested-virt)

Firectl

Firectl is a CLI to create Firecracker microVMs

```
firectl \
--kernel=hello-vmlinux.bin \
--root-drive=hello-rootfs.ext4
```

References and contribute

https://github.com/firecracker-microvm/

Thank you!