THE NIGHT SHIFT

Reclaiming Idle Infrastructure with 5-Spot, Confidential Containers, and DNS Automation

Erick Bourgeois · Director, Head of Kubernetes Platform Engineering

RBC Capital Markets · FINOS Open Source in Finance Forum

Erick Bourgeois

Head of Kubernetes Platform Engineering — RBC Capital Markets

Creator of 5-Spot — on-premises spot Kubernetes scheduler
Reclaiming idle bare-metal with ephemeral CAPI clusters

Creator & maintainer — firestoned/bindy
Rust · kube-rs · BIND9 Kubernetes operator for DNS lifecycle automation

KubeCon & FluxCon speaker · FINOS OSFF panelist
FluxCon keynote confirmed · KubeCon Japan 2025

24+ years in financial services infrastructure & platform engineering

Proudly Canadian — based in Montréal, QC

LinkedIn GitHub

Disclaimer: the information and opinions expressed here are my own and do not represent the views or positions of RBC Capital Markets or any of its affiliates.

Before We Start — Quick Hands Up

Who runs their own “on-premise” datacenter?

Who has servers sitting idle for more than 8 hours a night?

Who's done something useful with that idle capacity?

(If you answered yes to #2 but no to #3 — this talk is for you.)

2:00 AM — The Datacenter Never Sleeps

~70%

average server utilization
at peak business hours

Source: Gartner / Uptime Institute

---

drops to < 10% overnight

Power bills keep running

Servers draw ~60-70% idle power

Cooling keeps running

CRAC units don't take breaks

CapEx is sunk

That iron was already bought

Batch jobs only scratch the surface

Not elastic, not isolated

The Idea: Turn Idle Iron into Spot Compute

You know spot instances on public cloud?

We built the same model — on your own bare metal, with 5-Spot.

01 Donate

Teams flag machines as spot-eligible. Each server carries its own availability window — schedules are per-resource, not global.

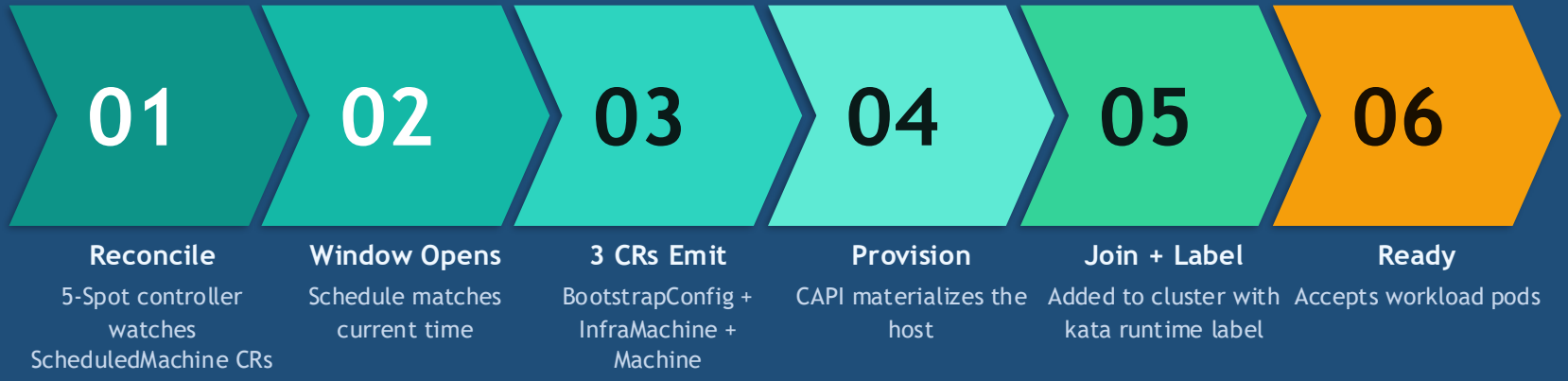
02 Reclaim

5-Spot activates when a window opens. It provisions an ephemeral cluster via CAPI — emitting a bootstrapRef and InfrastructureRef per node.

03 Return

At end of window, workloads drain and clusters dissolve. Servers return clean — ready for their daytime owners.

5-Spot Workflow — From Schedule to Ready Node



Step 03 detail — 5-Spot emits three Cluster API resources: an implementation of `bootstrap.cluster.x-k8s.io`, an `infrastructure.cluster.x-k8s.io`, and a `Machine.cluster.x-k8s.io` that binds them.

Event-driven · Per-resource scheduling · No persistent state between windows

Example Manifest — 5-Spot ScheduledMachine

```yaml
# Example ScheduledMachine resource
# This machine will be active Monday-Friday, 9am-5pm Eastern Time
# The controller will create:
#   1. A K0sWorkerConfig bootstrap resource
#   2. A RemoteMachine infrastructure resource
#   3. A CAPI Machine referencing both

apiVersion: 5spot.io/v1alpha1
kind: ScheduledMachine
metadata:
  name: business-hours-worker
  namespace: default
spec:
  # Name of the CAPI Cluster this machine will join
  clusterName: my-k0s-cluster

  # Schedule configuration (using day/hour format)
  schedule:
    daysOfWeek:
      - mon-fri
    hoursOfDay:
      - "9-17"
    timezone: America/New_York
    enabled: true

  # Inline bootstrap specification (creates K0sWorkerConfig)
  # Name is auto-generated as: business-hours-worker-bootstrap
  bootstrapSpec:
    apiVersion: bootstrap.cluster.x-k8s.io/v1beta1
    kind: K0sWorkerConfig
    spec:
      version: v1.30.0+k0s.0
      # Add any K0sWorkerConfig-specific fields here
```

```yaml
  # Inline infrastructure specification (creates RemoteMachine)
  # Name is auto-generated as: business-hours-worker-infra
  infrastructureSpec:
    apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
    kind: RemoteMachine
    spec:
      address: 192.168.1.100
      port: 22
      user: admin
      # Add any RemoteMachine-specific fields here

  # Optional: Labels and annotations for the created CAPI Machine
  machineTemplate:
    labels:
      node-role.kubernetes.io/worker: ""
      topology.kubernetes.io/zone: zone-a
    annotations:
      5spot.io/scheduled-by: business-hours-worker

  # Lifecycle settings
  priority: 50
  gracefulShutdownTimeout: 5m
  nodeDrainTimeout: 5m
  killSwitch: false
```

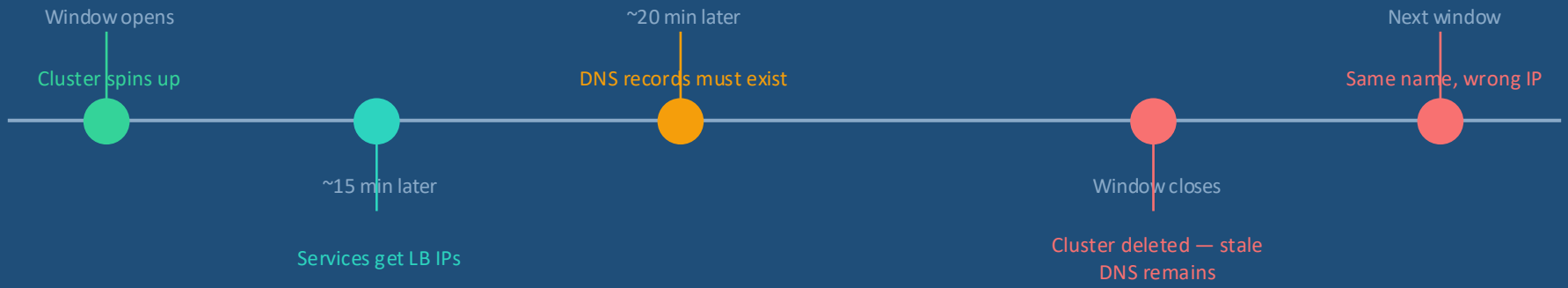
The Security Problem — Hardware You Don't Permanently Own

⚠ Trust problem

- Spot servers are borrowed from other teams — you don't control their firmware or BMC
- Multi-tenant workloads from different business lines share the same physical iron
- A regulated bank cannot rely on hypervisor-level isolation alone
- Attestation matters: how do you prove the execution environment is clean?

The Ephemeral DNS Problem

> *Every time a spot cluster spins up, its services need DNS records.*
> *Every time it dies, those records need to disappear.*
> *Manually — at 2am — is not an option.*



You need DNS that understands Kubernetes cluster lifecycle.

Bindy — DNS Lifecycle Automation for Ephemeral Infrastructure

What is Bindy?

Written in Rust using kube-rs — fast, memory-safe, production-grade

Kubernetes operator for BIND9 — manages DNS records via RFC 2136 dynamic updates

8 strongly-typed CRDs: A, AAAA, CNAME, MX, NS, PTR, SRV, TXT records

Watches cluster resources — auto-creates and deletes DNS records as services appear and vanish

TSIG authentication to BIND9 — no open dynamic update, no spoofing

```yaml
# spot-cluster-dns.yaml
apiVersion: bindy.firestoned.io/v1beta1
kind: ARecord
metadata:
  name: www-example
labels:
    # Used by DNSZone selector
    zone: example.com
spec:
  name: spot-01
  ipv4Addresses:
    - "192.0.2.1"
  ttl: 300
```

github.com/firestoned/bindy

Summary — One Spot Cluster, End to End

Orchestrated by 5-Spot • 4-layer runtime • 3 supporting pillars • 100% open source

THE RUNTIME STACK *what 5-Spot builds, top-down*

01 **Workload: Kata + Confidential Containers**
Each pod in its own VM kernel · hardware-isolated · TEE attestation

02 **Orchestration: k0s**
Single-binary Kubernetes workers · hosted control plane · clean teardown

03 **Host OS: enterprise Linux (kernels 4.18 → 5.14)**
OS-agnostic · zero mandate · runs on whatever the donor team ships

04 **Physical: donated bare-metal**
Availability windows per machine · overnight / weekend = spot capacity

SUPPORTING PILLARS

DNS
Bindy + BIND9
TTL=30 · per-cluster zones · zonewarden sync

Security
SPIRE + TEE attestation
TSIG · mTLS · per-workload identity

GitOps
FluxCD
All state in git · declarative by default

Results & Key Takeaways

~65%
idle capacity reclaimed per window

< 30s
DNS convergence on cluster spin-up

< 15 min
full cluster provision to ready

Take home:

- Your own datacenter can run a spot market — 5-Spot, CAPI, Kata, and CoCo make it real today
- Ephemeral infrastructure demands ephemeral DNS — automate it or suffer at 2am
- Immutable OS + separated control planes = clean lifecycle, every time
- In regulated environments, workload isolation at the kernel level is not paranoia — it's the baseline
- Open source (bindy, Kairos, k0rdent) makes this possible without vendor lock-in at every layer

Thank You

Open source is a team sport. Thank you all for being here at FINOS.

My Team at RBC Capital Markets

10 engineers who ship real infrastructure every day

This work exists because of our curiosity, your craftsmanship, and your willingness to build things that have never been built before.

Thank you for making this possible.

William Rizzo

KubeCon co-presenter · good friend

A collaborator who has pushed this work in unexpected directions — from confidential computing to rethinking what spot really means at the infrastructure layer.

Thank you for the conversations, the shared stage, and challenging me to think bigger.

The night shift never ends.

Put it to work.

5-Spot	*github.com/finos/5-spot*
bindy	*github.com/firestoned/bindy*
Kairos	*kairos.io*
k0rdent	*k0rdent.io*
LinkedIn	*linkedin.com/in/erick-bourgeois*

Erick Bourgeois · Director, Kubernetes Platform Engineering · RBC Capital Markets