Fleet Operations

Changing the Engine in Mid-Air

Jon Daniel, Lead Infrastructure Engineer
jon.daniel@salesforce.com | me@jondaniel.email
They/Them

Thank you

Jon Daniel

Lead Infrastructure Engineer
@binarycleric on X

salesforce

HEROKU DATA

Heroku Data operates over 50,000 EC2 Instances running over 250,000 PostgreSQL databases.

Heroku Data is a geographically distributed team (EU, UK, US).

HEROKU

salesforce

Take a journey with me.

All code samples are fictional and dramatically simplified.

You’re on-call at Heroku and you’ve got the whole Data fleet to maintain.

And your name is…

Codey. Software Engineer.

he/

Astro. Software Engineer.

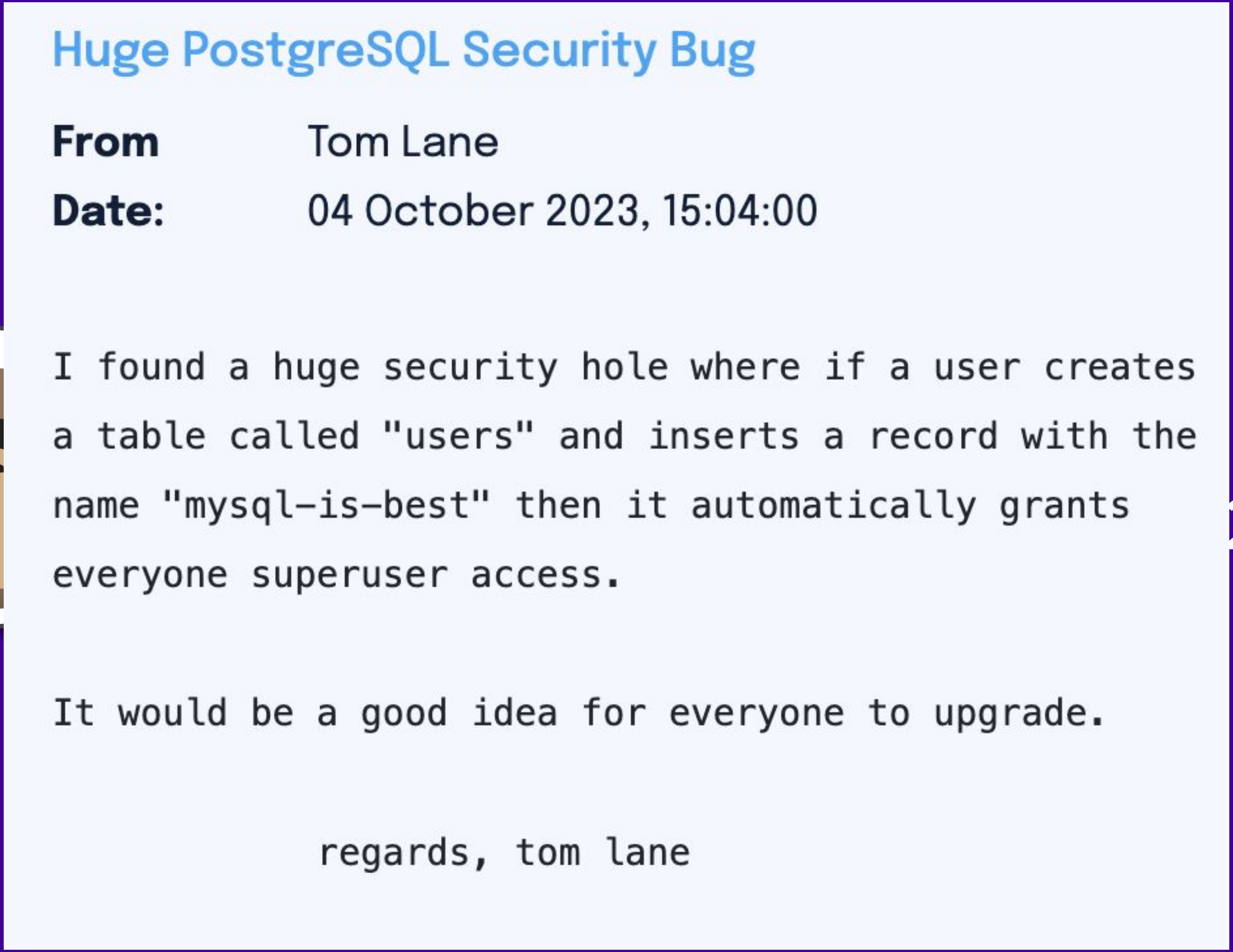
they/them

Hey buddy. Have you checked the postgres mailing list today?

No... What’s going on?

It’s bad. 😔

postgres.security/CVE-2X23-0623



Huge PostgreSQL Security Bug

From Tom Lane

Date: 04 October 2023, 15:04:00

```
I found a huge security hole where if a user creates
a table called "users" and inserts a record with the
name "mysql-is-best" then it automatically grants
everyone superuser access.

It would be a good idea for everyone to upgrade.

            regards, tom lane
```

23

Maybe this is a good time to take that camping trip we always talked about...

od time
ing trip
about…

Cloudy. Security Engineer.

she/her

I’m sure you saw the new PostgreSQL CVE. Security wants it pathed by the end of the day.

Do it by hand if necessary.
Just get it done!

We have 50,000 EC2 instances. I can't do those by hand.

Plus this upgrade causes customer downtime.

We have to respect maintenance windows.

Wow, that’s certainly way too many do to by hand.

I’ll see what I can do. Let me talk to my manager…

Codey anxiously stares at Slack for a little bit too long.

Cloudy is typing...

Cloudy is typing...

Cloudy is still typing...

Security can give you a 7 day extension but no more than that.

Please let me know if you have questions or need me to pass anything along.

That sucks though. Let me know if you want to vent over coffee sometime.

How much do I really care about this job?

Maybe I'll just live in the woods forever.

No, I’m a professional and my customers depend on me.

Codeying Intensifies!!!

S
99

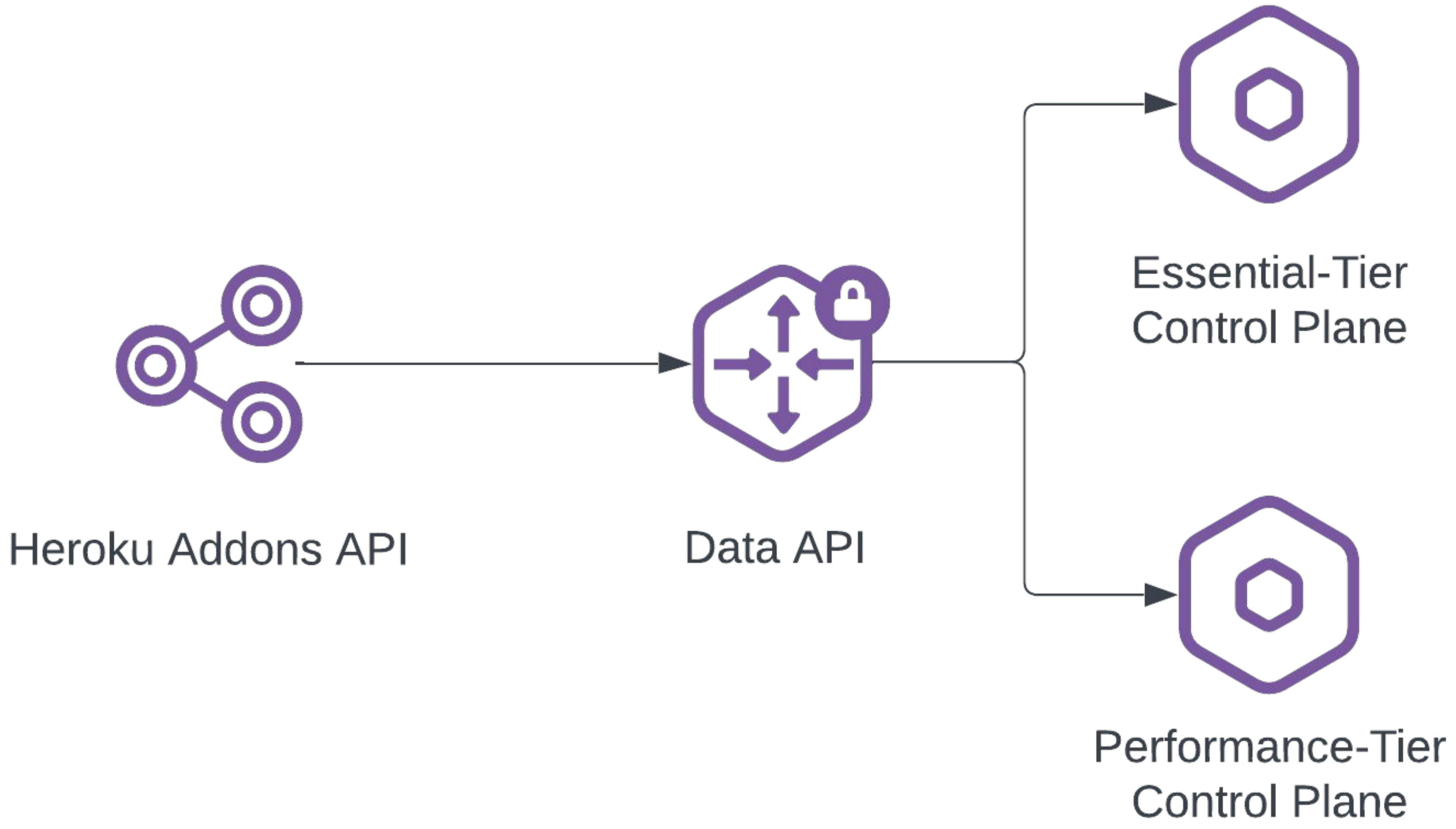
Don’t fret buddy, you don’t have to work on this alone.

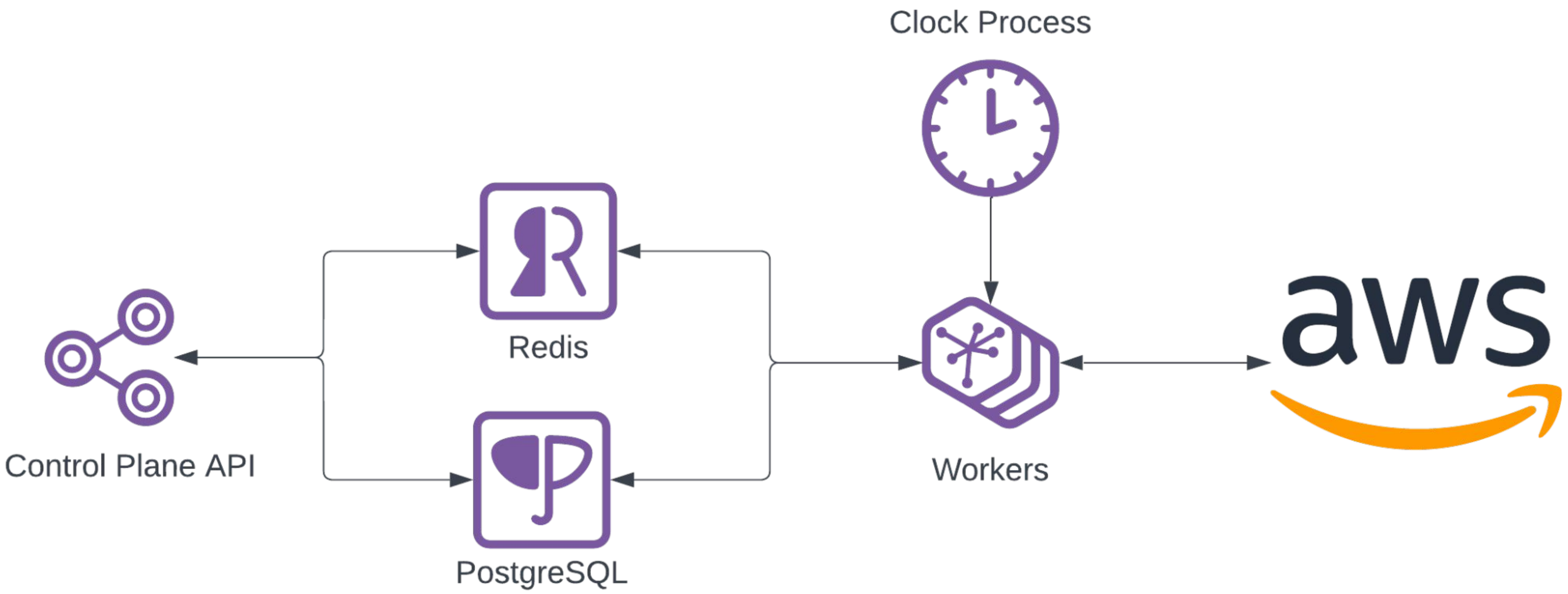
Let’s go over the basics first and work towards a solution.

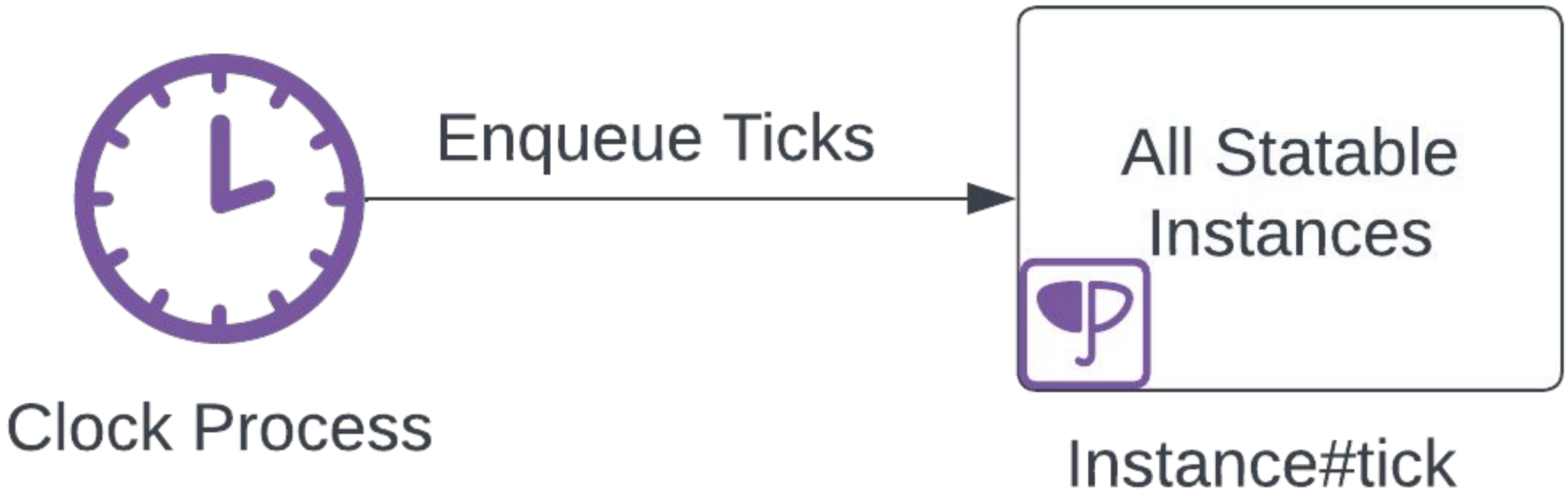
Think about how a Heroku Addon is created and we'll walk through the rest.

Then we can schedule maintenances to get those databases up-to-date!

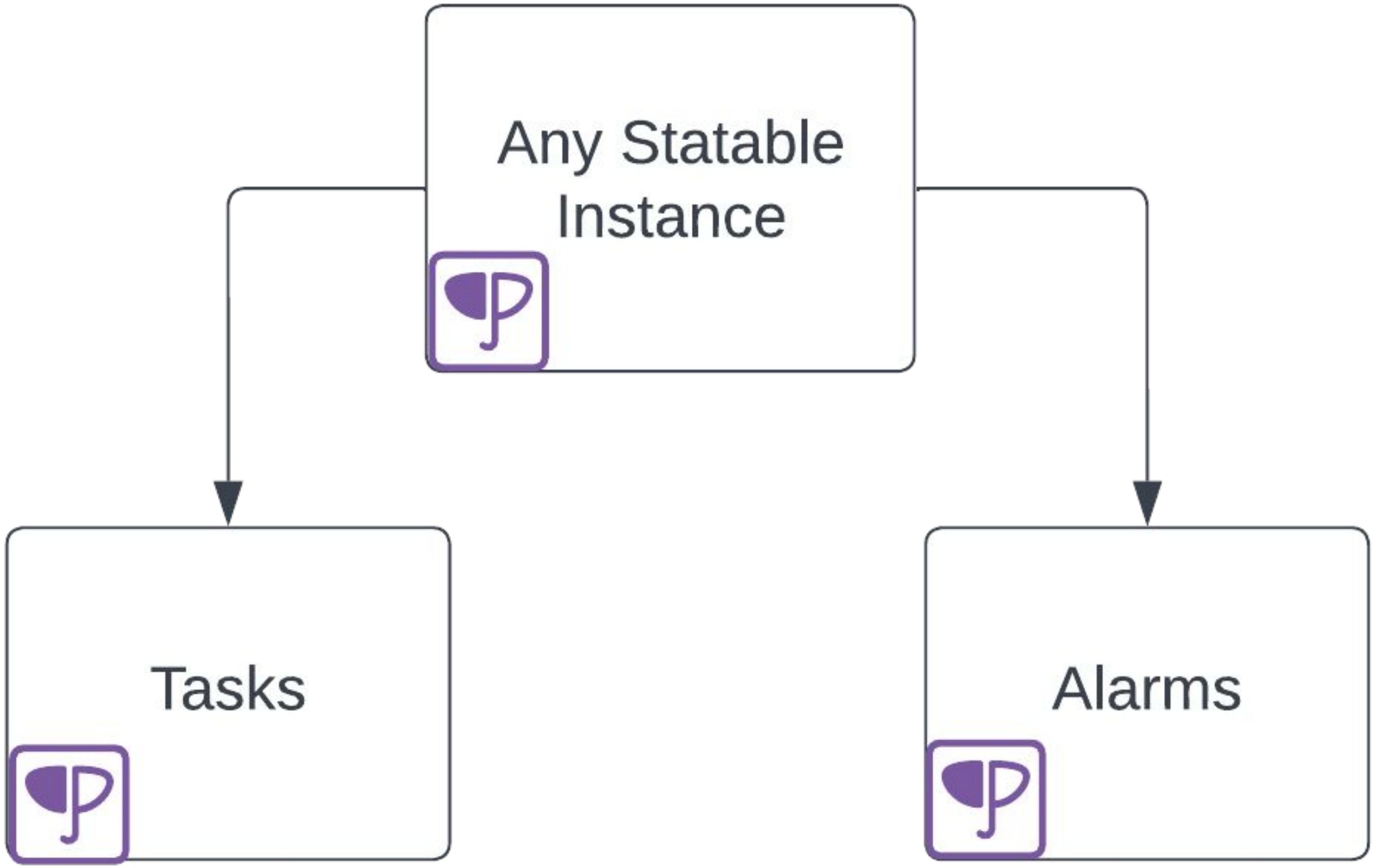
```
~/src/my-cool-app
➜ heroku addons:create \
    heroku-postgresql:standard-0
```


Heroku Addons API
Data API
Essential-Tier Control Plane
Performance-Tier Control Plane


Clock Process
Redis
Control Plane API
Workers
aws
PostgreSQL


Enqueue Ticks
All Statable Instances
Clock Process
Instance#tick

What does it mean to be a statable instance? Like a Finite State Machine?


Any Statable Instance
Tasks
Alarms

```ruby
class Server
  def after_create
    task_control(:provision).start
    transition "booting"
  end

  state "booting" do
    transition "running" if task_control(:provision).done?
  end

  state "running" do
    transition "uncertain" if !observation.pingable?
  end

  state "uncertain" do
    transition "running" if observation.pingable?
  end
end
```

```ruby
class Server
  task(:provision) do
    state "create_ec2_instance" do
      parent.ec2_instance = Ec2Instance.create
      parent.save_changes

      transition "wait_for_instance"
    end

    state "wait_for_instance" do
      if parent.ec2_instance.running?
        transition "create_dns_record"
      end
    end
  end
end
```

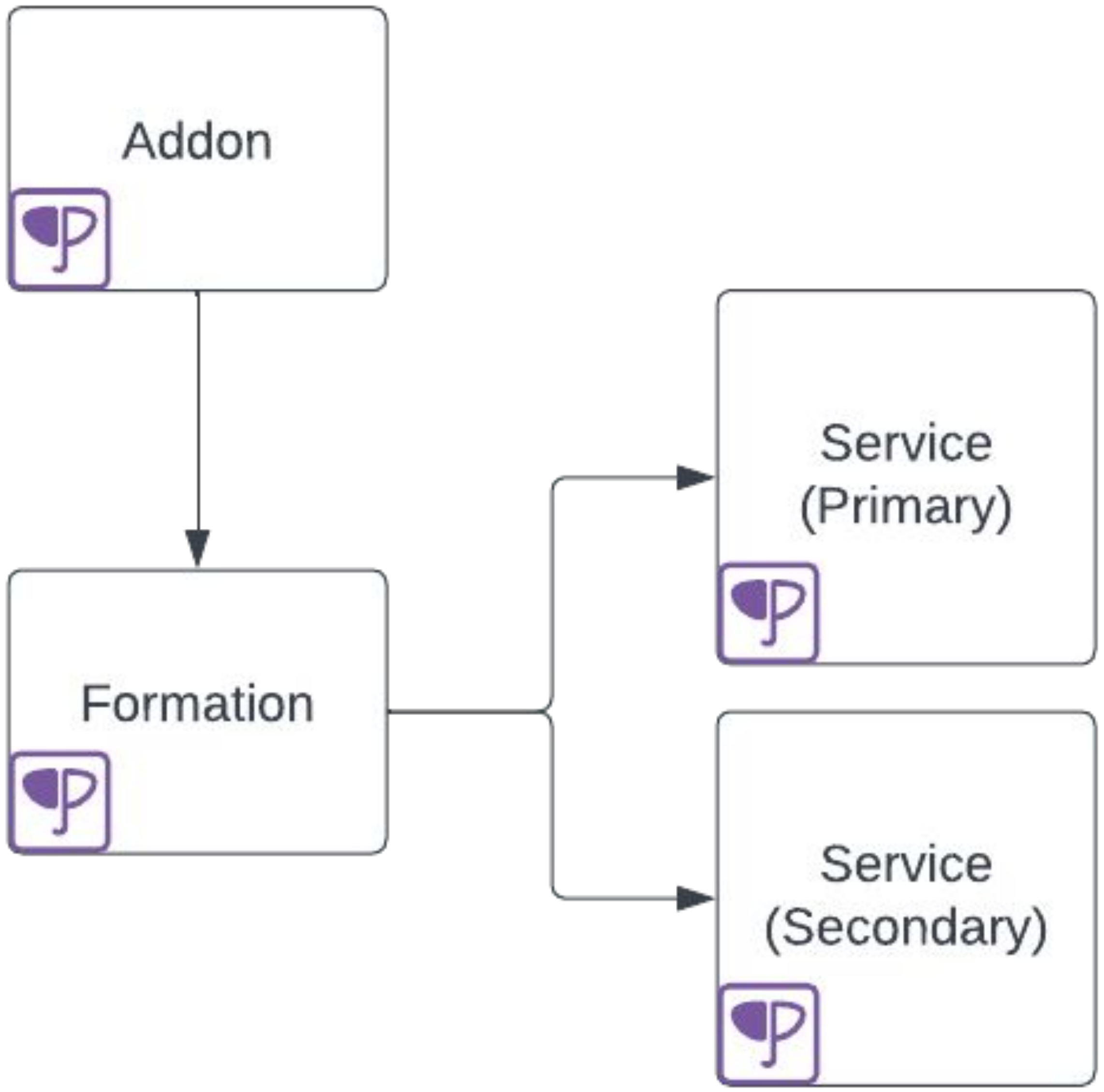
```ruby
alarm(:server_down) do
  panic_after 5.minutes
  on_panic :page_operator

  def self.on_start(server)
    server.reboot_ec2_instance
  end

  def self.should_start?(server)
    !server.observation.pingable? &&
      server.last_successful_observation <= 5.minutes.ago
  end
end
```

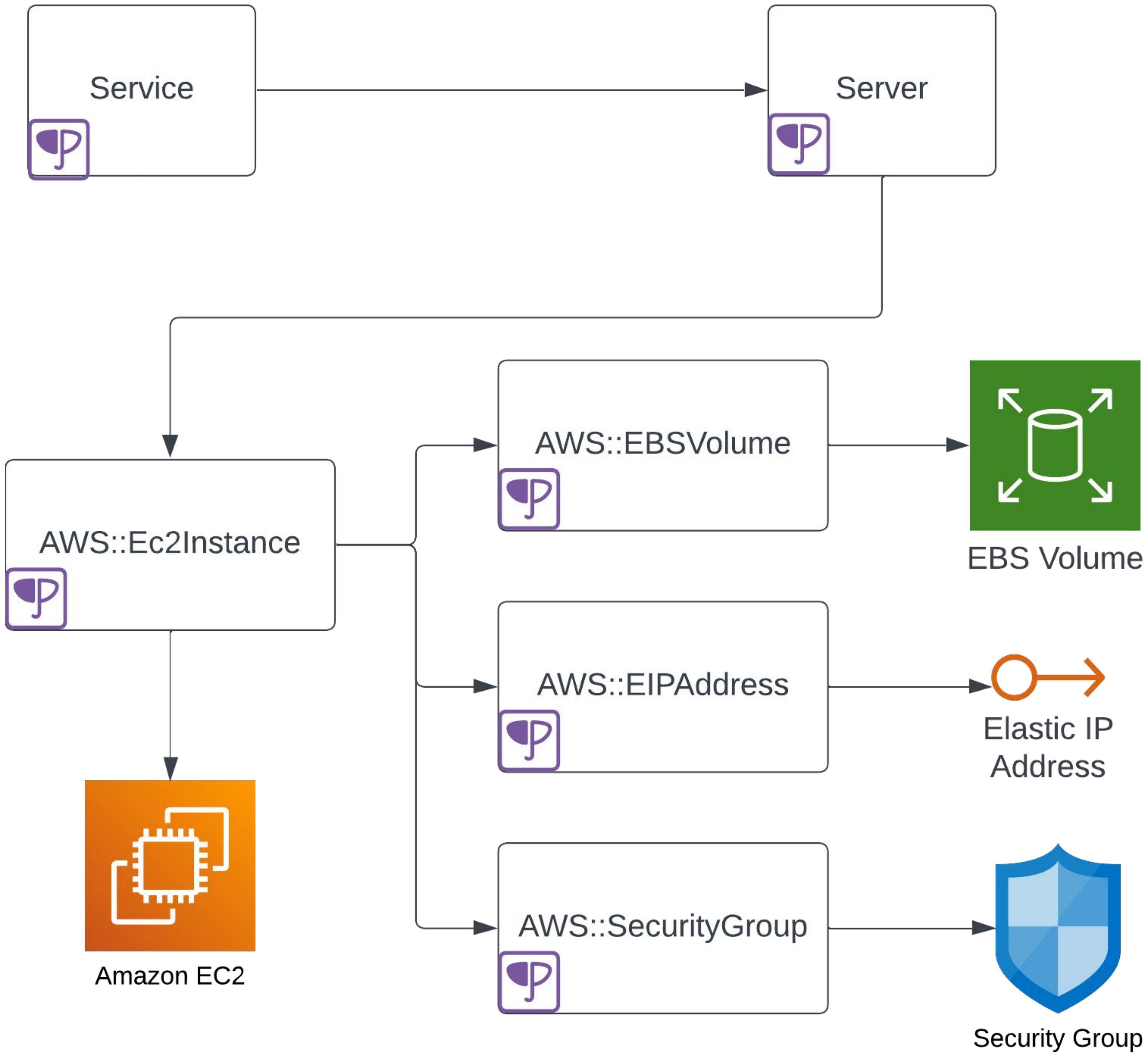
So tasks run operations and alarms signal that something has happened!

This design pattern lets us programmatically view the state of the world!


Addon
Formation
Service (Primary)
Service (Secondary)

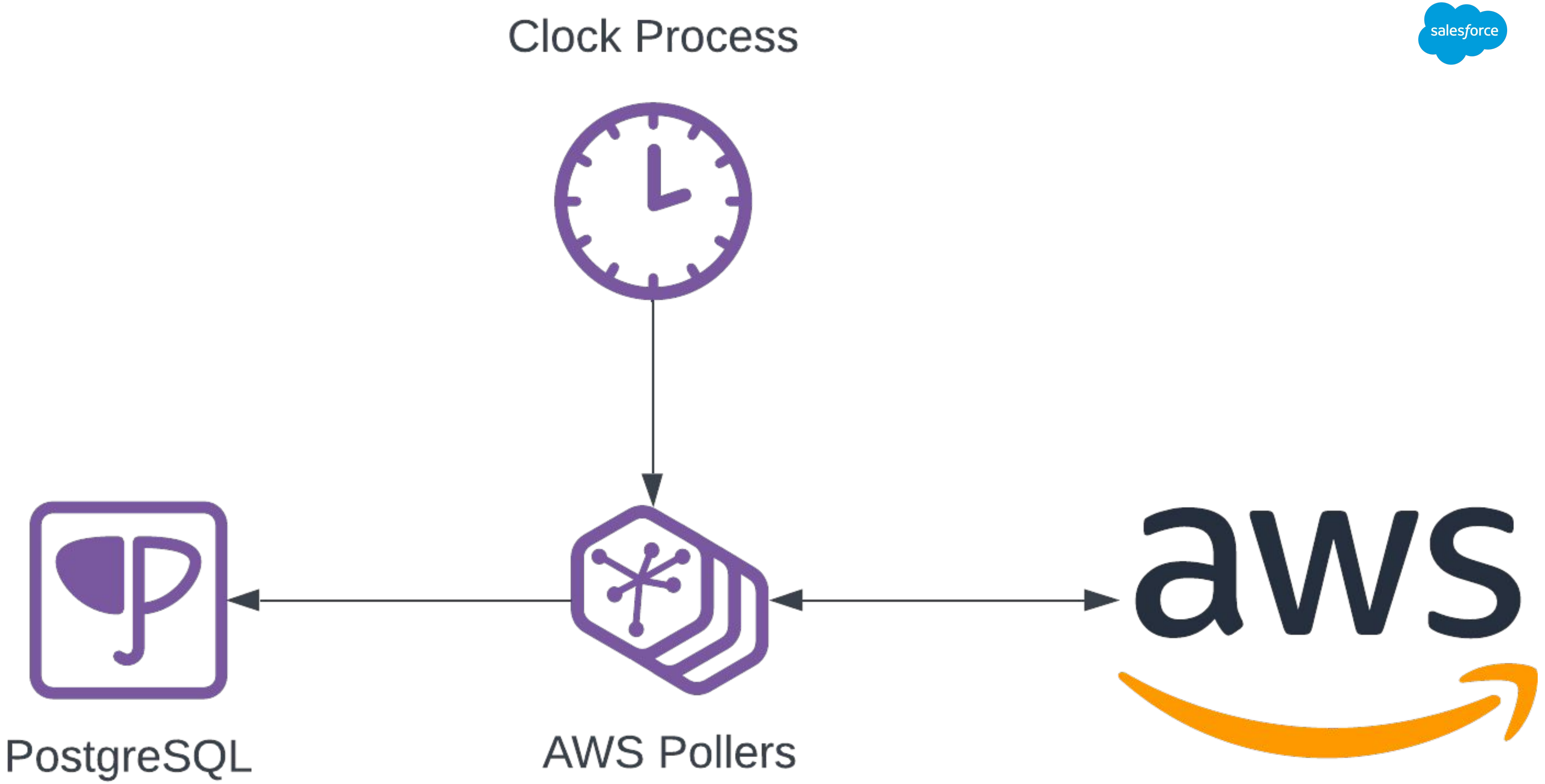
PostgreSQL Service

- PostgreSQL Version
- Database Credentials
- Forked From (optional)
- Following (optional)

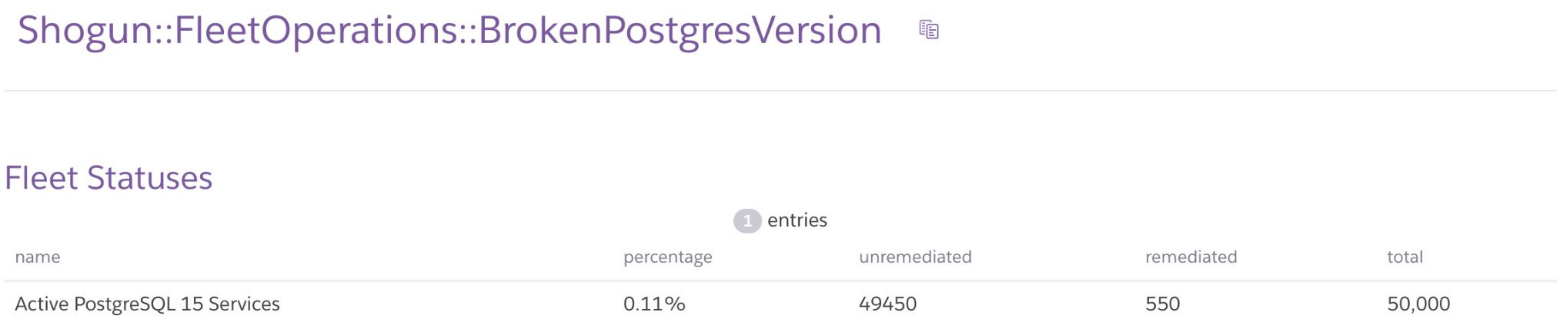


Service
Server
AWS::Ec2Instance
AWS::EBSVolume
EBS Volume
AWS::EIPAddress
Elastic IP Address
AWS::SecurityGroup
Security Group
Amazon EC2

So every AWS resource we manage also exists as a row in the database!


Clock Process
PostgreSQL
AWS Pollers
aws

I can build a dashboard so we know how many servers to update.



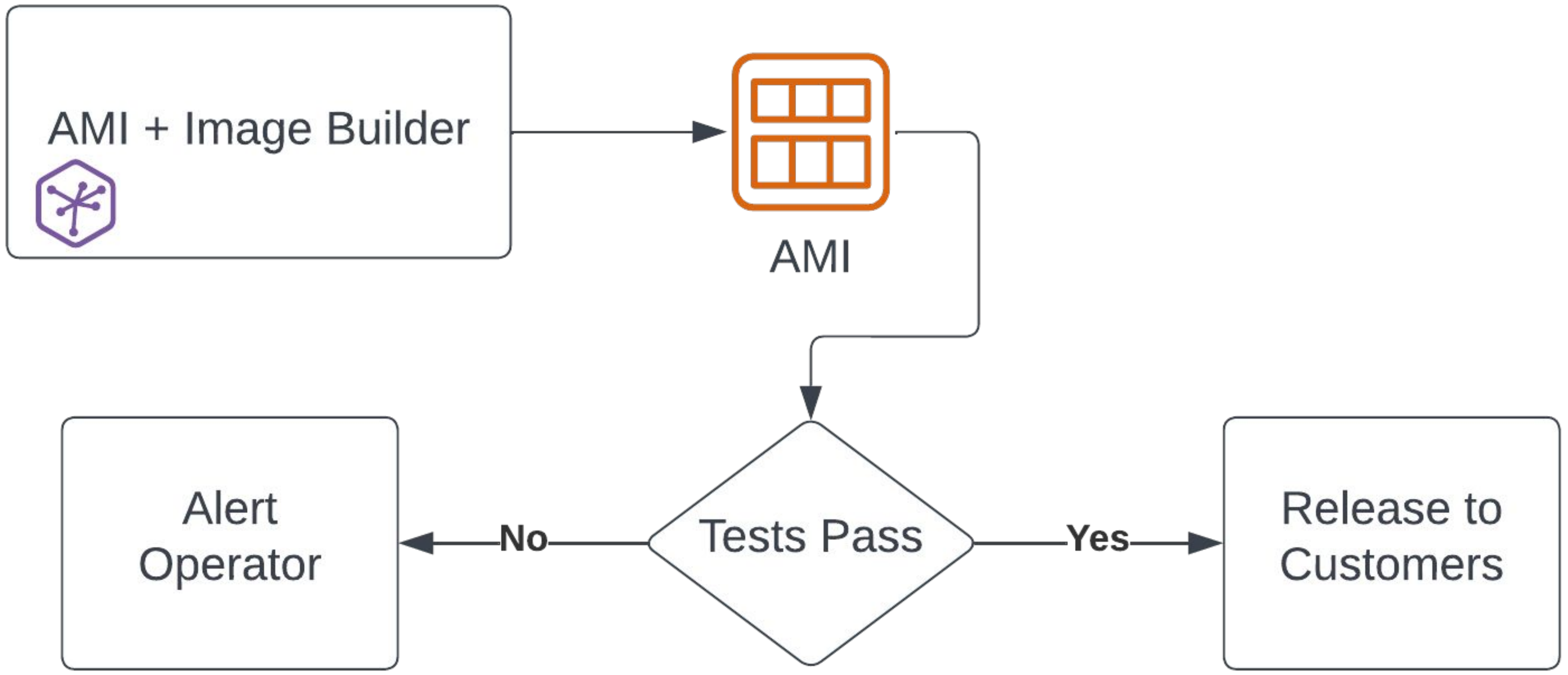
Shogun::FleetOperations::BrokenPostgresVersion

Fleet Statuses

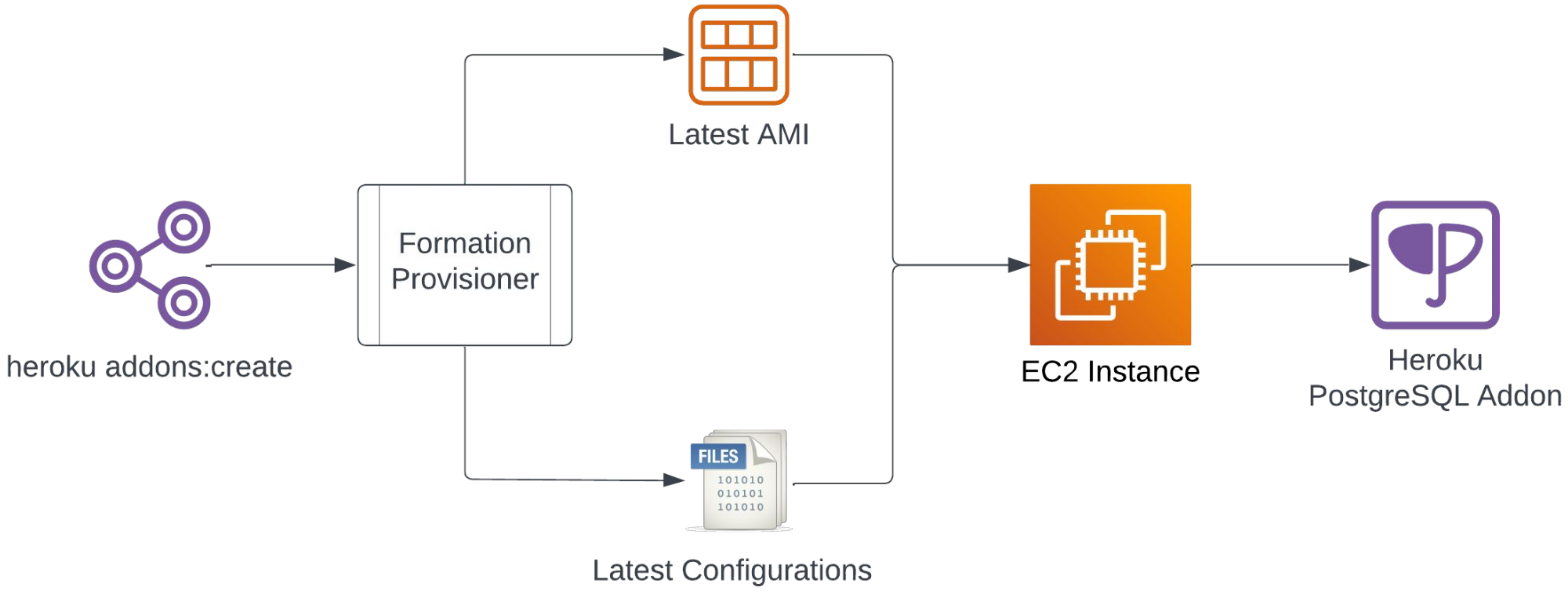
1 entries

name	percentage	unremediated	remediated	total
Active PostgreSQL 15 Services	0.11%	49450	550	50,000

Why do we already have 550 remediated servers?

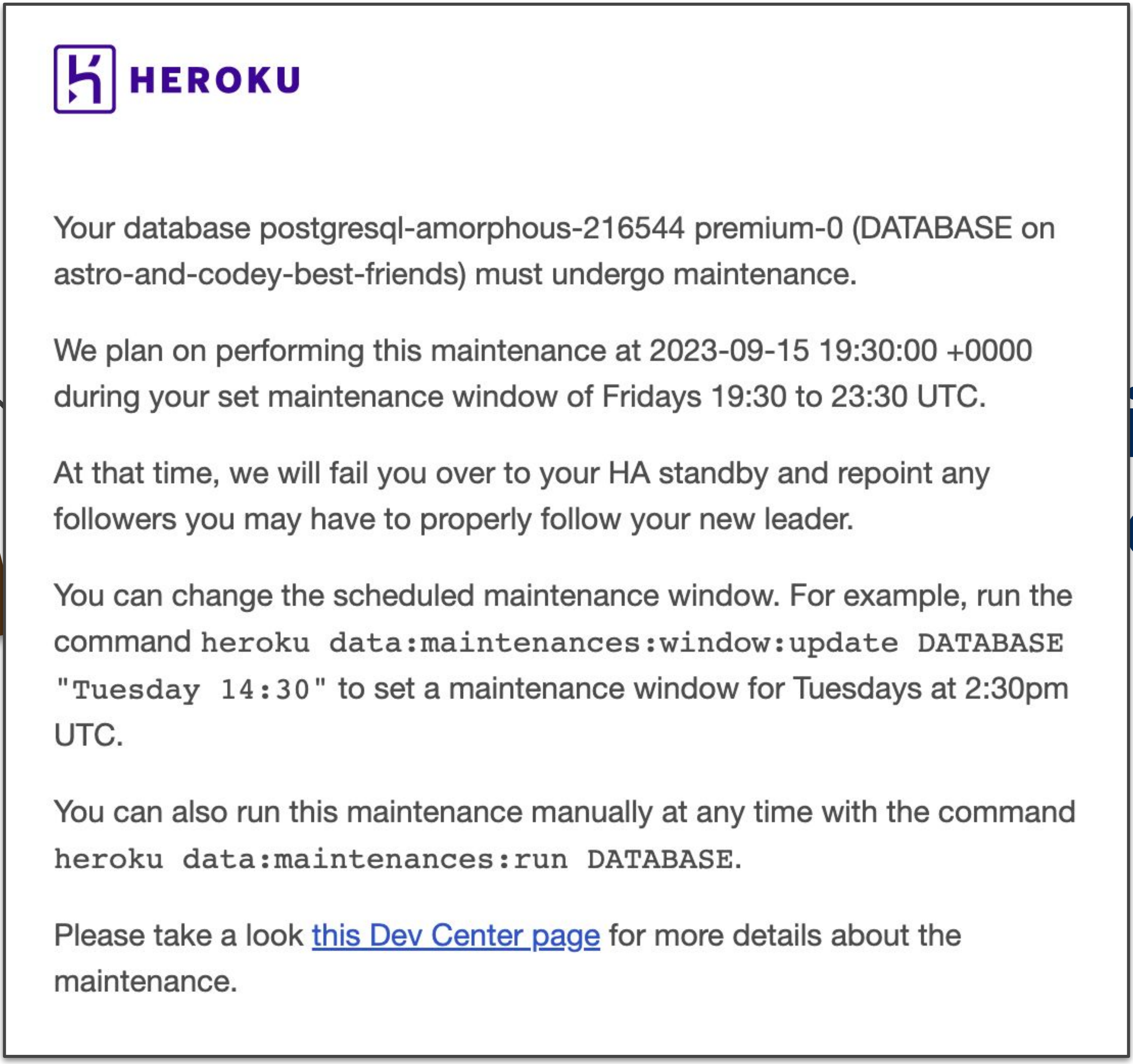

AMI + Image Builder
AMI
Tests Pass
No
Alert Operator
Yes
Release to Customers

So some customers got lucky and are safe from this exploit?


heroku addons:create
Formation Provisioner
Latest AMI
Latest Configurations
FILES
101010
010101
101010
EC2 Instance
Heroku PostgreSQL Addon

What about the remaining customers? We’ll have to upgrade them.

ing
o



HEROKU

Your database postgresql-amorphous-216544 premium-0 (DATABASE on astro-and-codey-best-friends) must undergo maintenance.

We plan on performing this maintenance at 2023-09-15 19:30:00 +0000 during your set maintenance window of Fridays 19:30 to 23:30 UTC.

At that time, we will fail you over to your HA standby and repoint any followers you may have to properly follow your new leader.

You can change the scheduled maintenance window. For example, run the command `heroku data:maintenances:window:update DATABASE "Tuesday 14:30"` to set a maintenance window for Tuesdays at 2:30pm UTC.

You can also run this maintenance manually at any time with the command `heroku data:maintenances:run DATABASE`.

Please take a look this Dev Center page for more details about the maintenance.

I never understood maintenance windows.

Can’t databases just run all the time?

AMI Decay (90 Days)

Changes requiring a PostgreSQL restart

Changes requiring an OS Reboot

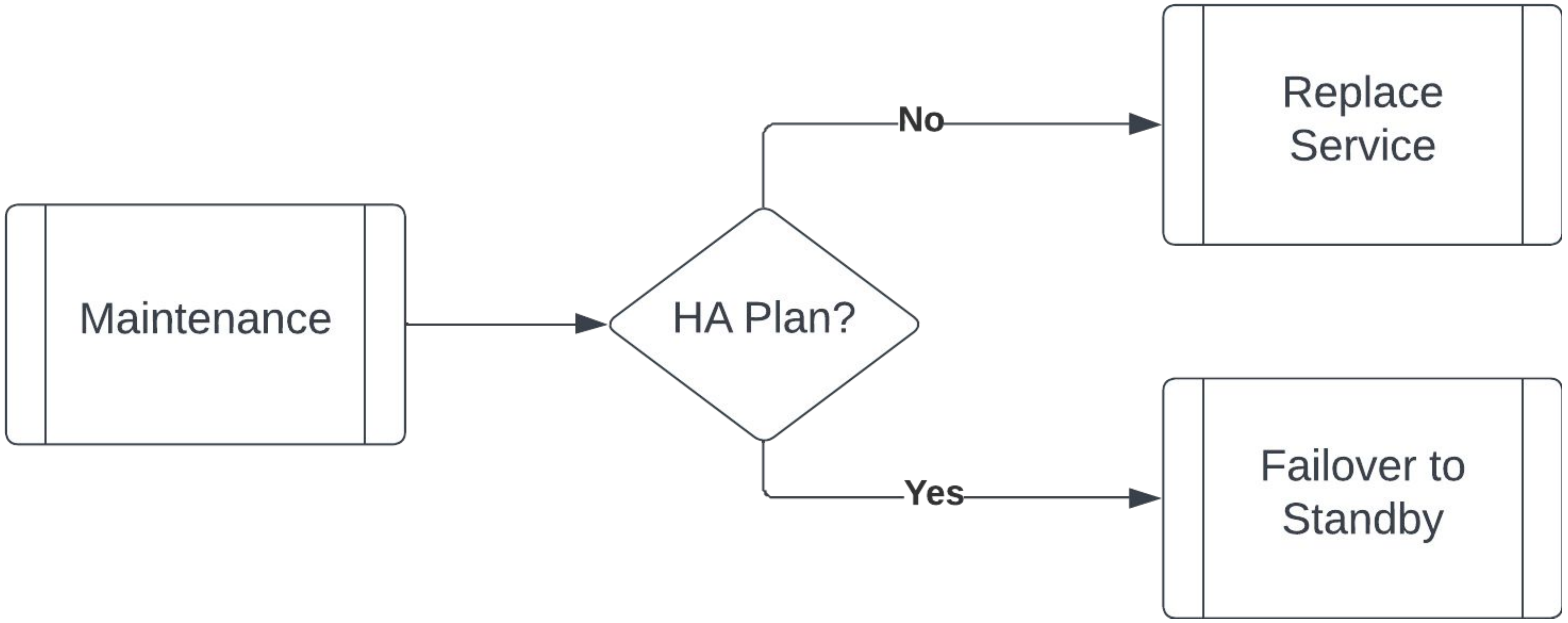
Underlying hardware failures

I thought EC2 just worked.
You mean EC2 hardware can fail too?

Good grief. I forgot that we're just renting parts of a datacenter.

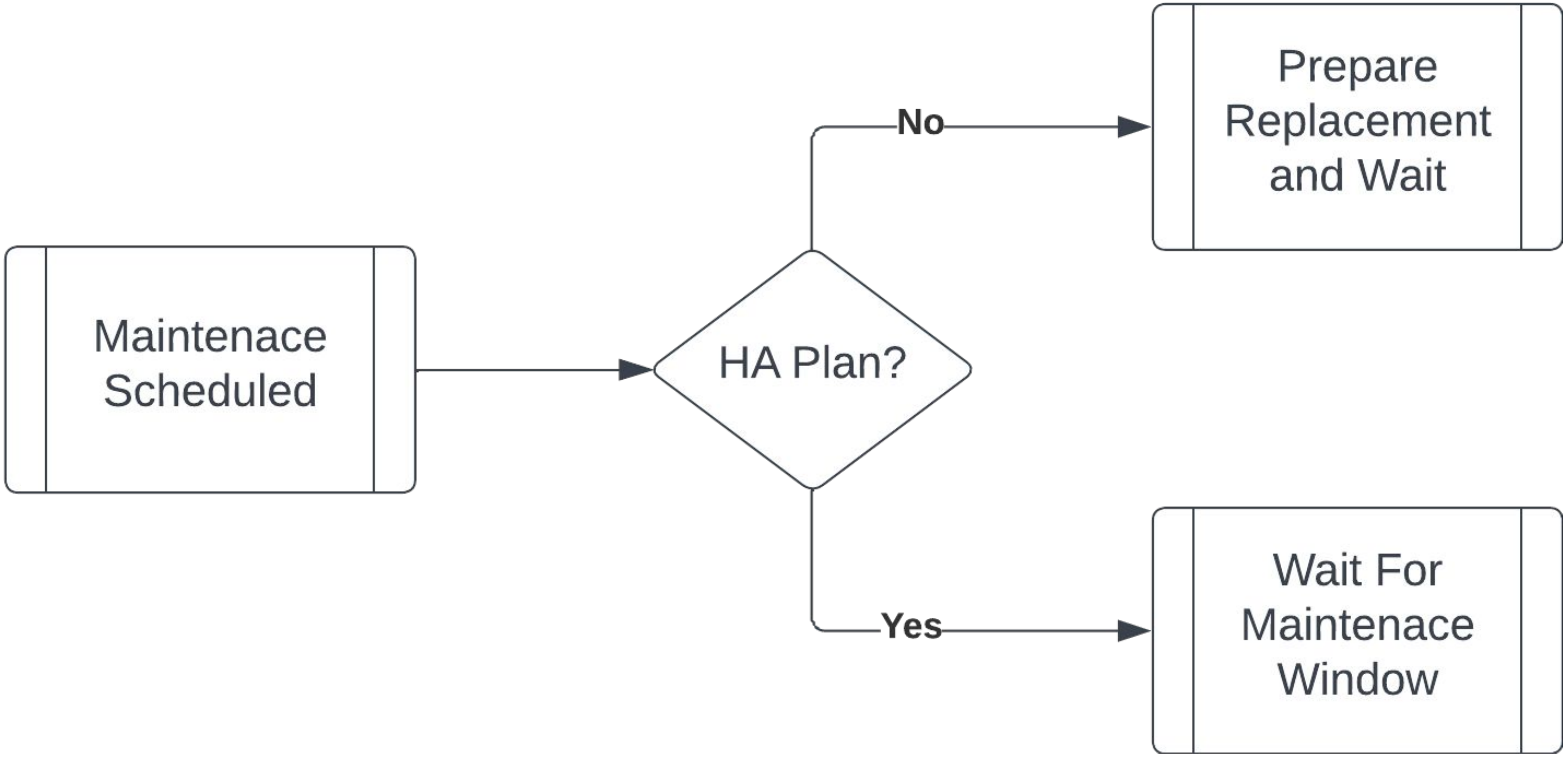
So updates to the Kernel or to PostgreSQL itself require restarts!

Do maintenances on highly-available databases require downtime?


Maintenance
HA Plan?
No
Replace Service
Yes
Failover to Standby

Standbys are replaced ASAP and failover is done during maintenances.

Does that mean non-HA databases are down during the whole maintenance?


Maintenace Scheduled
HA Plan?
No
Prepare Replacement and Wait
Yes
Wait For Maintenace Window

So they have replacements ready to go leading up to a maintenance!

If standard plans have replacements before a maintenance...

Then what makes them different from highly available plans?

HA Plans
(Premium, Private, Shield)

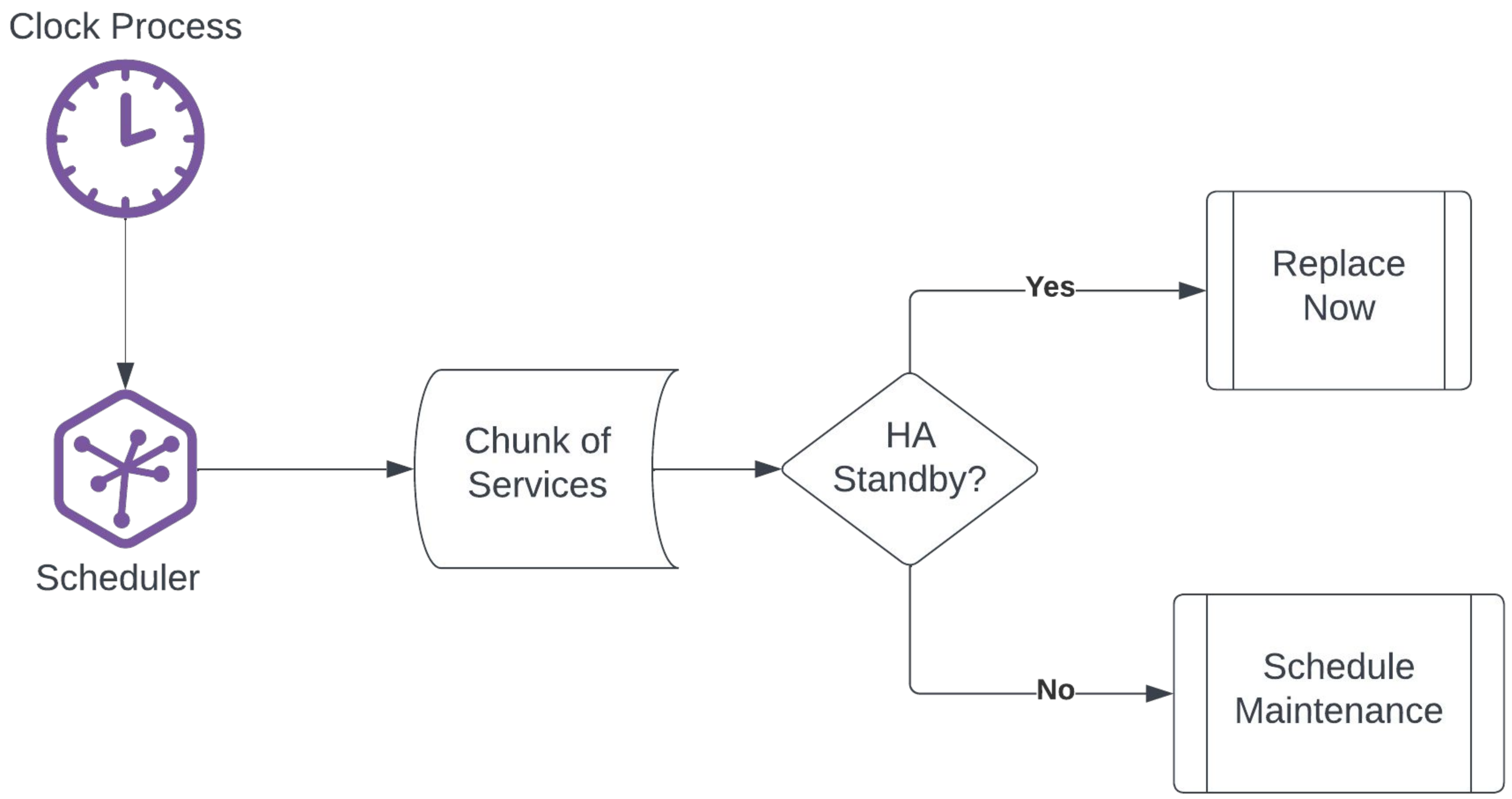
Downtime protection during UNPLANNED failures.
Fast to recover.

Standard Plans

Some downtime protection during planned maintenace.
Slow to recover otherwise.

A replacement is only guaranteed to be ready during a maintenance window.

How do I schedule all these maintenances? Do I just write a HUGE for loop?


Clock Process
Scheduler
Chunk of Services
HA Standby?
Yes
Replace Now
No
Schedule Maintenance

I know what a CPU scheduler is but what does that mean in this context?

```ruby
module Scheduler
  class AmiDecayScheduler < Base
    def remediated
      Server.active.amis_within_compliance
    end

    def unremediated
      Server.active.amis_approaching_expiration
    end
  end
end
```

```ruby
every 4.hours "schedule_ami_decay_maintenance" do
  scheduler = AmiDecayScheduler.new(fleet: Fleet::PostgresqlServers)

  scheduler.next_unremediated_batch.each do |server|
    if server.service.ha_follower?
      server.replace_now
    else
      server.schedule_maintenance
    end
  end
end
```

Why do we replace standbys now? Won’t that cause problems?

```ruby
state "replacing_standby" do
  self.new_standby = create_replacement_standby
  formation.leader.sync(self.new_standby)
  transition "syncing"
end

state "syncing" do
  formation.promote_standby(self.new_standby)
  transition "promoting"
end

state "promoting" do
  if formation.standby == self.new_standby
    old_standby.deprovision
    stop
  end
end
```

So a standby gets a standby and high availability is never broken!

What does scheduling a maintenance do anyway?

Latest AMI

Latest configurations

Latest PostgreSQL extensions and configurations

So, after a maintenance the customer’s database is on the latest AMI.

Which means the customer has the latest patches, making them up-to-date!

I can build a scheduler. I'll let you know when my PR is ready.

I refuse to do this joke but say it out loud if you must.

A Few Hours Later

All done! Mind reviewing my code?

https://bearcode.dev/heroku/data-cp

👍 LGTM!!!

Maintenances are being scheduled and our tracker from earlier is improving!

We’ve already got 1% of the fleet updated!

I’ll monitor over the next few days to make sure we’re still making progress.

I still refuse to do this joke but say it out loud if you must.

A Few Days Later

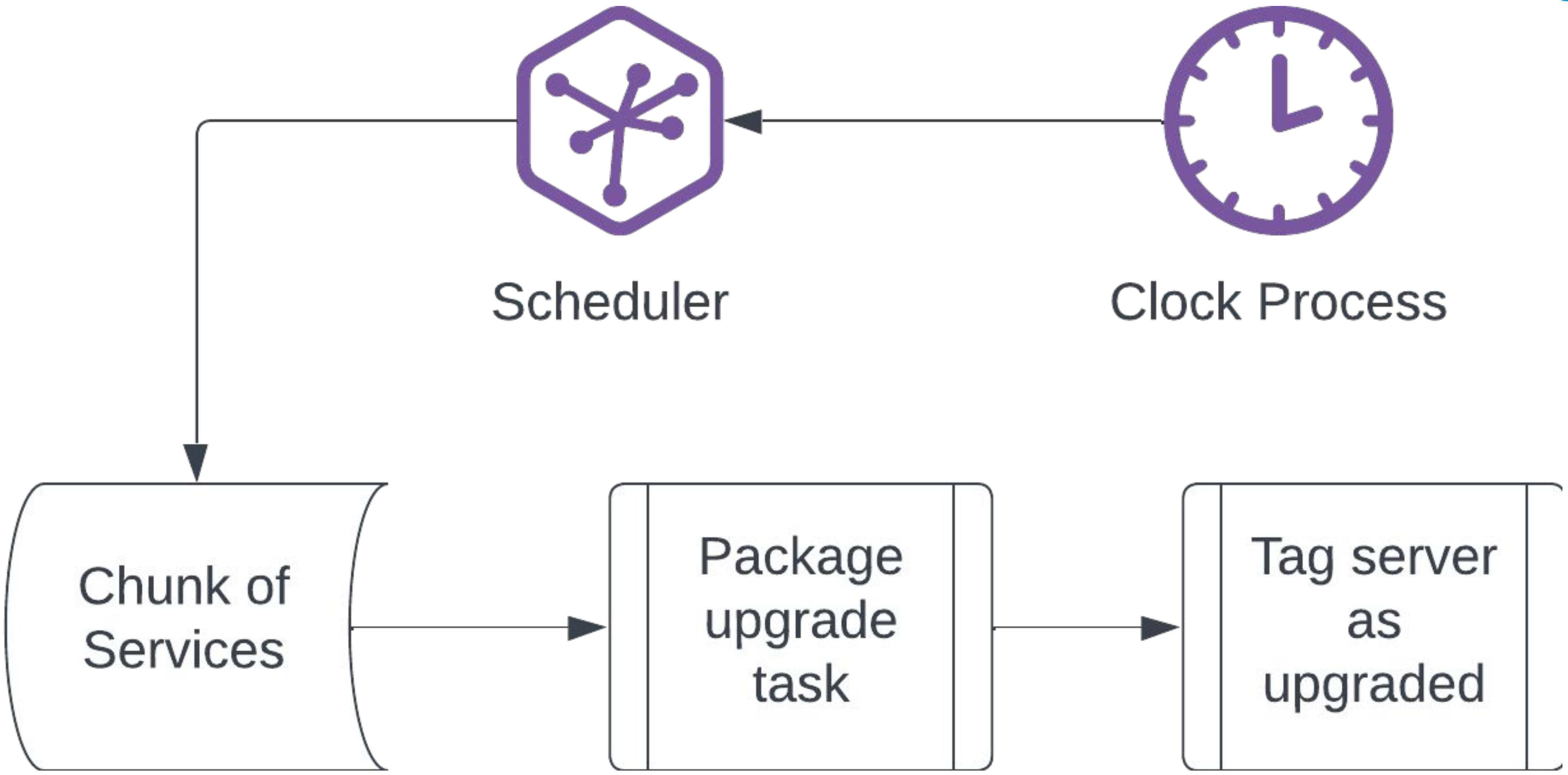
I just checked the tracker and we’re at 100%. We can tell security this is all done.

Thanks Codey! Compliance is coming back 👍 on our end.

You're one of the few teams that actually made the deadline.

This patch was a challenge because it required a PostgreSQL restart.

What about updates to packages the customer would never know about?


Scheduler
Clock Process
Chunk of Services
Package upgrade task
Tag server as upgraded

```ruby
module Scheduler
  class GlibcScheduler < Base
    INSTALLED = "glibc-2.38"
    LIMIT = 100

    def remediated
      Server.active.with_feature(:glibc_version_2_38)
    end

    def unremediated
      Server.active.missing_feature(:glibc_version_2_38)
    end
  end
end
```

```ruby
every 1.hour, "schedule_glibc_upgrades" do
  scheduler = GlibcScheduler.new(fleet: Fleet::PostgresqlServers)

  scheduler.next_unremediated_batch.each do |server|
    server.task_control(:upgrade_glibc).ensure_started
  end
end
```

```ruby
task(:upgrade_glibc) do
  field :run_id, type: String, default: nil

  state "start_package_upgrade" do
    begin
      self.run_id = server.ssm.task("apt-get upgrade glibc=glibc-2.38")
      save_changes
      transition "wait_for_completion"
    rescue Aws::SSM::Errors::RateLimited
      log("SSM rate limit exceeded, retrying on next tick.")
    end
  end

  state "wait_for_completion" do
    if server.ssm.state(self.run_id) == "complete"
      server.tag_feature(:glibc_version_2_38)
      stop
    end
  end
end
```

```ruby
class Server
  def tag_feature(feature_name)
    ssh.exec("touch /home/automation/features/#{feature_name}")
    self.features << feature_name
    save_changes
  end
end
```

I get it! Once the server is tagged then the scheduler will assume it is done.

Being a PaaS developer is such a rewarding challenge.

heroku.com/careers

Epilogue.

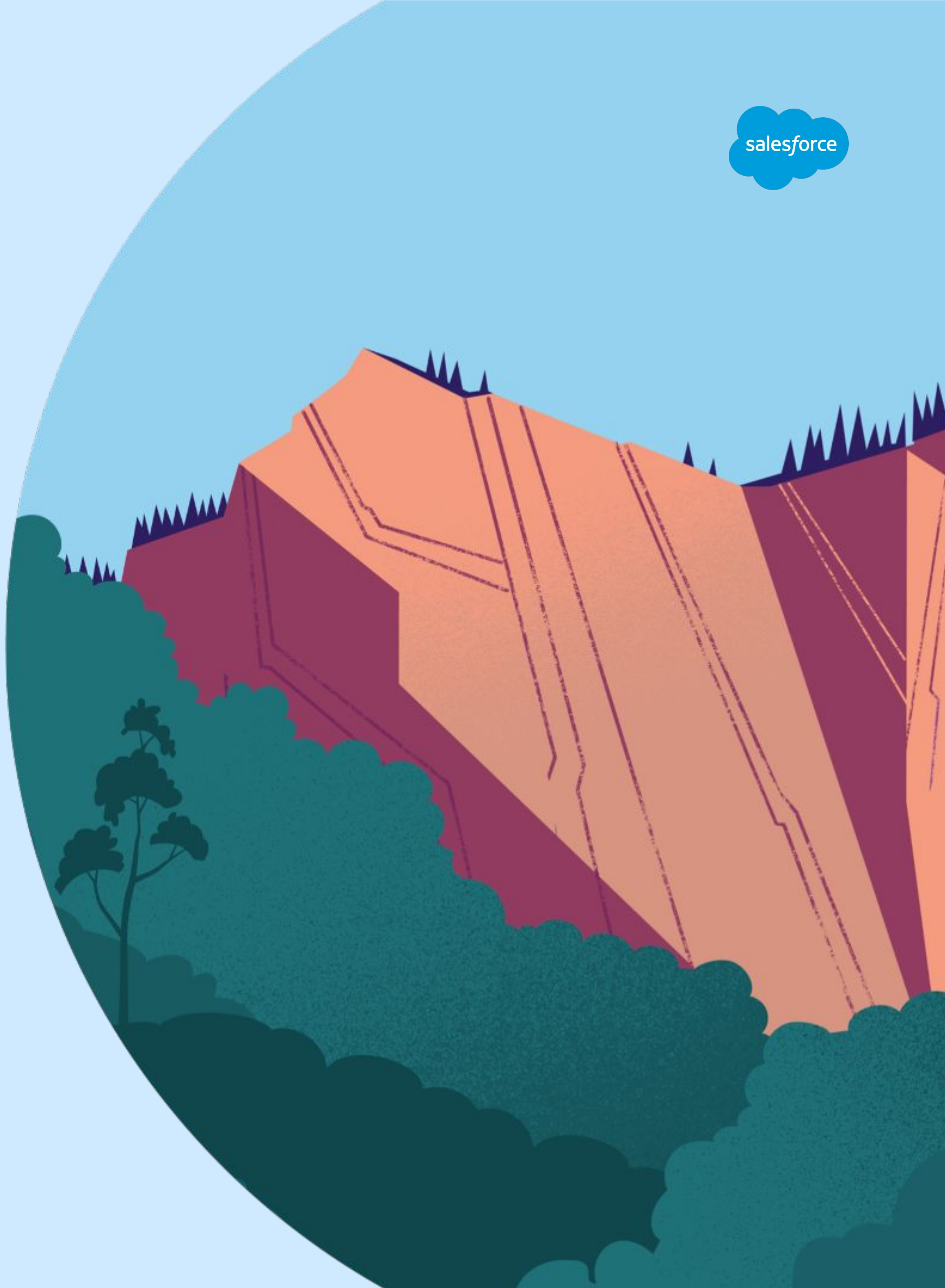
I can’t believe we patched 50,000 EC2 instances in a week.

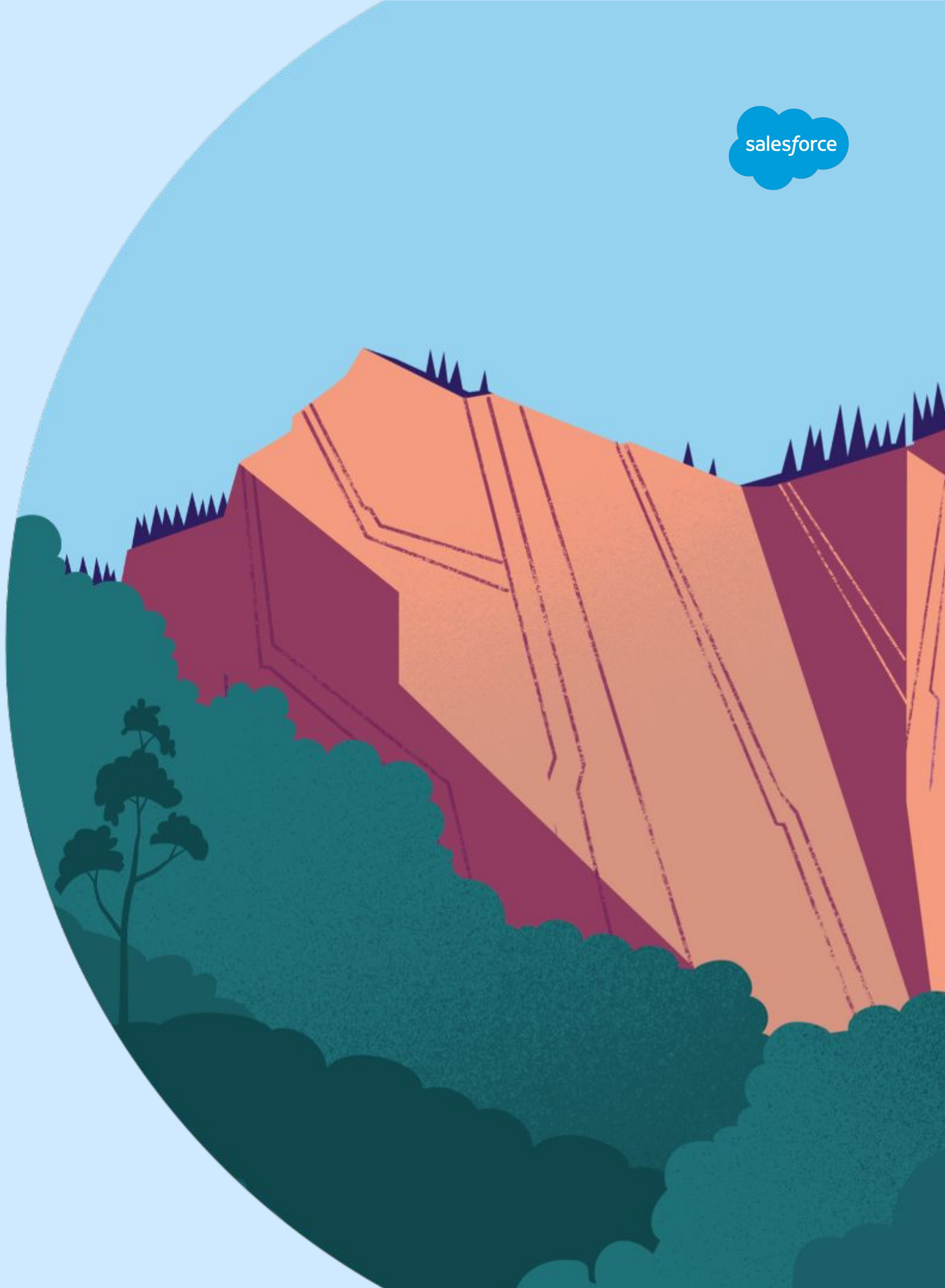
Yeah buddy! Now how about we both take some PTO?

Are you thinking what I’m thinking?

You know it, buddy! I'll pack my bag tonight. Do you still have that map?

Yep. I’ll pick you up tomorrow morning!!!





The End.

Thank you

Trailblazer
Trailblazer
Trailblazer
Trailblazer
Trailblazer

Fleet Operations

Changing the Engine in Mid-Air

Jon Daniel, Lead Infrastructure Engineer
jon.daniel@salesforce.com | me@jondaniel.email
They/Them