Cell architecture

Learn the basics of cells architecture, which team owns each component, how to get migration help, and find links to resources for interacting with cells.

Overview

This section of the New Relic Platform docs provides resources to understand cell architecture (cells) and the tools for migrating services from the CHI and EU data centers to cells.

Cells are our strategy to:

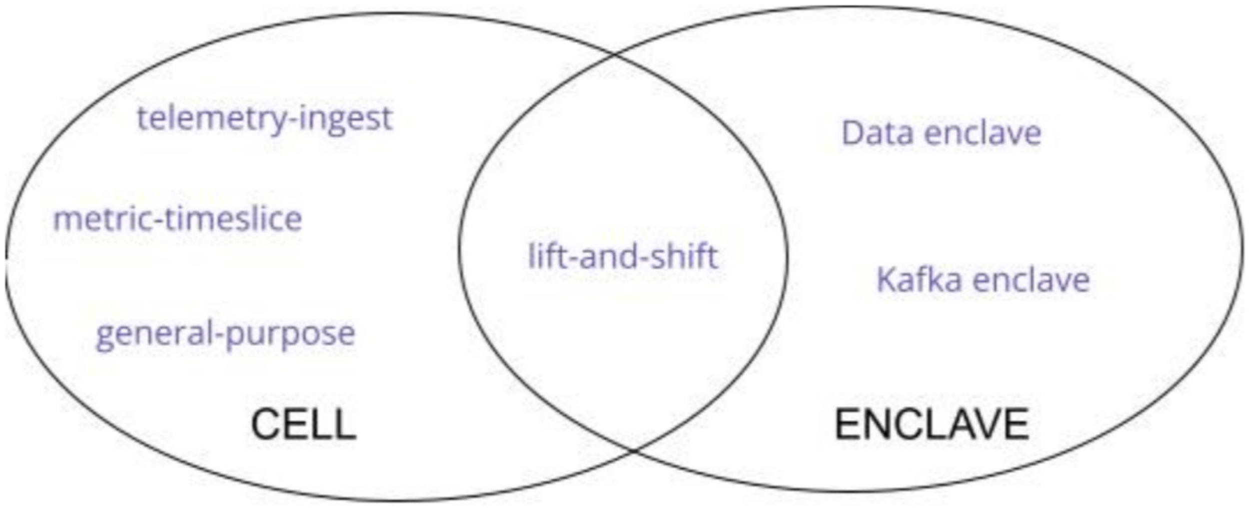
- Enable an incremental move to the public cloud.
- Unlock scaling for our services.
- Limit the blast radius of incidents.

For more background, see our founding vision for the cloud platform.

Cells are one of several options for running services or finding shared resources in the cloud. The options differ in several ways:

- Cell: Ephemeral, infinitely replicable, runs your code, and applications store data in enclaves.
- Enclave: Stateful, limited population, and runs your service on shared infrastructure.
- Lift and Shift Enclave (LSE): Runs your code but stores data in enclaves. Only one LSE exists at a time.

The image below helps visualize the relationship:



Cell types

Cells have different types that indicate their use case. To see the various cells and types, run `i9d cells list`. Many cells are infrastructure cells such as routing cells. Most cells run production applications.

GP cells

To avoid proliferation of special cases, most of New Relic’s services exist in a General Purpose Cell (General Pool Cell, GPC, or GP cell). GP Cells

- Stateless: GP cells run stateless services and don't save state in their environment. Stateless services which need databases or Kafka topics should use enclaves.

- Ephemeral: Cells are destroyed and recreated every 90 days. TDP ensures that removing a cell doesn't take your service offline.
- Infinitely replicable: GP cells are identical. Deploying more copies of a GP cell is one way to scale. If your service can't handle copies of itself, it is not suitable for GP cells.
- Your code: Deploy code to cells using Grand Central the same as you deploy it to the data centers.

For additional details on GP Cells, see Understanding GP cells. Migrating to GP Cells? See how to plan a migration to GP cells.

Enclaves

Enclaves don't run your code, and are only for shared services. There are several common stateful resources in enclaves such as databases, Redis, Kafka, and ZooKeeper.

There are a few use cases for enclaves:

- Data enclaves run Redis, Postgres, and MySQL for common use. See the DB self-service guide for details.
- Kafka enclaves hold a Kafka cluster for common use.

Lift and Shift Enclave

What if you need exactly one copy of your service? What if you need more than one copy of your service but can't handle an arbitrary number? Maybe you have some intra-service consensus protocol that needs an odd number of running instances less than ten.

Ultimately these applications should be made compatible with GP cells. However, we realize that not every application can handle GP cells. This led to the development of a Lift and Shift Enclave (LSE).

The LSE is a special instance of a GP cell. It shares some properties with a GP cell: it is stateless and runs your code. It also shares some properties with an enclave: it is long-lived and unique.

> The LSE is temporary. You can move your code into it but must plan to leave at some point. We haven't announced an official support horizon.

The LSE isn't as reliable as GP cells. The LSE doesn't scale, as there are inherent limits to the size of any k8s cluster. Also, there's more your team need to manage. For more details on the LSEs, see our migration guide.

Custom cells

These are specialized cells that perform a specific function where the components need to be tightly coupled. Examples include Telemetry Ingest (TI) and Metric Time Slice (MTS) cells. See the existing custom cell types, or learn how to create a new custom cell type.

If you don't see a cell type that works for your service, contact #help-cumulus-migration. We'll work together to determine if you can rework your application slightly to fit in GP cells or an existing cell type before deciding if a new cell type is warranted for your service.

Cell architecture basics

A cell is a unit of scale and isolation containing all relevant components for New Relic's products to work. Each cell is contained within one of the existing NR instances: `us`, `eu` or `staging`.

Cells run in AWS where the level of isolation is one AWS account per cell. Each cell contains AWS managed services (MSK, RDS, ElastiCache, etc) and at least one Kubernetes cluster. See the difference between the CHI data center and cell components.

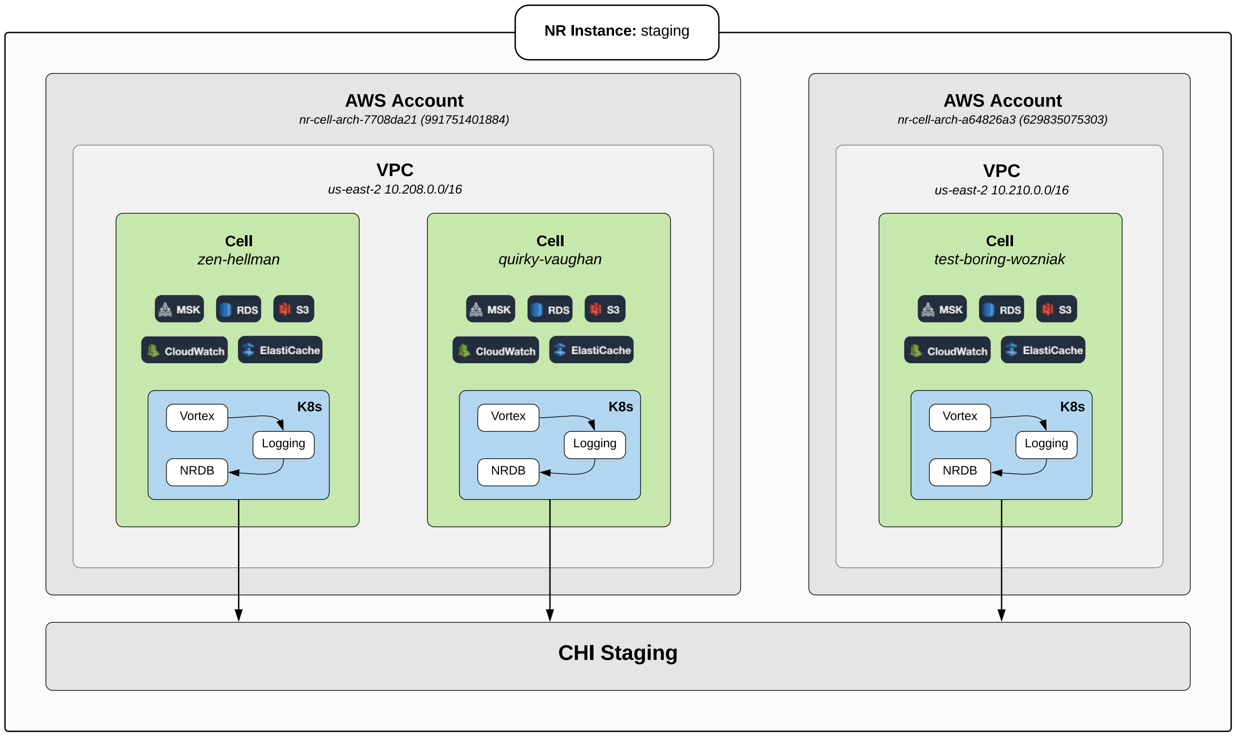
Although cells are currently based in AWS, work is already underway to consider a multi-cloud strategy using other platforms like Azure.

Cells are ephemeral and are regularly scheduled for build or decommission via Kitten Mittens. See this presentation that provides an overview of cell architecture.

The K8s cluster nodes are organized into pools:

- One general pool for most of the services
- Multiple dedicated pools for specialized workloads such as Vortex and NRDB

The diagram below shows how cells are laid out in the current `staging` NR instance:



Teams deploying to cells must set the alerts for the services they manage. Developers can use kubectl to inspect and troubleshoot their services.

Cell naming convention

Cells are given a unique name comprising the New Relic instance, an adjective, and a noun. `us-sweet-edison` as an example.

These names are human- and machine-readable. They don't convey all possible cell details (like what workloads it might run), but they help humans in normal operations and incidents because they are short names limited to no more than 18 characters.

Because of their ephemeral nature, cells are not renamed. Instead, a new cell is created and the older cell is decommissioned.

Migrating services to cells

Teams migrate their services from the Chicago datacenter to cells by choosing one of the following options:

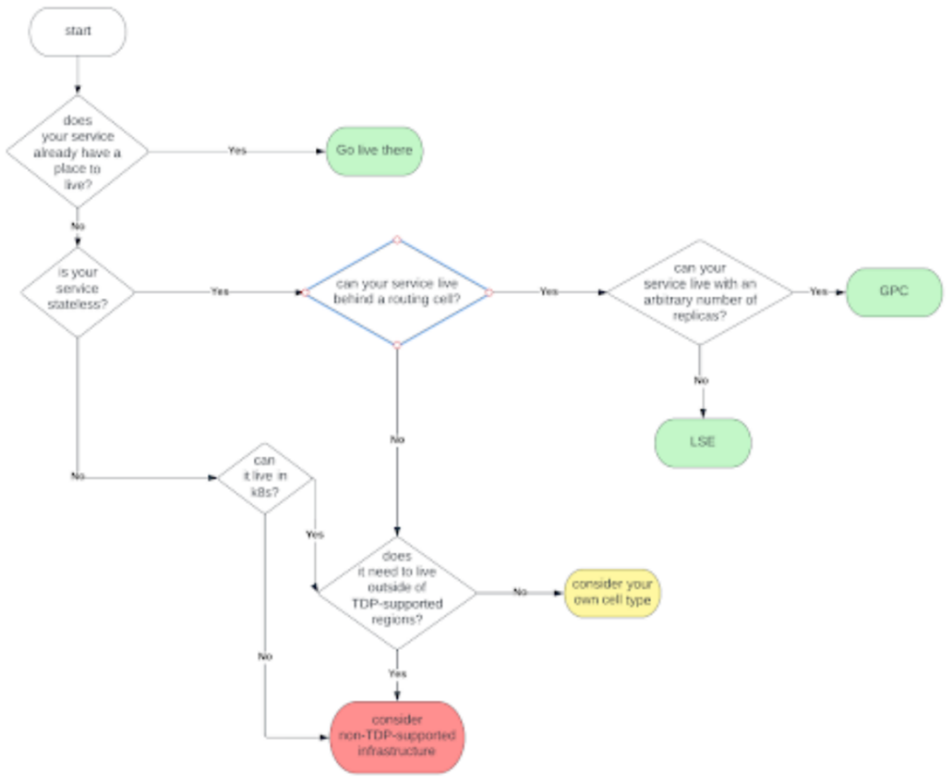
- Move stateless services to General Purpose Cells (GPC)

- Move to the Lift-and-Shift Enclave (LSE)
- Use an existing custom cell or create a new one

Choosing where to migrate your service

First, check your team's Jira under the Cumulus roadmap.

If the plan in Jira looks good then stick with it. If you're still not sure where to go, or if you think the plan in Jira needs adjusting, use this flowchart:


start
does your service already have a place to live?
Yes
Go live there
No
is your service stateless?
Yes
can your service live behind a routing cell?
Yes
can your service live with an arbitrary number of replicas?
Yes
GPC
No
LSE
No
No
can it live in k8s?
Yes
No
does it need to live outside of TDP-supported regions?
No
consider your own cell type
Yes
consider non-TDP-supported infrastructure

Bias your decision towards running in General Purpose cells unless you can justify not doing so.

There are a few things to sort out to facilitate migration:

- Not all languages have a FIPS-compliant crypto library. This impacts FEDRamp support.
- External fixed IPs.
- Custom-rolled VIPs and router setups.
- Zookeeper for shared services.

If you have concerns, questions, or need migration help, contact the #help-cumulus-migration Slack channel. Also, review the following resources before discussing migration plans with an architect:

- Understand what cells are and how they differ from CHI/Marathon deploys.
- Access kubernetes clusters in cells and understand cell deployments.

Getting help with Cumulus migrations

Cumulus has two main initiatives:

- exit the data centers
- exit the AWS Marathon cluster

Any work that fits into one of these initiatives is part of Cumulus. However, some teams have their own cloud computing environments outside of what the Platform, Data, and Intelligence (PDI) group offers. Those teams are also trying to harness PDI's overall cloud computing strategy. Those teams must use tooling in Cumulus and the people who support Cumulus also support this tooling.

However, services already in AWS on your own infrastructure are outside the scope of the Cumulus project and don't have the same priority to the Cumulus team. If you have questions or problems with specific pieces of technology, first visit the support channels for those technologies instead of #help-cumulus-migration:

- #db-team Slack for database concerns
- #help-kafka-platform Slack for Kafka concerns
- #help-elb Slack for ELB concerns
- #cell-arch Slack for questions and collaboration about cell architecture

If you have questions around cloud migration or don't know where to start, first talk to your architect. If you don't have an architect or you are the architect feel free to ask in #help-cumulus-migration but please tag it :no-party-cloud: This work is lower priority than Cumulus-specific work.

The Cumulus team cannot handle last-minute, time-sensitive, or complex problems where you're unsure where to start. Don't wait until the last minute to determine where you need help.

Compliance in the cloud

Compliance is tricky in the cloud. If you want to learn more about it, read our compliance documentation. More details are coming to make it easier to understand compliance requirements. This section will be updated with with those details.

Cell architecture component ownership

Check Team Store for email and slack channel details for the following teams or see support contacts.

Domain	Team
Architecture	Cell Leadership team (above)
Argo	Tools team
AWS ELBs in cells (load balancers)	Container Fabric team
AWS account access	Cloud Secrets & Identity
AWS resource creation (e.g. KMS, S3, ...)	requesting team
AWS tagging	requesting team
Account-based routing	Production Engineering
Cell builds and decommissions	Production Engineering
Cloudflare vendor relationship	Ingest
Direct Connect from cells to Chicago	NetEng team
Grand Central Kubernetes deploys	Tools team
Kafka mirror consultation	Kafka Platform Team
Kafka	Kafka Platform Team
Kubernetes clusters	Container Fabric team
Logging product	Logging team
Metrics product	Unified Data Streams team
NRDB operator	NRDB SRE team

Domain	Team
NRDB	NRDB teams
Relational DBs	DB team
Transit Gateway	NetEng team
Zookeeper	Kafka Platform Team

Related resources

- Migrating Kafka applications to the cloud
- Cumulus program page
- Cell build and decommission calendar
- Migrating to cells
- Accessing Kubernetes clusters

Back to top

Page last modified: Jun 15 2022 at 08:00 AM.

Edit this page on GHE