Package: starburst (via r-universe)

June 3, 2026

Type Package

Title Seamless AWS Cloud Bursting for Parallel R Workloads

Version 0.3.8

Description A 'future' backend that enables seamless execution of parallel R workloads on 'Amazon Web Services' ('AWS', <https://aws.amazon.com>), including 'EC2' and 'Fargate'. 'staRburst' handles environment synchronization, data transfer, quota management, and worker orchestration automatically, allowing users to scale from local execution to 100+ cloud workers with a single line of code change.

License Apache License 2.0

Encoding UTF-8

LazyData true

Depends R (>= 4.0.0)

Imports future (>= 1.33.0), globals, paws.compute, paws.storage, paws.management, paws.security.identity, qs2, uuid, renv, jsonlite, crayon, digest, base64enc, processx

Suggests future.apply, testthat (>= 3.0.0), knitr, rmarkdown, withr, mockery

VignetteBuilder knitr

URL https://starburst.ing, https://github.com/scttfrdmn/starburst

BugReports https://github.com/scttfrdmn/starburst/issues

Config/roxygen2/version 8.0.0

Config/pak/sysreqs make libxml2-dev libssl-dev

Repository https://scttfrdmn.r-universe.dev

Date/Publication 2026-06-03 00:46:11 UTC

RemoteUrl https://github.com/scttfrdmn/starburst

RemoteRef HEAD

RemoteSha d6499810b0df254ca0582c95724e5be3120c0e39

Contents

`future.starburst` *Create a Future using Starburst Backend*

Description

This is the entry point called by the Future package when a plan(starburst) is active

Usage

```
## S3 method for class 'starburst'
future(
  expr,
  envir = parent.frame(),
  substitute = TRUE,
```

```
  lazy = FALSE,
  seed = FALSE,
  globals = TRUE,
  packages = NULL,
  stdout = TRUE,
  conditions = "condition",
  label = NULL,
  ...
)
```

Arguments

expr	Expression to evaluate
envir	Environment for evaluation
substitute	Whether to substitute the expression
lazy	Whether to lazily evaluate (always FALSE for remote)
seed	Random seed
globals	Globals to export (TRUE for auto-detection, list for manual)
packages	Packages to load
stdout	Whether to capture stdout (TRUE, FALSE, or NA)
conditions	Character vector of condition classes to capture
label	Optional label for the future
...	Additional arguments

Value

A StarburstFuture object

---

launchFuture.StarburstBackend

Launch a future on the Starburst backend

Description

Launch a future on the Starburst backend

Usage

```
## S3 method for class 'StarburstBackend'
launchFuture(backend, future, ...)
```

Arguments

backend	A StarburstBackend object
future	The future object to launch
...	Additional arguments

Value

The future object (invisibly)

`listFutures.StarburstBackend` *List futures for StarburstBackend*

Description

List futures for StarburstBackend

Usage

```
## S3 method for class 'StarburstBackend'
listFutures(backend, ...)
```

Arguments

`backend`	A StarburstBackend object
`...`	Additional arguments

Value

List of futures (empty for this backend)

`nbrOfWorkers.StarburstBackend` *Number of workers for StarburstBackend*

Description

Number of workers for StarburstBackend

Usage

```
## S3 method for class 'StarburstBackend'
nbrOfWorkers(evaluator)
```

Arguments

`evaluator`	A StarburstBackend object

Value

Number of workers

`plan.starburst` *staRburst Future Backend*

Description

A future backend for running parallel R workloads on AWS (EC2 or Fargate)

Usage

```
## S3 method for class 'starburst'
plan(
  strategy,
  workers = 10,
  cpu = 4,
  memory = "8GB",
  region = NULL,
  timeout = 3600,
  auto_quota_request = interactive(),
  launch_type = "EC2",
  instance_type = "c7g.xlarge",
  use_spot = TRUE,
  warm_pool_timeout = 3600,
  detached = FALSE,
  ...
)
```

Arguments

`strategy`	The starburst strategy marker (ignored, for S3 dispatch)
`workers`	Number of parallel workers
`cpu`	vCPUs per worker (1, 2, 4, 8, or 16)
`memory`	Memory per worker (supports GB notation, e.g., "8GB")
`region`	AWS region (default: from config or "us-east-1")
`timeout`	Maximum runtime in seconds (default: 3600)
`auto_quota_request`	Automatically request quota increases (default: interactive())
`launch_type`	Launch type: EC2 or FARGATE (default: EC2)
`instance_type`	EC2 instance type when using EC2 launch type (default: c7g.xlarge)
`use_spot`	Use EC2 Spot instances for cost savings (default: TRUE)
`warm_pool_timeout`	Timeout for warm pool in seconds (default: 3600)
`detached`	Use detached session mode (deprecated, use starburst_session instead)
`...`	Additional arguments passed to future backend

Value

A future plan object

Examples

```
if (starburst_is_configured()) {
  future::plan(starburst, workers = 50)
  results <- future.apply::future_lapply(1:100, function(i) i^2)
}
```

`print.StarburstSessionStatus`
Print method for session status

Description

Print method for session status

Usage

```
## S3 method for class 'StarburstSessionStatus'
print(x, ...)
```

Arguments

`x`	A StarburstSessionStatus object
`...`	Additional arguments (ignored)

Value

Invisibly returns `x`.

`resolved.StarburstFuture`
Check if StarburstFuture is Resolved

Description

Checks whether the future task has completed execution

Usage

```
## S3 method for class 'StarburstFuture'
resolved(x, ...)
```

Arguments

x	A StarburstFuture object
...	Additional arguments

Value

Logical indicating if the future is resolved

---

result.StarburstFuture *Get Result from StarburstFuture*

Description

Retrieves the result from a resolved future

Usage

```
## S3 method for class 'StarburstFuture'
result(future, ...)
```

Arguments

future	A StarburstFuture object
...	Additional arguments

Value

A FutureResult object

---

run.StarburstFuture *Run a StarburstFuture*

Description

Submits the future task to AWS Fargate for execution

Usage

```
## S3 method for class 'StarburstFuture'
run(future, ...)
```

Arguments

future	A StarburstFuture object
...	Additional arguments

Value

The future object (invisibly)

`session-api` *Detached Session API*

Description

User-facing API for creating and managing detached sessions

`starburst` *Starburst strategy marker*

Description

This function should never be called directly. Use plan(starburst, ...) instead.

Usage

```
starburst(...)
```

Arguments

`...`	Arguments passed to StarburstBackend()

Value

Does not return a value; always signals an error if called directly. This object exists as a strategy marker for `plan`.

starburst_check_quota_request *Monitor quota increase request*

Description

Monitor quota increase request

Usage

```
starburst_check_quota_request(case_id, region = NULL)
```

Arguments

case_id	Case ID from quota increase request
region	AWS region

Value

Invisibly returns the quota request details, or NULL on error.

Examples

```
if (starburst_is_configured()) {
  starburst_check_quota_request("case-12345")
}
```

starburst_cleanup_ecr *Clean up staRburst ECR images*

Description

Manually delete Docker images from ECR to save storage costs. Images will be rebuilt on next use (adds 3-5 min delay).

Usage

```
starburst_cleanup_ecr(force = FALSE, region = NULL)
```

Arguments

force	Delete all images immediately, ignoring TTL
region	AWS region (default: from config)

Value

Invisibly returns TRUE on success or FALSE if not configured.

Examples

```
if (starburst_is_configured()) {
  # Delete images past TTL
  starburst_cleanup_ecr()

  # Delete all images immediately (save $0.50/month)
  starburst_cleanup_ecr(force = TRUE)
}
```

---

starburst_cluster *Create a Starburst Cluster*

Description

Creates a cluster object for managing AWS Fargate workers using Future backend

Usage

```
starburst_cluster(
  workers = 10,
  cpu = 4,
  memory = "8GB",
  platform = "X86_64",
  region = NULL,
  timeout = 3600
)
```

Arguments

workers	Number of parallel workers
cpu	CPU units per worker
memory	Memory per worker
platform	CPU architecture (X86_64 or ARM64)
region	AWS region
timeout	Maximum runtime in seconds

Value

A starburst_cluster object

Examples

```
if (starburst_is_configured()) {
  cluster <- starburst_cluster(workers = 20)
  results <- cluster$map(data, function(x) x * 2)
}
```

`starburst_config` *Configure staRburst options*

Description

Configure staRburst options

Usage

```
starburst_config(
  max_cost_per_job = NULL,
  cost_alert_threshold = NULL,
  auto_cleanup_s3 = NULL,
  ...
)
```

Arguments

`max_cost_per_job`	Maximum cost per job in dollars
`cost_alert_threshold`	Cost threshold for alerts
`auto_cleanup_s3`	Automatically clean up S3 files after completion
`...`	Additional configuration options

Value

Invisibly returns the updated configuration list.

Examples

```
if (starburst_is_configured()) {
  starburst_config(
    max_cost_per_job = 10,
    cost_alert_threshold = 5
  )
}
```

`starburst_estimate` *Estimate Cloud Performance and Cost*

Description

Runs a small sample of tasks locally to estimate cloud execution time and cost. Provides informed prediction before spending money on cloud execution.

Usage

```
starburst_estimate(
  .x,
  .f,
  workers = 10,
  cpu = 2,
  memory = "8GB",
  platform = "X86_64",
  sample_size = 10,
  region = NULL,
  ...
)
```

Arguments

`.x`	A vector or list to iterate over
`.f`	A function to apply to each element
`workers`	Number of parallel workers to estimate for
`cpu`	CPU units per worker (1, 2, 4, 8, or 16)
`memory`	Memory per worker (e.g., "8GB")
`platform`	CPU architecture: "X86_64" (default) or "ARM64" (Graviton3)
`sample_size`	Number of items to run locally for estimation (default: 10)
`region`	AWS region
`...`	Additional arguments passed to .f

Value

Invisible list with estimates, prints summary to console

Examples

```
if (starburst_is_configured()) {
  # Estimate before running
  starburst_estimate(1:1000, expensive_function, workers = 50)

  # Then decide whether to proceed
```

```
  results <- starburst_map(1:1000, expensive_function, workers = 50)
}
```

`starburst_is_configured` *Check if staRburst is configured*

Description

Returns `TRUE` if `starburst_setup()` has been run, the configuration file exists, and AWS credentials are available. Useful for guarding example code that requires AWS credentials.

Usage

```
starburst_is_configured()
```

Value

`TRUE` if configured and credentials are available, `FALSE` otherwise.

Examples

```
starburst_is_configured()
```

`starburst_list_sessions` *List All Sessions*

Description

List all detached sessions in S3

Usage

```
starburst_list_sessions(region = NULL)
```

Arguments

`region`	AWS region (default: from config)

Value

Data frame with session information

Examples

```
if (starburst_is_configured()) {
  sessions <- starburst_list_sessions()
  print(sessions)
}
```

`starburst_logs` *View worker logs*

Description

View worker logs

Usage

```
starburst_logs(task_id = NULL, cluster_id = NULL, last_n = 50, region = NULL)
```

Arguments

`task_id`	Optional task ID to view logs for specific task
`cluster_id`	Optional cluster ID to view logs for specific cluster
`last_n`	Number of last log lines to show (default: 50)
`region`	AWS region (default: from config)

Value

Invisibly returns the list of log events, or `NULL` if no events were found.

Examples

```
if (starburst_is_configured()) {
  # View recent logs
  starburst_logs()

  # View logs for specific task
  starburst_logs(task_id = "abc-123")

  # View last 100 lines
  starburst_logs(last_n = 100)
}
```

starburst_map *Map Function Over Data Using AWS Fargate*

Description

Parallel map function that executes on AWS Fargate using the Future backend

Usage

```
starburst_map(
  .x,
  .f,
  workers = 10,
  cpu = 4,
  memory = "8GB",
  platform = "X86_64",
  region = NULL,
  timeout = 3600,
  .progress = TRUE,
  ...
)
```

Arguments

.x	A vector or list to iterate over
.f	A function to apply to each element
workers	Number of parallel workers (default: 10)
cpu	CPU units per worker (1, 2, 4, 8, or 16)
memory	Memory per worker (e.g., 8GB)
platform	CPU architecture (X86_64 or ARM64)
region	AWS region
timeout	Maximum runtime in seconds per task
.progress	Show progress bar (default: TRUE)
...	Additional arguments passed to .f

Value

A list of results, one per element of .x

Examples

```
if (starburst_is_configured()) {
  # Simple parallel computation
  results <- starburst_map(1:100, function(x) x^2, workers = 10)

  # With custom configuration
  results <- starburst_map(
    data_list,
    expensive_function,
    workers = 50,
    cpu = 4,
    memory = "8GB"
  )
}
```

`starburst_quota_status` *Show quota status*

Description

Show quota status

Usage

```
starburst_quota_status(region = NULL)
```

Arguments

`region`	AWS region (default: from config)

Value

Invisibly returns a list with quota information including current limit, usage, and any pending requests.

Examples

```
if (starburst_is_configured()) {
  starburst_quota_status()
}
```

starburst_rebuild_environment

Rebuild environment image

Description

Rebuild environment image

Usage

```
starburst_rebuild_environment(region = NULL, force = FALSE)
```

Arguments

region	AWS region (default: from config)
force	Force rebuild even if current environment hasn't changed

Value

Invisibly returns NULL. Called for its side effect of rebuilding and pushing the Docker environment image.

Examples

```
if (starburst_is_configured()) {
  starburst_rebuild_environment()
}
```

starburst_request_quota_increase

Request quota increase (user-facing)

Description

Request quota increase (user-facing)

Usage

```
starburst_request_quota_increase(vcpus = 500, region = NULL)
```

Arguments

vcpus	Desired vCPU quota
region	AWS region (default: from config)

Value

Invisibly returns TRUE if the increase was requested, FALSE if already sufficient or cancelled.

Examples

```
if (starburst_is_configured()) {
  starburst_request_quota_increase(vcpus = 500)
}
```

`starburst_session` *Create a Detached Starburst Session*

Description

Creates a new detached session that can run computations independently of your R session. You can close R and reattach later to collect results.

Usage

```
starburst_session(
  workers = 10,
  cpu = 4,
  memory = "8GB",
  region = NULL,
  timeout = 3600,
  session_timeout = 3600,
  absolute_timeout = 86400,
  launch_type = "EC2",
  instance_type = "c7g.xlarge",
  use_spot = TRUE,
  warm_pool_timeout = 3600
)
```

Arguments

`workers`	Number of parallel workers (default: 10)
`cpu`	vCPUs per worker (default: 4)
`memory`	Memory per worker, e.g., "8GB" (default: "8GB")
`region`	AWS region (default: from config or "us-east-1")
`timeout`	Task timeout in seconds (default: 3600)
`session_timeout`	Active timeout in seconds (default: 3600)
`absolute_timeout`	Maximum session lifetime in seconds (default: 86400)

launch_type "FARGATE" or "EC2" (default: "FARGATE")

instance_type EC2 instance type for EC2 launch (default: "c6a.large")

use_spot Use spot instances for EC2 (default: FALSE)

warm_pool_timeout EC2 warm pool timeout in seconds (default: 3600)

Value

A StarburstSession object with methods:

- `submit(expr, ...)` - Submit a task to the session
- `status()` - Get progress summary
- `collect(wait = FALSE)` - Collect completed results
- `extend(seconds = 3600)` - Extend timeout
- `cleanup()` - Terminate and cleanup

Examples

```
if (starburst_is_configured()) {
  # Create detached session
  session <- starburst_session(workers = 10)

  # Submit tasks
  task_ids <- lapply(1:100, function(i) {
    session$submit(quote(expensive_computation(i)))
  })

  # Close R and come back later...
  session_id <- session$session_id

  # Reattach
  session <- starburst_session_attach(session_id)

  # Collect results
  results <- session$collect(wait = TRUE)
}
```

---

starburst_session_attach

Reattach to Existing Session

Description

Reattach to a previously created detached session

Usage

```
starburst_session_attach(session_id, region = NULL)
```

Arguments

session_id	Session identifier
region	AWS region (default: from config)

Value

A StarburstSession object

Examples

```
if (starburst_is_configured()) {
  session <- starburst_session_attach("session-abc123")
  status <- session$status()
  results <- session$collect()
}
```

---

starburst_setup *Setup staRburst*

Description

One-time configuration to set up AWS resources for staRburst

Usage

```
starburst_setup(
  region = "us-east-1",
  force = FALSE,
  use_public_base = TRUE,
  ecr_image_ttl_days = NULL,
  build_image = TRUE
)
```

Arguments

region	AWS region (default: "us-east-1")
force	Force re-setup even if already configured
use_public_base	Use public base Docker images (default: TRUE). Set to FALSE to build private base images in your ECR.

ecr_image_ttl_days
: Number of days to keep Docker images in ECR (default: NULL = never delete). AWS will automatically delete images older than this many days. This prevents surprise costs if you stop using staRburst. Recommended: 30 days for regular users, 7 days for occasional users. When images are deleted, they will be rebuilt on next use (adds 3-5 min).

build_image
: Build the worker environment image during setup (default: TRUE). Set to FALSE to provision AWS resources (S3/ECR/ECS/VPC), write config, and check quotas without triggering the multi-minute Docker image build. The image is then built lazily on first worker launch via `ensure_environment()`. Useful for CI / connectivity checks.

Value

Invisibly returns the configuration list.

Examples

```
if (starburst_is_configured()) {
  # Default: keep images forever (~$0.50/month idle cost)
  starburst_setup()

  # Auto-delete images after 30 days (saves money if you stop using it)
  starburst_setup(ecr_image_ttl_days = 30)

  # Use private base images with 7-day cleanup
  starburst_setup(use_public_base = FALSE, ecr_image_ttl_days = 7)

  # Provision resources without building the image (fast; CI / connectivity checks)
  starburst_setup(build_image = FALSE)
}
```

---

starburst_setup_ec2 *Setup EC2 capacity providers for staRburst*

Description

One-time setup for EC2 launch type. Creates IAM roles, instance profiles, and capacity providers for specified instance types.

Usage

```
starburst_setup_ec2(
  region = "us-east-1",
  instance_types = c("c7g.xlarge", "c7i.xlarge"),
  force = FALSE
)
```

Arguments

region	AWS region (default: "us-east-1")
instance_types	Character vector of instance types to setup (default: c("c7g.xlarge", "c7i.xlarge"))
force	Force re-setup even if already configured

Value

Invisibly returns TRUE on success or FALSE on failure or cancellation.

Examples

```
if (starburst_is_configured()) {
  # Setup with default instance types (Graviton and Intel)
  starburst_setup_ec2()

  # Setup with custom instance types
  starburst_setup_ec2(instance_types = c("c7g.2xlarge", "r7g.xlarge"))
}
```

---

starburst_status *Show staRburst status*

Description

Show staRburst status

Usage

```
starburst_status()
```

Value

Invisibly returns a list with current configuration and quota information.

`StarburstBackend` *Starburst Future Backend*

Description

A future backend for running parallel R workloads on AWS ECS

Usage

```
StarburstBackend(
  workers = 10,
  cpu = 4,
  memory = "8GB",
  region = NULL,
  timeout = 3600,
  launch_type = "EC2",
  instance_type = "c6a.large",
  use_spot = FALSE,
  warm_pool_timeout = 3600,
  ...
)
```

Arguments

`workers`	Number of parallel workers
`cpu`	vCPUs per worker (1, 2, 4, 8, or 16)
`memory`	Memory per worker (supports GB notation, e.g., "8GB")
`region`	AWS region (default: from config or "us-east-1")
`timeout`	Maximum runtime in seconds (default: 3600)
`launch_type`	"EC2" or "FARGATE" (default: "EC2")
`instance_type`	EC2 instance type (e.g., "c6a.large")
`use_spot`	Use spot instances (default: FALSE)
`warm_pool_timeout`	Pool timeout in seconds (default: 3600)
`...`	Additional arguments

Value

A StarburstBackend object

Index