Package: crew.aws.batch (via r-universe)

September 9, 2024

Title A Crew Launcher Plugin for AWS Batch

Description In computationally demanding analysis projects,
statisticians and data scientists asynchronously deploy
long-running tasks to distributed systems, ranging from
traditional clusters to cloud services. The 'crew.aws.batch'
package extends the 'mirai'-powered 'crew' package with a
worker launcher plugin for AWS Batch. Inspiration also comes
from packages 'mirai' by Gao (2023)
<https://github.com/shikokuchuo/mirai>, 'future' by Bengtsson
(2021) <doi:10.32614/RJ-2021-048>, 'rrq' by FitzJohn and Ashton
(2023) <https://github.com/mrc-ide/rrq>, 'clustermq' by
Schubert (2019) <doi:10.1093/bioinformatics/btz284>), and
'batchtools' by Lang, Bischl, and Surmann (2017).
<doi:10.21105/joss.00135>.

Version 0.0.6.9005

License MIT + file LICENSE

URL https://wlandau.github.io/crew.aws.batch/,
https://github.com/wlandau/crew.aws.batch

BugReports https://github.com/wlandau/crew.aws.batch/issues

Depends R (>= 4.0.0)

Imports cli (>= 3.1.0), crew (>= 0.9.5.9007), paws.common (>= 0.7.0),
paws.compute, paws.management, R6, rlang, tibble, utils

Suggests knitr (>= 1.30), markdown (>= 1.1), rmarkdown (>= 2.4),
testthat (>= 3.0.0)

Encoding UTF-8

Language en-US

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

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

RemoteUrl https://github.com/wlandau/crew.aws.batch

RemoteRef HEAD

RemoteSha 9d71570b75de8679176480227e976c3ce7ba4eac

Contents

---

crew.aws.batch-package

crew.aws.batch: a crew launcher plugin for AWS Batch

Description

In computationally demanding analysis projects, statisticians and data scientists asynchronously deploy long-running tasks to distributed systems, ranging from traditional clusters to cloud services. The crew.aws.batch package extends the `mirai`-powered `crew` package with worker launcher plugins for AWS Batch. Inspiration also comes from packages `mirai`, `future`, `rrq`, `clustermq`, and `batchtools`.

---

crew_class_definition_aws_batch

AWS Batch definition class

Description

AWS Batch definition `R6` class

Details

See `crew_definition_aws_batch()`.

IAM policies

In order for the AWS Batch `crew` job definition class to function properly, your IAM policy needs permission to perform the `RegisterJobDefinition`, `DeregisterJobDefinition`, and `DescribeJobDefinitions` AWS Batch API calls. For more information on AWS policies and permissions, please visit `https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html`.

Active bindings

job_queue See `crew_definition_aws_batch()`.

job_definition See `crew_definition_aws_batch()`.

log_group See `crew_definition_aws_batch()`.

config See `crew_definition_aws_batch()`.

credentials See `crew_definition_aws_batch()`.

endpoint See `crew_definition_aws_batch()`.

region See `crew_definition_aws_batch()`.

Methods

Public methods:

- `crew_class_definition_aws_batch$new()`
- `crew_class_definition_aws_batch$validate()`
- `crew_class_definition_aws_batch$register()`
- `crew_class_definition_aws_batch$deregister()`
- `crew_class_definition_aws_batch$describe()`
- `crew_class_definition_aws_batch$submit()`

Method new(): AWS Batch job definition constructor.

Usage:
```
crew_class_definition_aws_batch$new(
  job_queue = NULL,
  job_definition = NULL,
  log_group = NULL,
  config = NULL,
  credentials = NULL,
  endpoint = NULL,
  region = NULL
)
```

Arguments:

job_queue See `crew_definition_aws_batch()`.

job_definition See `crew_definition_aws_batch()`.

log_group See `crew_definition_aws_batch()`.

config See `crew_definition_aws_batch()`.

credentials See `crew_definition_aws_batch()`.

endpoint See `crew_definition_aws_batch()`.

region See [crew_definition_aws_batch()](#).

Returns: AWS Batch job definition object.

Method `validate()`: Validate the object.

Usage:

```
crew_class_definition_aws_batch$validate()
```

Returns: NULL (invisibly). Throws an error if a field is invalid.

Method `register()`: Register a job definition.

Usage:

```
crew_class_definition_aws_batch$register(
  image,
  platform_capabilities = "EC2",
  memory_units = "gigabytes",
  memory = NULL,
  cpus = NULL,
  gpus = NULL,
  seconds_timeout = NULL,
  scheduling_priority = NULL,
  tags = NULL,
  propagate_tags = NULL,
  parameters = NULL,
  job_role_arn = NULL,
  execution_role_arn = NULL
)
```

Arguments:

image Character of length 1, Docker image used for each job. You can supply a path to an
    image in Docker Hub or the full URI of an image in an Amazon ECR repository.

platform_capabilities Optional character of length 1, either `"EC2"` to run on EC2 or `"FARGATE"`
    to run on Fargate.

memory_units Character of length 1, either `"gigabytes"` or `"mebibytes"` to set the units of
    the `memory` argument. `"gigabytes"` is simpler for EC2 jobs, but Fargate has strict require-
    ments about specifying exact amounts of mebibytes (MiB). for details, read [https://docs.](https://docs.aws.amazon.com/cli/latest/reference/batch/register-job-definition.html)
    [aws.amazon.com/cli/latest/reference/batch/register-job-definition.html](https://docs.aws.amazon.com/cli/latest/reference/batch/register-job-definition.html) # no-
    lint

memory Positive numeric of length 1, amount of memory to request for each job.

cpus Positive numeric of length 1, number of virtual CPUs to request for each job.

gpus Positive numeric of length 1, number of GPUs to request for each job.

seconds_timeout Optional positive numeric of length 1, number of seconds until a job times
    out.

scheduling_priority Optional nonnegative integer of length 1 between `0` and `9999`, priority
    of jobs. Jobs with higher-valued priorities are scheduled first. The priority only applies if
    the job queue has a fair share policy. Set to `NULL` to omit.

tags Optional character vector of tags.

propagate_tags Optional logical of length 1, whether to propagate tags from the job or defi-
    nition to the ECS task.

parameters Optional character vector of key-value pairs designating parameters for job sub-mission.

job_role_arn Character of length 1, Amazon resource name (ARN) of the job role.

execution_role_arn Character of length 1, Amazon resource name (ARN) of the execution role.

Details: The register() method registers a simple job definition using the job definition name and log group originally supplied to crew_definition_aws_batch(). Job definitions created with $register() are container-based and use the AWS log driver. For more complicated kinds of jobs, we recommend skipping register(): first call https://www.paws-r-sdk.com/docs/batch_register_job_definition/ to register the job definition, then supply the job definition name to the job_definition argument of crew_definition_aws_batch().

Returns: A one-row tibble with the job definition name, ARN, and revision number of the registered job definition.

Method deregister(): Attempt to deregister a revision of the job definition.

Usage:

```
crew_class_definition_aws_batch$deregister(revision = NULL)
```

Arguments:

revision Finite positive integer of length 1, optional revision number to deregister. If NULL, then only the highest revision number of the job definition is deregistered, if it exists.

Details: Attempt to deregister the job definition whose name was originally supplied to the job_definition argument of crew_definition_aws_batch().

Returns: NULL (invisibly).

Method describe(): Describe the revisions of the job definition.

Usage:

```
crew_class_definition_aws_batch$describe(revision = NULL, active = FALSE)
```

Arguments:

revision Positive integer of length 1, optional revision number to describe.

active Logical of length 1, whether to filter on just the active job definition.

Returns: A tibble with job definition information. There is one row per revision. Some fields may be nested lists.

Method submit(): Submit an AWS Batch job with the given job definition.

Usage:

```
crew_class_definition_aws_batch$submit(
  command = c("sleep", "300"),
  name = paste0("crew-aws-batch-job-", crew::crew_random_name()),
  memory_units = "gigabytes",
  memory = NULL,
  cpus = NULL,
  gpus = NULL,
  seconds_timeout = NULL,
  share_identifier = NULL,
```

```
  scheduling_priority_override = NULL,
  tags = NULL,
  propagate_tags = NULL,
  parameters = NULL
)
```

Arguments:

command Character vector with the command to submit for the job. Usually a Linux shell
    command with each term in its own character string.

name Character of length 1 with the job name.

memory_units Character of length 1, either "gigabytes" or "mebibytes" to set the units of
    the memory argument. "gigabytes" is simpler for EC2 jobs, but Fargate has strict require-
    ments about specifying exact amounts of mebibytes (MiB). for details, read [https://docs.](https://docs.aws.amazon.com/cli/latest/reference/batch/register-job-definition.html)
    [aws.amazon.com/cli/latest/reference/batch/register-job-definition.html](https://docs.aws.amazon.com/cli/latest/reference/batch/register-job-definition.html) # no-
    lint

memory Positive numeric of length 1, amount of memory to request for each job.

cpus Positive numeric of length 1, number of virtual CPUs to request for each job.

gpus Positive numeric of length 1, number of GPUs to request for each job.

seconds_timeout Optional positive numeric of length 1, number of seconds until a job times
    out.

share_identifier Character of length 1 with the share identifier of the job. Only applies if
    the job queue has a scheduling policy. Read the official AWS Batch documentation for
    details.

scheduling_priority_override Optional nonnegative integer of length between 0 and 9999,
    priority of the job. This value overrides the priority in the job definition. Jobs with higher-
    valued priorities are scheduled first. The priority applies if the job queue has a fair share
    policy. Set to NULL to omit.

tags Optional character vector of tags.

propagate_tags Optional logical of length 1, whether to propagate tags from the job or defi-
    nition to the ECS task.

parameters Optional character vector of key-value pairs designating parameters for job sub-
    mission.

Details: This method uses the job queue and job definition that were supplied through [crew_definition_aws_batch().](crew_definition_aws_batch)
Any jobs submitted this way are different from the crew workers that the crew controller starts
automatically using the AWS Batch launcher plugin. You may use the submit() method in the
definition for different purposes such as testing.

Returns: A one-row tibble with the name, ID, and Amazon resource name (ARN) of the job.

See Also

Other definition: [crew_definition_aws_batch()](crew_definition_aws_batch)

crew_class_launcher_aws_batch

AWS Batch launcher class

Description

AWS Batch launcher R6 class

Details

See `crew_launcher_aws_batch()`.

IAM policies

In order for the AWS Batch `crew` plugin to function properly, your IAM policy needs permission to perform the `SubmitJob` and `TerminateJob` AWS Batch API calls. For more information on AWS policies and permissions, please visit https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html.

AWS arguments

The AWS Batch controller and launcher accept many arguments which start with `"aws_batch_"`. These arguments are AWS-Batch-specific parameters forwarded directly to the `submit_job()` method for the Batch client in the `paws.compute` R package

For a full description of each argument, including its meaning and format, please visit https://www.paws-r-sdk.com/docs/batch_submit_job/. The upstream API documentation is at https://docs.aws.amazon.com/batch/latest/APIReference/API_SubmitJob.html and the analogous CLI documentation is at https://docs.aws.amazon.com/cli/latest/reference/batch/submit-job.html.

The actual argument names may vary slightly, depending on which : for example, the `aws_batch_job_definition` argument of the `crew` AWS Batch launcher/controller corresponds to the `jobDefinition` argument of the web API and `paws.compute::batch()$submit_job()`, and both correspond to the `--job-definition` argument of the CLI.

Verbosity

Control verbosity with the `paws.log_level` global option in R. Set to 0 for minimum verbosity and 3 for maximum verbosity.

Super class

`crew::crew_class_launcher` -> `crew_class_launcher_aws_batch`

Active bindings

aws_batch_config See crew_launcher_aws_batch().

aws_batch_credentials See crew_launcher_aws_batch().

aws_batch_endpoint See crew_launcher_aws_batch().

aws_batch_region See crew_launcher_aws_batch().

aws_batch_job_definition See crew_launcher_aws_batch().

aws_batch_job_queue See crew_launcher_aws_batch().

aws_batch_share_identifier See crew_launcher_aws_batch().

aws_batch_scheduling_priority_override See crew_launcher_aws_batch().

aws_batch_parameters See crew_launcher_aws_batch().

aws_batch_container_overrides See crew_launcher_aws_batch().

aws_batch_node_overrides See crew_launcher_aws_batch().

aws_batch_retry_strategy See crew_launcher_aws_batch().

aws_batch_propagate_tags See crew_launcher_aws_batch().

aws_batch_timeout See crew_launcher_aws_batch().

aws_batch_tags See crew_launcher_aws_batch().

aws_batch_eks_properties_override See crew_launcher_aws_batch().

Methods

Public methods:

- crew_class_launcher_aws_batch$new()
- crew_class_launcher_aws_batch$validate()
- crew_class_launcher_aws_batch$launch_worker()
- crew_class_launcher_aws_batch$terminate_worker()

Method new(): Abstract launcher constructor.

Usage:

```
crew_class_launcher_aws_batch$new(
  name = NULL,
  seconds_interval = NULL,
  seconds_timeout = NULL,
  seconds_launch = NULL,
  seconds_idle = NULL,
  seconds_wall = NULL,
  tasks_max = NULL,
  tasks_timers = NULL,
  reset_globals = NULL,
  reset_packages = NULL,
  reset_options = NULL,
  garbage_collection = NULL,
  launch_max = NULL,
```

```
      tls = NULL,
      processes = NULL,
      r_arguments = NULL,
      aws_batch_config = NULL,
      aws_batch_credentials = NULL,
      aws_batch_endpoint = NULL,
      aws_batch_region = NULL,
      aws_batch_job_definition = NULL,
      aws_batch_job_queue = NULL,
      aws_batch_share_identifier = NULL,
      aws_batch_scheduling_priority_override = NULL,
      aws_batch_parameters = NULL,
      aws_batch_container_overrides = NULL,
      aws_batch_node_overrides = NULL,
      aws_batch_retry_strategy = NULL,
      aws_batch_propagate_tags = NULL,
      aws_batch_timeout = NULL,
      aws_batch_tags = NULL,
      aws_batch_eks_properties_override = NULL
    )
```

Arguments:

name See `crew_launcher_aws_batch()`.

seconds_interval See `crew_launcher_aws_batch()`.

seconds_timeout See `crew_launcher_aws_batch()`.

seconds_launch See `crew_launcher_aws_batch()`.

seconds_idle See `crew_launcher_aws_batch()`.

seconds_wall See `crew_launcher_aws_batch()`.

tasks_max See `crew_launcher_aws_batch()`.

tasks_timers See `crew_launcher_aws_batch()`.

reset_globals See `crew_launcher_aws_batch()`.

reset_packages See `crew_launcher_aws_batch()`.

reset_options See `crew_launcher_aws_batch()`.

garbage_collection See `crew_launcher_aws_batch()`.

launch_max See `crew_launcher_aws_batch()`.

tls See `crew_launcher_aws_batch()`.

processes See `crew_launcher_aws_batch()`.

r_arguments See `crew_launcher_aws_batch()`.

aws_batch_config See `crew_launcher_aws_batch()`.

aws_batch_credentials See `crew_launcher_aws_batch()`.

aws_batch_endpoint See `crew_launcher_aws_batch()`.

aws_batch_region See `crew_launcher_aws_batch()`.

aws_batch_job_definition See `crew_launcher_aws_batch()`.

aws_batch_job_queue See `crew_launcher_aws_batch()`.

aws_batch_share_identifier See `crew_launcher_aws_batch()`.

aws_batch_scheduling_priority_override See crew_launcher_aws_batch().

aws_batch_parameters See crew_launcher_aws_batch().

aws_batch_container_overrides See crew_launcher_aws_batch().

aws_batch_node_overrides See crew_launcher_aws_batch().

aws_batch_retry_strategy See crew_launcher_aws_batch().

aws_batch_propagate_tags See crew_launcher_aws_batch().

aws_batch_timeout See crew_launcher_aws_batch().

aws_batch_tags See crew_launcher_aws_batch().

aws_batch_eks_properties_override See crew_launcher_aws_batch().

Returns: An abstract launcher object.

Method `validate()`: Validate the launcher.

Usage:

`crew_class_launcher_aws_batch$validate()`

Returns: NULL (invisibly). Throws an error if a field is invalid.

Method `launch_worker()`: Launch a local process worker which will dial into a socket.

Usage:

```
crew_class_launcher_aws_batch$launch_worker(
  call,
  name,
  launcher,
  worker,
  instance
)
```

Arguments:

call Character of length 1, a namespaced call to crew::crew_worker() which will run in the worker and accept tasks.

name Character of length 1, an informative worker name.

launcher Character of length 1, name of the launcher.

worker Positive integer of length 1, index of the worker. This worker index remains the same even when the current instance of the worker exits and a new instance launches. It is always between 1 and the maximum number of concurrent workers.

instance Character of length 1 to uniquely identify the current instance of the worker.

Details: The `call` argument is R code that will run to initiate the worker.

Returns: A handle object to allow the termination of the worker later on.

Method `terminate_worker()`: Terminate a local process worker.

Usage:

`crew_class_launcher_aws_batch$terminate_worker(handle)`

Arguments:

handle A process handle object previously returned by `launch_worker()`.

Returns: NULL (invisibly).

See Also

Other plugin_aws_batch: `crew_controller_aws_batch()`, `crew_launcher_aws_batch()`

---

crew_class_monitor_aws_batch

AWS Batch monitor class

Description

AWS Batch monitor R6 class

Details

See `crew_monitor_aws_batch()`.

IAM policies

In order for the AWS Batch `crew` monitor class to function properly, your IAM policy needs permission to perform the `SubmitJob`, `TerminateJob`, `ListJobs`, and `DescribeJobs` AWS Batch API calls. In addition, to download CloudWatch logs with the `log()` method, your IAM policy also needs permission to perform the `GetLogEvents` CloudWatch logs API call. For more information on AWS policies and permissions, please visit `https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html`.

Active bindings

job_queue See `crew_monitor_aws_batch()`.

job_definition See `crew_monitor_aws_batch()`.

log_group See `crew_monitor_aws_batch()`.

config See `crew_monitor_aws_batch()`.

credentials See `crew_monitor_aws_batch()`.

endpoint See `crew_monitor_aws_batch()`.

region See `crew_monitor_aws_batch()`.

Methods

Public methods:

- `crew_class_monitor_aws_batch$new()`
- `crew_class_monitor_aws_batch$validate()`
- `crew_class_monitor_aws_batch$terminate()`
- `crew_class_monitor_aws_batch$status()`
- `crew_class_monitor_aws_batch$log()`
- `crew_class_monitor_aws_batch$jobs()`
- `crew_class_monitor_aws_batch$active()`

- [crew_class_monitor_aws_batch$inactive()](#)
- [crew_class_monitor_aws_batch$submitted()](#)
- [crew_class_monitor_aws_batch$pending()](#)
- [crew_class_monitor_aws_batch$runnable()](#)
- [crew_class_monitor_aws_batch$starting()](#)
- [crew_class_monitor_aws_batch$running()](#)
- [crew_class_monitor_aws_batch$succeeded()](#)
- [crew_class_monitor_aws_batch$failed()](#)

Method `new()`: AWS Batch job definition constructor.

Usage:
```
crew_class_monitor_aws_batch$new(
  job_queue = NULL,
  job_definition = NULL,
  log_group = NULL,
  config = NULL,
  credentials = NULL,
  endpoint = NULL,
  region = NULL
)
```

Arguments:

`job_queue` See [crew_monitor_aws_batch()](#).

`job_definition` See [crew_monitor_aws_batch()](#).

`log_group` See [crew_monitor_aws_batch()](#).

`config` See [crew_monitor_aws_batch()](#).

`credentials` See [crew_monitor_aws_batch()](#).

`endpoint` See [crew_monitor_aws_batch()](#).

`region` See [crew_monitor_aws_batch()](#).

Returns: AWS Batch job definition object.

Method `validate()`: Validate the object.

Usage:
```
crew_class_monitor_aws_batch$validate()
```

Returns: NULL (invisibly). Throws an error if a field is invalid.

Method `terminate()`: Terminate one or more AWS Batch jobs.

Usage:
```
crew_class_monitor_aws_batch$terminate(
  ids = NULL,
  all = FALSE,
  reason = "cancelled/terminated by crew.aws.batch monitor",
  verbose = TRUE
)
```

Arguments:

ids Character vector with the IDs of the AWS Batch jobs to terminate. Leave as NULL if all is
TRUE.

all TRUE to terminate all jobs belonging to the previously specified job definition. FALSE to
terminate only the job IDs given in the ids argument.

reason Character of length 1, natural language explaining the reason the job was terminated.

verbose Logical of length 1, whether to show a progress bar if the R process is interactive and
length(ids) is greater than 1.

Returns: NULL (invisibly).

Method status(): Get the status of a single job

Usage:

crew_class_monitor_aws_batch$status(id)

Arguments:

id Character of length 1, job ID. This is different from the user-supplied job name.

Returns: A one-row tibble with information about the job.

Method log(): Get the CloudWatch log of a job.

Usage:

crew_class_monitor_aws_batch$log(id, start_from_head = FALSE)

Arguments:

id Character of length 1, job ID. This is different from the user-supplied job name.

start_from_head Logical of length 1, whether to print earlier log events before later ones.

Details: This method assumes the job has log driver "awslogs" (specifying AWS CloudWatch)
and that the log group is the one prespecified in the log_group argument of crew_monitor_aws_batch().
This method cannot use other log drivers such as Splunk, and it will fail if the log group is wrong
or missing.

Returns: A tibble with log information.

Method jobs(): List all the jobs in the given job queue with the given job definition.

Usage:

```
crew_class_monitor_aws_batch$jobs(
  status = c("submitted", "pending", "runnable", "starting", "running", "succeeded",
    "failed")
)
```

Arguments:

status Character vector of job states. Results are limited to these job states.

Details: The output only includes jobs under the job queue and job definition that were supplied
through crew_monitor_aws_batch().

Returns: A tibble with one row per job and columns with job information.

Method active(): List active jobs: submitted, pending, runnable, starting, or running (not
succeeded or failed).

Usage:

```
crew_class_monitor_aws_batch$active()
```

Details: The output only includes jobs under the job queue and job definition that were supplied through crew_monitor_aws_batch().

Returns: A tibble with one row per job and columns with job information.

Method inactive(): List inactive jobs: ones whose status is succeeded or failed (not submitted, pending, runnable, starting, or running).

Usage:

```
crew_class_monitor_aws_batch$inactive()
```

Details: The output only includes jobs under the job queue and job definition that were supplied through crew_monitor_aws_batch().

Returns: A tibble with one row per job and columns with job information.

Method submitted(): List jobs whose status is "submitted".

Usage:

```
crew_class_monitor_aws_batch$submitted()
```

Details: The output only includes jobs under the job queue and job definition that were supplied through crew_monitor_aws_batch().

Returns: A tibble with one row per job and columns with job information.

Method pending(): List jobs whose status is "pending".

Usage:

```
crew_class_monitor_aws_batch$pending()
```

Details: The output only includes jobs under the job queue and job definition that were supplied through crew_monitor_aws_batch().

Returns: A tibble with one row per job and columns with job information.

Method runnable(): List jobs whose status is "runnable".

Usage:

```
crew_class_monitor_aws_batch$runnable()
```

Details: The output only includes jobs under the job queue and job definition that were supplied through crew_monitor_aws_batch().

Returns: A tibble with one row per job and columns with job information.

Method starting(): List jobs whose status is "starting".

Usage:

```
crew_class_monitor_aws_batch$starting()
```

Details: The output only includes jobs under the job queue and job definition that were supplied through crew_monitor_aws_batch().

Returns: A tibble with one row per job and columns with job information.

Method running(): List jobs whose status is "running".

Usage:

```
crew_class_monitor_aws_batch$running()
```

Details: The output only includes jobs under the job queue and job definition that were supplied through [crew_monitor_aws_batch()](#).

Returns: A `tibble` with one row per job and columns with job information.

Method `succeeded()`: List jobs whose status is `"succeeded"`.

Usage:

```
crew_class_monitor_aws_batch$succeeded()
```

Details: The output only includes jobs under the job queue and job definition that were supplied through [crew_monitor_aws_batch()](#).

Returns: A `tibble` with one row per job and columns with job information.

Method `failed()`: List jobs whose status is `"failed"`.

Usage:

```
crew_class_monitor_aws_batch$failed()
```

Details: The output only includes jobs under the job queue and job definition that were supplied through [crew_monitor_aws_batch()](#).

Returns: A `tibble` with one row per job and columns with job information.

See Also

Other monitor: [crew_monitor_aws_batch()](#)

---

crew_controller_aws_batch

Create a controller with an AWS Batch launcher.

Description

Create an R6 object to submit tasks and launch workers on AWS Batch workers.

Usage

```
crew_controller_aws_batch(
  name = NULL,
  workers = 1L,
  host = NULL,
  port = NULL,
  tls = crew::crew_tls(mode = "automatic"),
  tls_enable = NULL,
  tls_config = NULL,
  seconds_interval = 0.5,
  seconds_timeout = 60,
  seconds_launch = 1800,
```

```
    seconds_idle = Inf,
    seconds_wall = Inf,
    retry_tasks = TRUE,
    log_resources = NULL,
    tasks_max = Inf,
    tasks_timers = 0L,
    reset_globals = TRUE,
    reset_packages = FALSE,
    reset_options = FALSE,
    garbage_collection = FALSE,
    launch_max = 5L,
    processes = NULL,
    r_arguments = c("--no-save", "--no-restore"),
    aws_batch_config = list(),
    aws_batch_credentials = list(),
    aws_batch_endpoint = NULL,
    aws_batch_region = NULL,
    aws_batch_job_definition,
    aws_batch_job_queue,
    aws_batch_share_identifier = NULL,
    aws_batch_scheduling_priority_override = NULL,
    aws_batch_parameters = NULL,
    aws_batch_container_overrides = NULL,
    aws_batch_node_overrides = NULL,
    aws_batch_retry_strategy = NULL,
    aws_batch_propagate_tags = NULL,
    aws_batch_timeout = NULL,
    aws_batch_tags = NULL,
    aws_batch_eks_properties_override = NULL
)
```

Arguments

name	Name of the client object. If NULL, a name is automatically generated.
workers	Integer, maximum number of parallel workers to run.
host	IP address of the mirai client to send and receive tasks. If NULL, the host defaults to the local IP address.
port	TCP port to listen for the workers. If NULL, then an available ephemeral port is automatically chosen.
tls	A TLS configuration object from [crew_tls()]().
tls_enable	Deprecated on 2023-09-15 in version 0.4.1. Use argument tls instead.
tls_config	Deprecated on 2023-09-15 in version 0.4.1. Use argument tls instead.
seconds_interval	
	Number of seconds between polling intervals waiting for certain internal synchronous operations to complete, such as checking mirai::status()
seconds_timeout	
	Number of seconds until timing out while waiting for certain synchronous operations to complete, such as checking mirai::status().

seconds_launch   Seconds of startup time to allow. A worker is unconditionally assumed to be alive from the moment of its launch until seconds_launch seconds later. After seconds_launch seconds, the worker is only considered alive if it is actively connected to its assign websocket.

seconds_idle   Maximum number of seconds that a worker can idle since the completion of the last task. If exceeded, the worker exits. But the timer does not launch until tasks_timers tasks have completed. See the idletime argument of mirai::daemon(). crew does not excel with perfectly transient workers because it does not micromanage the assignment of tasks to workers, so please allow enough idle time for a new worker to be delegated a new task.

seconds_wall   Soft wall time in seconds. The timer does not launch until tasks_timers tasks have completed. See the walltime argument of mirai::daemon().

retry_tasks   TRUE to automatically retry a task in the event of an unexpected worker exit. FALSE to give up on the first exit and return a mirai error code (code number 19). TRUE (default) is recommended in most situations. Use FALSE for debugging purposes, e.g. to confirm that a task is causing a worker to run out of memory or crash in some other way.

log_resources   Optional character string with a file path to a text file to log memory consumption. Set log_resources to NULL to avoid writing to a log file. If you supply a path, then the log() method will write memory usage statistics to the file, and most controller methods will do the same with throttling so resource consumption is recorded throughout the whole life cycle of the controller.

The log file is in comma-separated values (CSV) format which can be easily read by readr::read_csv(). The controller automatically deletes the old log file when it starts (when controller$start() is called for the first time, but not subsequent times).

The log file has one row per observation of a process, including the current R process ("client") and the mirai dispatcher. If the dispatcher is not included in the output, it means the dispatcher process is not running. Columns include: * type: the type of process (client or dispatcher) * pid: the process ID. * status: The process status (from ps::ps_status()). * rss: resident set size (RSS). RS is the total memory held by a process, including shared libraries which may also be in use by other processes. RSS is obtained from ps::ps_memory_info() and shown in bytes. * elapsed: number of elapsed seconds since the R process was started (from proc.time()["elapsed"]).

tasks_max   Maximum number of tasks that a worker will do before exiting. See the maxtasks argument of mirai::daemon(). crew does not excel with perfectly transient workers because it does not micromanage the assignment of tasks to workers, it is recommended to set tasks_max to a value greater than 1.

tasks_timers   Number of tasks to do before activating the timers for seconds_idle and seconds_wall. See the timerstart argument of mirai::daemon().

reset_globals   TRUE to reset global environment variables between tasks, FALSE to leave them alone.

reset_packages   TRUE to unload any packages loaded during a task (runs between each task), FALSE to leave packages alone.

reset_options   TRUE to reset global options to their original state between each task, FALSE oth-
                  erwise. It is recommended to only set reset_options = TRUE if reset_packages
                  is also TRUE because packages sometimes rely on options they set at loading
                  time.

garbage_collection
                  TRUE to run garbage collection between tasks, FALSE to skip.

launch_max   Positive integer of length 1, maximum allowed consecutive launch attempts
                  which do not complete any tasks. Enforced on a worker-by-worker basis. The
                  futile launch count resets to back 0 for each worker that completes a task. It is
                  recommended to set launch_max above 0 because sometimes workers are un-
                  productive under perfectly ordinary circumstances. But launch_max should still
                  be small enough to detect errors in the underlying platform.

processes   NULL or positive integer of length 1, number of local processes to launch to allow
                  worker launches to happen asynchronously. If NULL, then no local processes are
                  launched. If 1 or greater, then the launcher starts the processes on start()
                  and ends them on terminate(). Plugins that may use these processes should
                  run asynchronous calls using launcher$async$eval() and expect a mirai task
                  object as the return value.

r_arguments   Optional character vector of command line arguments to pass to R when starting
                  a worker. Example: r_arguments = c("--vanilla", "--max-connections=32").

aws_batch_config
                  Named list, config argument of paws.compute::batch() with optional con-
                  figuration details.

aws_batch_credentials
                  Named list. credentials argument of paws.compute::batch() with optional
                  credentials (if not already provided through environment variables such as AWS_ACCESS_KEY_ID).

aws_batch_endpoint
                  Character of length 1. endpoint argument of paws.compute::batch() with
                  the endpoint to send HTTP requests.

aws_batch_region
                  Character of length 1. region argument of paws.compute::batch() with an
                  AWS region string such as "us-east-2".

aws_batch_job_definition
                  Character of length 1, name of the AWS Batch job definition to use. There is no
                  default for this argument, and a job definition must be created prior to running
                  the controller. Please see https://docs.aws.amazon.com/batch/ for details.

                  To create a job definition, you will need to create a Docker-compatible image
                  which can run R and crew. You may which to inherit from the images at https:
                  //github.com/rocker-org/rocker-versioned2.

aws_batch_job_queue
                  Character of length 1, name of the AWS Batch job queue to use. There is no
                  default for this argument, and a job queue must be created prior to running the
                  controller. Please see https://docs.aws.amazon.com/batch/ for details.

aws_batch_share_identifier
                  NULL or character of length 1. For details, visit https://www.paws-r-sdk.
                  com/docs/batch_submit_job/ and the "AWS arguments" sections of this help
                  file.

aws_batch_scheduling_priority_override

        NULL or integer of length 1. For details, visit https://www.paws-r-sdk.com/docs/batch_submit_job/ and the "AWS arguments" sections of this help file.

aws_batch_parameters

        NULL or a nonempty list. For details, visit https://www.paws-r-sdk.com/docs/batch_submit_job/ and the "AWS arguments" sections of this help file.

aws_batch_container_overrides

        NULL or a nonempty named list of fields to override in the container specified in the job definition. Any overrides for the command field are ignored because crew.aws.batch needs to override the command to run the crew worker. For more details, visit https://www.paws-r-sdk.com/docs/batch_submit_job/ and the "AWS arguments" sections of this help file.

aws_batch_node_overrides

        NULL or a nonempty named list. For more details, visit https://www.paws-r-sdk.com/docs/batch_submit_job/ and the "AWS arguments" sections of this help file.

aws_batch_retry_strategy

        NULL or a nonempty named list. For more details, visit https://www.paws-r-sdk.com/docs/batch_submit_job/ and the "AWS arguments" sections of this help file.

aws_batch_propagate_tags

        NULL or a logical of length 1. For more details, visit https://www.paws-r-sdk.com/docs/batch_submit_job/ and the "AWS arguments" sections of this help file.

aws_batch_timeout

        NULL or a nonempty named list. For more details, visit https://www.paws-r-sdk.com/docs/batch_submit_job/ and the "AWS arguments" sections of this help file.

aws_batch_tags NULL or a nonempty named list. For more details, visit https://www.paws-r-sdk.com/docs/batch_submit_job/ and the "AWS arguments" sections of this help file.

aws_batch_eks_properties_override

        NULL or a nonempty named list. For more details, visit https://www.paws-r-sdk.com/docs/batch_submit_job/ and the "AWS arguments" sections of this help file.

IAM policies

In order for the AWS Batch crew plugin to function properly, your IAM policy needs permission to perform the SubmitJob and TerminateJob AWS Batch API calls. For more information on AWS policies and permissions, please visit https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html.

AWS arguments

The AWS Batch controller and launcher accept many arguments which start with "aws_batch_". These arguments are AWS-Batch-specific parameters forwarded directly to the submit_job() method for the Batch client in the paws.compute R package

For a full description of each argument, including its meaning and format, please visit https://www.paws-r-sdk.com/docs/batch_submit_job/. The upstream API documentation is at https://docs.aws.amazon.com/batch/latest/APIReference/API_SubmitJob.html and the analogous CLI documentation is at https://docs.aws.amazon.com/cli/latest/reference/batch/submit-job.html.

The actual argument names may vary slightly, depending on which : for example, the aws_batch_job_definition argument of the crew AWS Batch launcher/controller corresponds to the jobDefinition argument of the web API and paws.compute::batch()$submit_job(), and both correspond to the --job-definition argument of the CLI.

Verbosity

Control verbosity with the paws.log_level global option in R. Set to 0 for minimum verbosity and 3 for maximum verbosity.

See Also

Other plugin_aws_batch: crew_class_launcher_aws_batch, crew_launcher_aws_batch()

Examples

```
if (identical(Sys.getenv("CREW_EXAMPLES"), "true")) {
controller <- crew_controller_aws_batch(
  aws_batch_job_definition = "YOUR_JOB_DEFINITION_NAME",
  aws_batch_job_queue = "YOUR_JOB_QUEUE_NAME"
)
controller$start()
controller$push(name = "task", command = sqrt(4))
controller$wait()
controller$pop()$result
controller$terminate()
}
```

crew_definition_aws_batch

Create an AWS Batch job definition object.

Description

Create an R6 object to manage a job definition for AWS Batch jobs.

Usage

```
crew_definition_aws_batch(
  job_queue,
 job_definition = paste0("crew-aws-batch-job-definition-", crew::crew_random_name()),
  log_group = "/aws/batch/job",
  config = NULL,
```

```
    credentials = NULL,
    endpoint = NULL,
    region = NULL
)
```

Arguments

job_queue	Character of length 1, name of the AWS Batch job queue.
job_definition	Character of length 1, name of the AWS Batch job definition. The job definition might or might not exist at the time crew_definition_aws_batch() is called. Either way is fine.
log_group	Character of length 1, AWS Batch CloudWatch log group to get job logs. The default log group is often "/aws/batch/job", but not always. It is not easy to get the log group of an active job or job definition, so if you have a non-default log group and you do not know its name, please consult your system administrator.
config	Optional named list, config argument of paws.compute::batch() with optional configuration details.
credentials	Optional named list. credentials argument of paws.compute::batch() with optional credentials (if not already provided through environment variables such as AWS_ACCESS_KEY_ID).
endpoint	Optional character of length 1. endpoint argument of paws.compute::batch() with the endpoint to send HTTP requests.
region	Character of length 1. region argument of paws.compute::batch() with an AWS region string such as "us-east-2". Serves as the region for both AWS Batch and CloudWatch. Tries to default to paws.common::get_config()$region, then to Sys.getenv("AWS_REGION") if unsuccessful, then Sys.getenv("AWS_REGION"), then Sys.getenv("AWS_DEFAULT_REGION").

Value

An R6 job definition object.

IAM policies

In order for the AWS Batch crew job definition class to function properly, your IAM policy needs permission to perform the RegisterJobDefinition, DeregisterJobDefinition, and DescribeJobDefinitions AWS Batch API calls. For more information on AWS policies and permissions, please visit https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html.

See Also

Other definition: crew_class_definition_aws_batch

---

crew_launcher_aws_batch

Create an AWS Batch launcher object.

Description

Create an R6 AWS Batch launcher object.

Usage

```
crew_launcher_aws_batch(
  name = NULL,
  seconds_interval = 0.5,
  seconds_timeout = 60,
  seconds_launch = 1800,
  seconds_idle = Inf,
  seconds_wall = Inf,
  tasks_max = Inf,
  tasks_timers = 0L,
  reset_globals = TRUE,
  reset_packages = FALSE,
  reset_options = FALSE,
  garbage_collection = FALSE,
  launch_max = 5L,
  tls = crew::crew_tls(mode = "automatic"),
  processes = NULL,
  r_arguments = c("--no-save", "--no-restore"),
  aws_batch_config = list(),
  aws_batch_credentials = list(),
  aws_batch_endpoint = NULL,
  aws_batch_region = NULL,
  aws_batch_job_definition,
  aws_batch_job_queue,
  aws_batch_share_identifier = NULL,
  aws_batch_scheduling_priority_override = NULL,
  aws_batch_parameters = NULL,
  aws_batch_container_overrides = NULL,
  aws_batch_node_overrides = NULL,
  aws_batch_retry_strategy = NULL,
  aws_batch_propagate_tags = NULL,
  aws_batch_timeout = NULL,
  aws_batch_tags = NULL,
  aws_batch_eks_properties_override = NULL
)
```

Arguments

`name`	Name of the launcher.
`seconds_interval`	
	Number of seconds between polling intervals waiting for certain internal synchronous operations to complete, such as checking `mirai::status()`.
`seconds_timeout`	
	Number of seconds until timing out while waiting for certain synchronous operations to complete, such as checking `mirai::status()`.
`seconds_launch`	Seconds of startup time to allow. A worker is unconditionally assumed to be alive from the moment of its launch until `seconds_launch` seconds later. After `seconds_launch` seconds, the worker is only considered alive if it is actively connected to its assign websocket.
`seconds_idle`	Maximum number of seconds that a worker can idle since the completion of the last task. If exceeded, the worker exits. But the timer does not launch until `tasks_timers` tasks have completed. See the `idletime` argument of `mirai::daemon()`. crew does not excel with perfectly transient workers because it does not micromanage the assignment of tasks to workers, so please allow enough idle time for a new worker to be delegated a new task.
`seconds_wall`	Soft wall time in seconds. The timer does not launch until `tasks_timers` tasks have completed. See the `walltime` argument of `mirai::daemon()`.
`tasks_max`	Maximum number of tasks that a worker will do before exiting. See the `maxtasks` argument of `mirai::daemon()`. crew does not excel with perfectly transient workers because it does not micromanage the assignment of tasks to workers, it is recommended to set `tasks_max` to a value greater than 1.
`tasks_timers`	Number of tasks to do before activating the timers for `seconds_idle` and `seconds_wall`. See the `timerstart` argument of `mirai::daemon()`.
`reset_globals`	TRUE to reset global environment variables between tasks, FALSE to leave them alone.
`reset_packages`	TRUE to unload any packages loaded during a task (runs between each task), FALSE to leave packages alone.
`reset_options`	TRUE to reset global options to their original state between each task, FALSE otherwise. It is recommended to only set `reset_options = TRUE` if `reset_packages` is also TRUE because packages sometimes rely on options they set at loading time.
`garbage_collection`	
	TRUE to run garbage collection between tasks, FALSE to skip.
`launch_max`	Positive integer of length 1, maximum allowed consecutive launch attempts which do not complete any tasks. Enforced on a worker-by-worker basis. The futile launch count resets to back 0 for each worker that completes a task. It is recommended to set `launch_max` above 0 because sometimes workers are unproductive under perfectly ordinary circumstances. But `launch_max` should still be small enough to detect errors in the underlying platform.
`tls`	A TLS configuration object from [crew_tls()]().

processes   NULL or positive integer of length 1, number of local processes to launch to allow
                   worker launches to happen asynchronously. If NULL, then no local processes are
                   launched. If 1 or greater, then the launcher starts the processes on start()
                   and ends them on terminate(). Plugins that may use these processes should
                   run asynchronous calls using launcher$async$eval() and expect a mirai task
                   object as the return value.

r_arguments   Optional character vector of command line arguments to pass to R when starting
                   a worker. Example: r_arguments = c("--vanilla", "--max-connections=32").

aws_batch_config
                   Named list, config argument of paws.compute::batch() with optional con-
                   figuration details.

aws_batch_credentials
                   Named list. credentials argument of paws.compute::batch() with optional
                   credentials (if not already provided through environment variables such as AWS_ACCESS_KEY_ID).

aws_batch_endpoint
                   Character of length 1. endpoint argument of paws.compute::batch() with
                   the endpoint to send HTTP requests.

aws_batch_region
                   Character of length 1. region argument of paws.compute::batch() with an
                   AWS region string such as "us-east-2".

aws_batch_job_definition
                   Character of length 1, name of the AWS Batch job definition to use. There is no
                   default for this argument, and a job definition must be created prior to running
                   the controller. Please see https://docs.aws.amazon.com/batch/ for details.

                   To create a job definition, you will need to create a Docker-compatible image
                   which can run R and crew. You may which to inherit from the images at https:
                   //github.com/rocker-org/rocker-versioned2.

aws_batch_job_queue
                   Character of length 1, name of the AWS Batch job queue to use. There is no
                   default for this argument, and a job queue must be created prior to running the
                   controller. Please see https://docs.aws.amazon.com/batch/ for details.

aws_batch_share_identifier
                   NULL or character of length 1. For details, visit https://www.paws-r-sdk.
                   com/docs/batch_submit_job/ and the "AWS arguments" sections of this help
                   file.

aws_batch_scheduling_priority_override
                   NULL or integer of length 1. For details, visit https://www.paws-r-sdk.com/
                   docs/batch_submit_job/ and the "AWS arguments" sections of this help file.

aws_batch_parameters
                   NULL or a nonempty list. For details, visit https://www.paws-r-sdk.com/
                   docs/batch_submit_job/ and the "AWS arguments" sections of this help file.

aws_batch_container_overrides
                   NULL or a nonempty named list of fields to override in the container speci-
                   fied in the job definition. Any overrides for the command field are ignored be-
                   cause crew.aws.batch needs to override the command to run the crew worker.
                   For more details, visit https://www.paws-r-sdk.com/docs/batch_submit_
                   job/ and the "AWS arguments" sections of this help file.

aws_batch_node_overrides

> NULL or a nonempty named list. For more details, visit https://www.paws-r-sdk.com/docs/batch_submit_job/ and the "AWS arguments" sections of this help file.

aws_batch_retry_strategy

> NULL or a nonempty named list. For more details, visit https://www.paws-r-sdk.com/docs/batch_submit_job/ and the "AWS arguments" sections of this help file.

aws_batch_propagate_tags

> NULL or a logical of length 1. For more details, visit https://www.paws-r-sdk.com/docs/batch_submit_job/ and the "AWS arguments" sections of this help file.

aws_batch_timeout

> NULL or a nonempty named list. For more details, visit https://www.paws-r-sdk.com/docs/batch_submit_job/ and the "AWS arguments" sections of this help file.

aws_batch_tags NULL or a nonempty named list. For more details, visit https://www.paws-r-sdk.com/docs/batch_submit_job/ and the "AWS arguments" sections of this help file.

aws_batch_eks_properties_override

> NULL or a nonempty named list. For more details, visit https://www.paws-r-sdk.com/docs/batch_submit_job/ and the "AWS arguments" sections of this help file.

Value

An R6 AWS Batch launcher object.

IAM policies

In order for the AWS Batch crew plugin to function properly, your IAM policy needs permission to perform the SubmitJob and TerminateJob AWS Batch API calls. For more information on AWS policies and permissions, please visit https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html.

AWS arguments

The AWS Batch controller and launcher accept many arguments which start with "aws_batch_". These arguments are AWS-Batch-specific parameters forwarded directly to the submit_job() method for the Batch client in the paws.compute R package

For a full description of each argument, including its meaning and format, please visit https://www.paws-r-sdk.com/docs/batch_submit_job/. The upstream API documentation is at https://docs.aws.amazon.com/batch/latest/APIReference/API_SubmitJob.html and the analogous CLI documentation is at https://docs.aws.amazon.com/cli/latest/reference/batch/submit-job.html.

The actual argument names may vary slightly, depending on which : for example, the aws_batch_job_definition argument of the crew AWS Batch launcher/controller corresponds to the jobDefinition argument of the web API and paws.compute::batch()$submit_job(), and both correspond to the --job-definition argument of the CLI.

Verbosity

Control verbosity with the paws.log_level global option in R. Set to 0 for minimum verbosity and 3 for maximum verbosity.

See Also

Other plugin_aws_batch: crew_class_launcher_aws_batch, crew_controller_aws_batch()

---

crew_monitor_aws_batch

Create an AWS Batch monitor object.

Description

Create an R6 object to list, inspect, and terminate AWS Batch jobs.

Usage

```
crew_monitor_aws_batch(
  job_queue,
  job_definition,
  log_group = "/aws/batch/job",
  config = NULL,
  credentials = NULL,
  endpoint = NULL,
  region = NULL
)
```

Arguments

job_queue	Character of length 1, name of the AWS Batch job queue.
job_definition	Character of length 1, name of the AWS Batch job definition.
log_group	Character of length 1, AWS Batch CloudWatch log group to get job logs. The default log group is often "/aws/batch/job", but not always. It is not easy to get the log group of an active job or job definition, so if you have a non-default log group and you do not know its name, please consult your system administrator.
config	Optional named list, config argument of paws.compute::batch() with optional configuration details.
credentials	Optional named list. credentials argument of paws.compute::batch() with optional credentials (if not already provided through environment variables such as AWS_ACCESS_KEY_ID).
endpoint	Optional character of length 1. endpoint argument of paws.compute::batch() with the endpoint to send HTTP requests.

region   Character of length 1. `region` argument of `paws.compute::batch()` with an AWS region string such as `"us-east-2"`. Serves as the region for both AWS Batch and CloudWatch. Tries to default to `paws.common::get_config()$region`, then to `Sys.getenv("AWS_REGION")` if unsuccessful, then `Sys.getenv("AWS_REGION")`, then `Sys.getenv("AWS_DEFAULT_REGION")`.

IAM policies

In order for the AWS Batch `crew` monitor class to function properly, your IAM policy needs permission to perform the `SubmitJob`, `TerminateJob`, `ListJobs`, and `DescribeJobs` AWS Batch API calls. In addition, to download CloudWatch logs with the `log()` method, your IAM policy also needs permission to perform the `GetLogEvents` CloudWatch logs API call. For more information on AWS policies and permissions, please visit https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html.

See Also

Other monitor: crew_class_monitor_aws_batch

Index