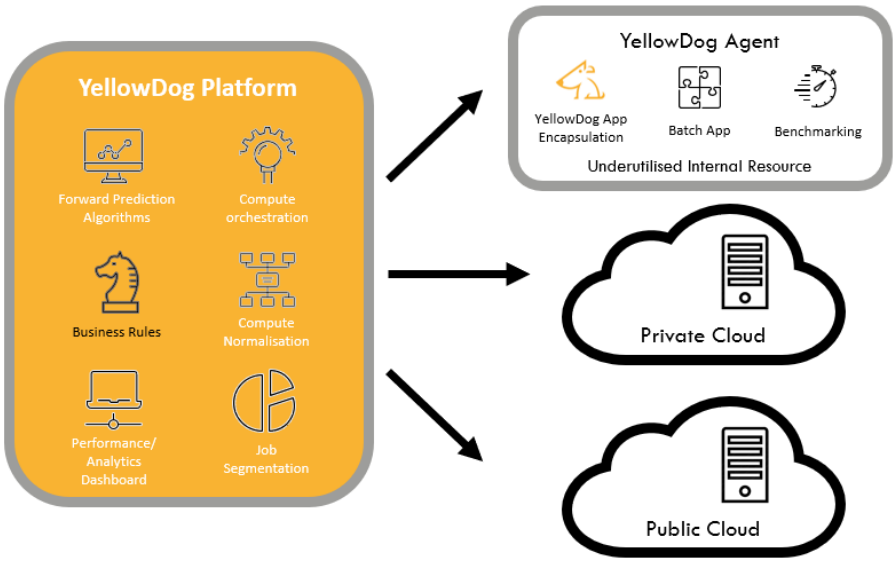
YellowDog Scheduler Module

The YellowDog Scheduler module is part of a suite of modules which constitute the YellowDog Platform. The module is the main route for all customers to define their jobs and how they want them to be processed. It supports set business rules which determine where jobs will be processed across both single or multiple compute sources. The processing destinations can be on premise or cloud or a combination of both and are configured through a simple, flexible and powerful interface.

The Scheduler Module can be hosted by YellowDog as part of the core hosted YellowDog service or be integrated in YellowDog's on premise solutions. The module is also scalable; able to run in a cluster to provide high throughput and availability.

The YellowDog Scheduler module works alongside the YellowDog Compute module and is accessed via a clean, straightforward and secure API.



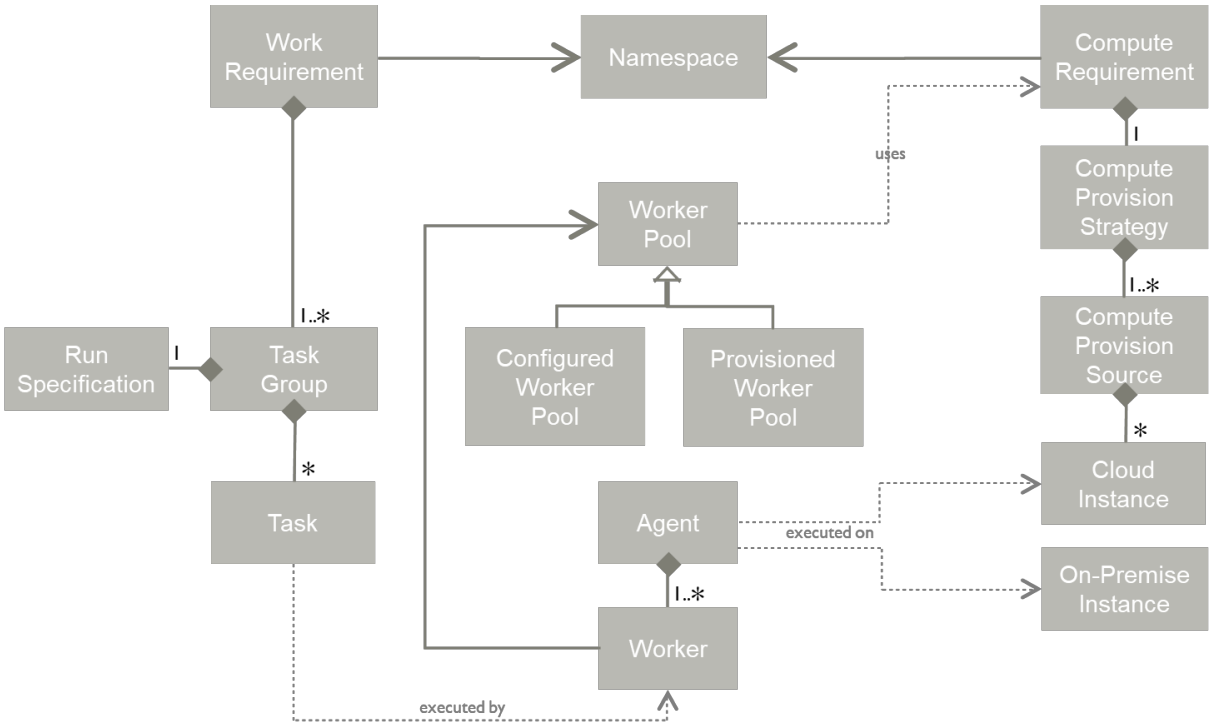
Overview

The YellowDog Scheduler has several key concepts however in summary: -

Work Requirements are submitted to the YellowDog Scheduler via the YellowDog Client API. Work Requirements contain Tasks, Task Groups and Run Specifications which in turn specify which Workers which should execute these tasks, how the Workers are claimed, released and shared; and how the number of Workers are maintained as the compute resource available changes.

The Workers are instantiated by the YellowDog Agent; the number started is defined by configuration per agent. The Agent itself runs on configured (on-premise) or provisioned (cloud) instances. The provisioning and automatic agent configuration of cloud instances is managed via a Compute Requirement which details provisioning strategies.

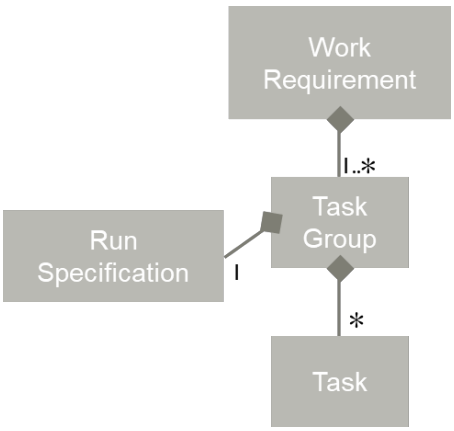
A Namespace (analogous to a Project) is a user defined way to group Work and Compute Requirements together.



YellowDog Scheduler resources

Work Requirements

A Work Requirement is a collection of groups of tasks to be executed and the technical specification of the instances and images that these tasks should be executed on. A Work Requirement is submitted to the YellowDog Scheduler for processing via the YellowDog Client API. The API can also be used to query the status of the Work Requirement. A Work Requirement can be "pending" until enough workers to fulfil the Work Requirement can be claimed. As well as a user-defined name, users can optionally set tags on Work Requirements to identify them and add further information.



Tasks:

Tasks have a task type and initialisation data for the Worker performing the task. The YellowDog Scheduler will only allocate tasks to Workers that support the same task type to execute these tasks. As well as a user-defined name, users can optionally set tags on Tasks to identify them and add further information.

Task Groups:

Tasks are grouped into Task Groups along with the Run Specification for those tasks. When starting a Work Requirement, the Scheduler waits until all Task Groups are ready to start and then starts them all at the same time. As well as a user-defined name, users can optionally set tags on Task Groups to identify them and add further information.

Run Specification:

The Run Specification contains a list of acceptable Machine Configurations (Instance Type and Image ID) required to execute the tasks. It also allows the tasks to be constrained to execute on particular cloud providers and/or regions.

The Run Specification defines the Task Run Type for the task queue which defines how many Workers are required:

Task Run Type

- **Queued.** This run type allows an ideal and minimum number of Workers to be set. The Scheduler will try and claim the ideal number of Workers but will not consider the Task Group ready for processing until at least the minimum is claimed.
- **Start-up.** The Scheduler will claim the same number of Workers as there are Tasks in the Task Group. The Task Group is ready for processing when it has a Worker for each Task.

The Run Specification also details the claim and release strategies for workers:

Claim Strategies

- **Start-up Only.** The YellowDog Scheduler claims Workers at the start of the Work Requirement but otherwise will not take further actions if Workers are no longer available.
- **Maintain Ideal.** The YellowDog Scheduler checks Workers at regular intervals and tries to claim more Workers or re-provision compute to maintain the ideal level.

Release Strategies

- **No Waiting Tasks.** This strategy will release unused Workers from a Task Group when it no longer has any waiting tasks.
- **All Tasks Completed.** This strategy will release all Workers when all the Tasks within a Task Group are finished.
- **Work Requirement Finished.** This strategy will only release Workers when the Work Requirement is finished.

Other properties within the Run Specification are:

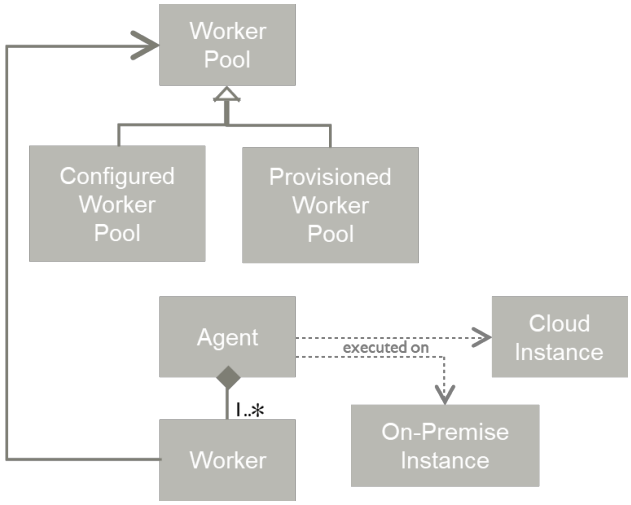
- **Share Workers.** Allows workers claimed by the Task Group to be shared with other Task Groups under the same subscription.
- **Maximum Task Retries.** Defines how many times a Task should be retried if it is failed by a Worker.

Priorities

Processing priorities can be set on Work Requirements, Task Groups and Tasks. Priorities can be positive or negative, the default priority is zero, allowing a priority setting to prioritize or de-prioritize relative to the default.

Worker Pools

Worker Pools are collections of Workers running on either on-premise or cloud compute that will be responsible for executing tasks. Compute is either "configured" – from invested / on-premise sources and/or "provisioned" – from cloud resources.



Claim and Release

The Worker Pool becomes "active" when at least one Worker has been claimed. When all Workers are released, the system tells all the Workers to shut down before terminating the provisioned instances (where the source compute is provisioned rather than configured).

YellowDog Agent

The YellowDog Agent runs on both on-premise and cloud instances. It is responsible for launching Workers on these instances. The YellowDog Agent (Agent jar) starts when the host instance is started. The Agent determines how many Workers to start up and how to communicate with the Scheduler via a local configuration file. If the Agent is installed for an on-premise Configured Worker Pool, the local configuration file is defined as part of setting up the installation. When the Agent is initiated for a cloud-based Provisioned Worker Pool that configuration is automatically provided by the Scheduler.

Agent stack

The Agent stack on the Worker Node has 3 layers:

- Executive. Application level program(s) that perform the task allocated to the Work Agent.
- Agent and Workers. Communicates with Scheduler service to get tasks to perform. Also publishes task level stats to the YellowDog Platform.
- Instance Monitor (optional). Publishes low level system stats to the YellowDog Platform.

Workers

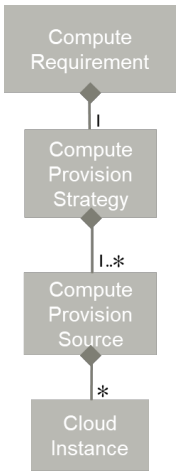
Workers are allocated Tasks of a specific Task Type which must be supported by the Worker. When Workers are started, they register with the YellowDog Scheduler using a Worker Token provided during configuration. The Scheduler will return an ID which the Worker will use to communicate with the Scheduler. The Worker will subsequently ask for instructions from the Scheduler. The instructions can be for the Worker to perform a task, sleep for a period of time or shut down. When a worker has completed a task, or awoken from a sleep period, it requests further instructions from the Scheduler.

The Worker performs a Task by executing a program installed on the instance. The program that is executed is defined in the Task Specification which is defined as part of the Agent configuration. The task specific configuration (initData.txt) is saved to the Worker's working folder so that it can be accessed by the program when it is started. The initData.txt data contains the string value specified in the Task's initData property as defined in the Work Requirement. This value can be as simple as a set of program arguments or it can be structured text data e.g. JSON or XML.

The YellowDog Scheduler actively monitors Workers by means of a heartbeat mechanism. If the Worker does not respond with in a specified timeframe, the Scheduler will automatically retry tasks if the Run Specification has a positive maximum retries value.

Compute Requirement

The Compute Requirement defines the strategy to use when provisioning compute together with the sources of compute to use. When submitted to the Compute Service extra data is added to the Compute Requirement describing its current state and the state of any provisioned computer machine instances. The latest state of the Compute Requirement and its associated instances can be retrieved at any point from the Compute Service.



Compute Provision Strategy

The Compute Provision Strategy contains the provisioning source and strategy. The provisioning strategies available are:-

- Single Source Provision Strategy. Instructs YellowDog Compute to use a single compute provision source for the compute requirement.
- Waterfall Provision Strategy. Instructs YellowDog Compute to provision computer machine instances by acquiring the maximum available from each compute provision source in order, until the required number of instances is reached.
- Split Provision Strategy: Instructs YellowDog Compute to split the provision of computer machine instances as evenly as possible across the compute provision sources.