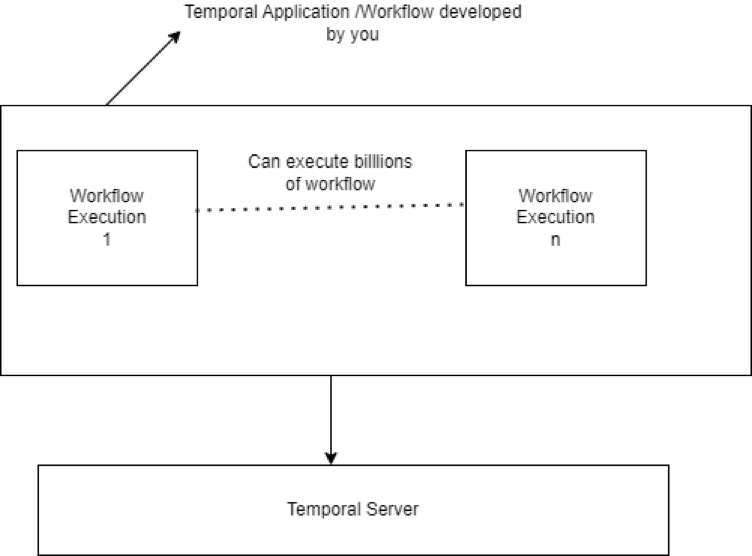
**TEMPORAL DOCUMENTATION**

**What is temporal ?**

Temporal is an open-source distributed system for managing stateful and long-running workflows in a scalable and fault-tolerant manner. It provides a programming model that enables developers to build applications that coordinate multiple services, manage complex business processes, and automate recurring tasks.



**Features of Temporal**

Temporal provides a high-level API for defining workflows and activities, as well as a runtime engine that manages the execution of workflows and tracks the state of the workflow instances. It also provides a number of features that make it well-suited for building mission-critical applications

**Fault-tolerance:** Temporal can recover from failures in a fault-tolerant manner, ensuring that workflows are executed reliably and without interruption.

**Scalability**: Temporal can scale to handle high volumes of workflow instances, activities, and events.

**Durability**: Temporal stores the state of workflow instances in a durable storage system, such as a database or object store, to ensure that the state is preserved even in the face of failures.

**Visibility**: Temporal provides tools for monitoring the status and progress of workflow instances, as well as for debugging and troubleshooting.

**Some of the Common Terminologies used in Temporal**

1. Workflow: A workflow is a sequence of steps or activities that accomplish a specific goal. In Temporal, workflows are defined using code and executed by the Temporal engine. Workflows can be long-running and can interact with external services and resources.
2. Activity: An activity is a unit of work that is performed as part of a workflow. Activities can be short-lived or long-running and can perform tasks such as calling external services, performing calculations, or sending notifications.
3. Workflow instance: A workflow instance is an individual execution of a workflow. Each workflow instance has a unique ID and a state that is managed by the Temporal engine.
4. Task queue: A task queue is a logical grouping of tasks that are processed by a set of workers. Task queues are used to distribute tasks to workers and to ensure that tasks are executed in the correct order.
5. Worker: A worker is a process or thread that executes tasks on behalf of a workflow. Workers listen to task queues and pick up tasks as they become available.
6. Workflow type: A workflow type is a definition of a specific workflow, including its name, version, and the code that implements it.
7. Workflow task: A workflow task is a unit of work that is performed by the Temporal engine as part of the execution of a workflow. Workflow tasks include activities, timers, and child workflows.
8. Timeouts: Timeouts are used to control the duration of various operations, such as executing activities or waiting for a signal.

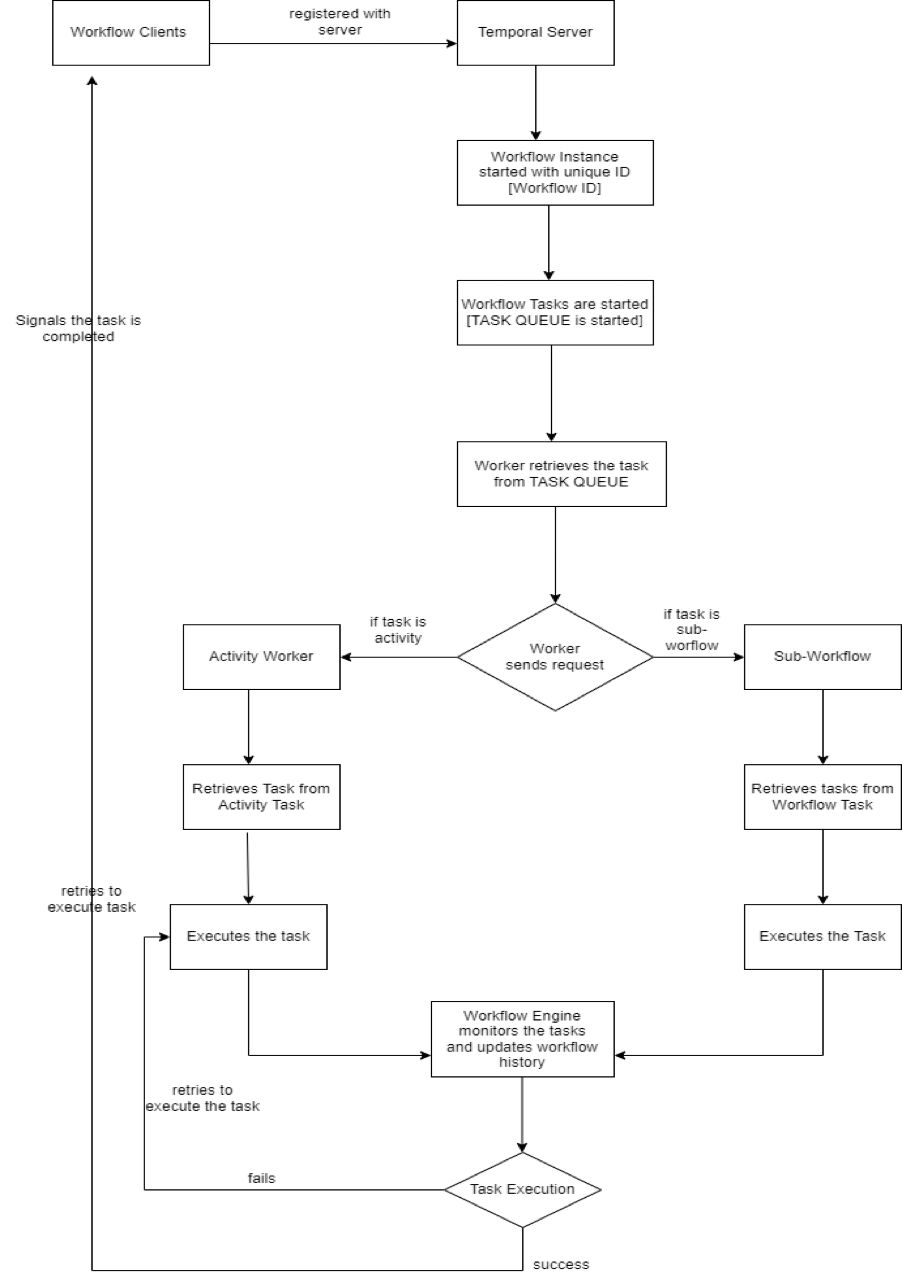
Temporal supports several types of timeouts, including:

* **Workflow timeout:** A workflow timeout is a maximum duration that a workflow can run. If a workflow does not complete within the specified timeout, it is automatically terminated. Workflow timeouts can be set when a workflow is started or can be updated during the execution of the workflow.
* **Activity timeout:** An activity timeout is a maximum duration that an activity can run. If an activity does not complete within the specified timeout, it is automatically canceled. Activity timeouts can be set when an activity is scheduled or can be updated during the execution of the workflow.
* **Schedule-to-close timeou**t: A schedule-to-close timeout is a maximum duration that a workflow can run from the time it is started until it completes. This timeout includes the time spent in waiting for child workflows, timers, and signals.
* **Schedule-to-start timeout**: A schedule-to-start timeout is a maximum duration that a workflow can wait before it is started. If a workflow is not started within the specified timeout, it is automatically canceled.
* **Schedule-to-fire timeout**: A schedule-to-fire timeout is a maximum duration that a timer can wait before it fires. If a timer does not fire within the specified timeout, it is automatically canceled.

1. Execution Time:Execution time refers to the total amount of time that is required to execute a workflow or an activity.
2. Cron Schedules: Cron schedules can be used to trigger workflow executions at specific intervals or times, using a syntax similar to the Unix cron system. Cron schedules are specified as strings that define the frequency and timing of the workflow executions.

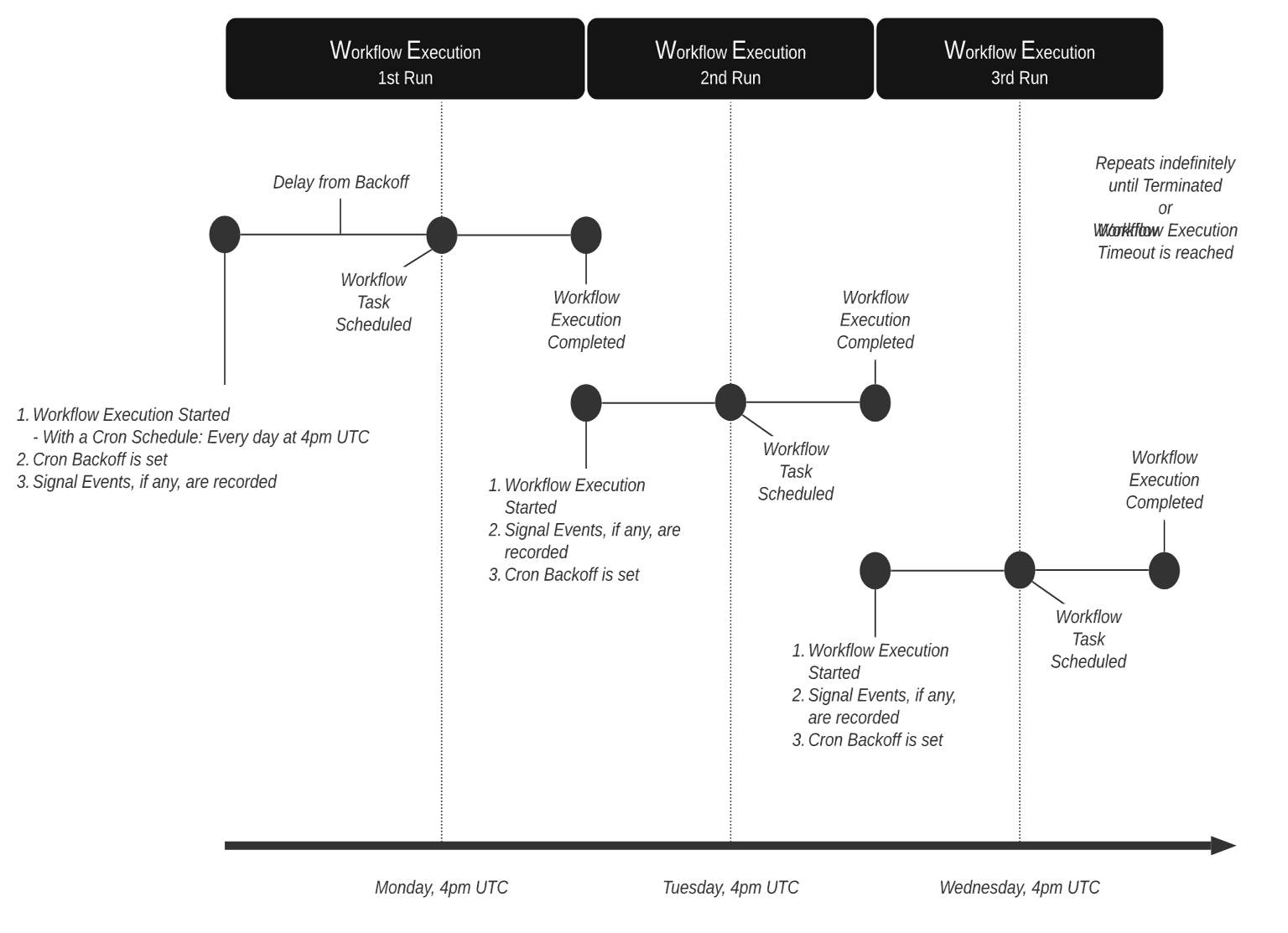
**Here is a high-level workflow execution flow diagram in Temporal:**

1. A workflow is defined in code and registered with the Temporal server.
2. A workflow instance is started by a client application, which sends a request to the Temporal server to start the workflow.
3. The Temporal server creates a new workflow instance and assigns it a unique ID.
4. The workflow engine loads the workflow definition and begins executing the workflow tasks.
5. The workflow engine schedules the first task in the workflow based on the task dependencies and any other scheduling criteria.
6. A workflow worker retrieves the task from the workflow task queue and begins executing the task.
7. If the task is an activity, the workflow worker sends a request to an activity worker to execute the activity. The activity worker retrieves the activity task from the activity task queue and executes the activity code.
8. If the task is a sub-workflow, the workflow worker sends a request to the Temporal server to start the sub-workflow.



1. The workflow engine monitors the progress of the tasks and updates the workflow history accordingly.
2. If a task fails, the workflow engine retries the task based on the retry policy defined in the workflow definition.
3. Once all tasks are completed, the workflow engine marks the workflow instance as complete and returns the output parameters to the client application.
4. The workflow history is persisted and can be used for debugging, auditing, and other purposes.

**Temporal Cron Job Execution**

****

**Cron Job Execution**

* A Temporal Cron Job is similar to a classic unix cron job.
* Just as a unix cron job accepts a command and a schedule on which to execute that command, a Cron Schedule can be provided with the call to spawn a Workflow Execution.
* If a Cron Schedule is provided, the Temporal Server will spawn an execution for the associated Workflow Type per the schedule.
* Each Workflow Execution within the series is considered a Run.
* Each Run receives the same input parameters as the initial Run.
* Each Run inherits the same Workflow Options as the initial Run.