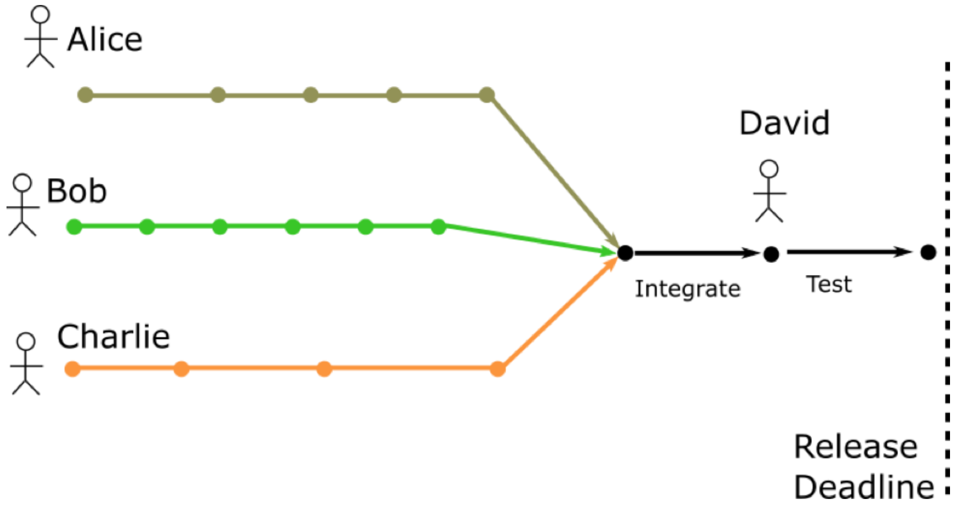
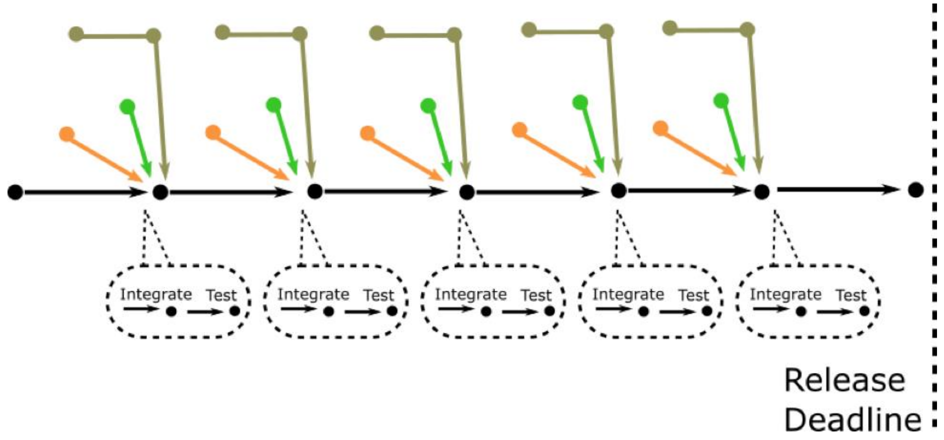
CI with GitHub Actions

Continuous Integration

The problem: Longer independent development times result in more incompatible code to integrate.



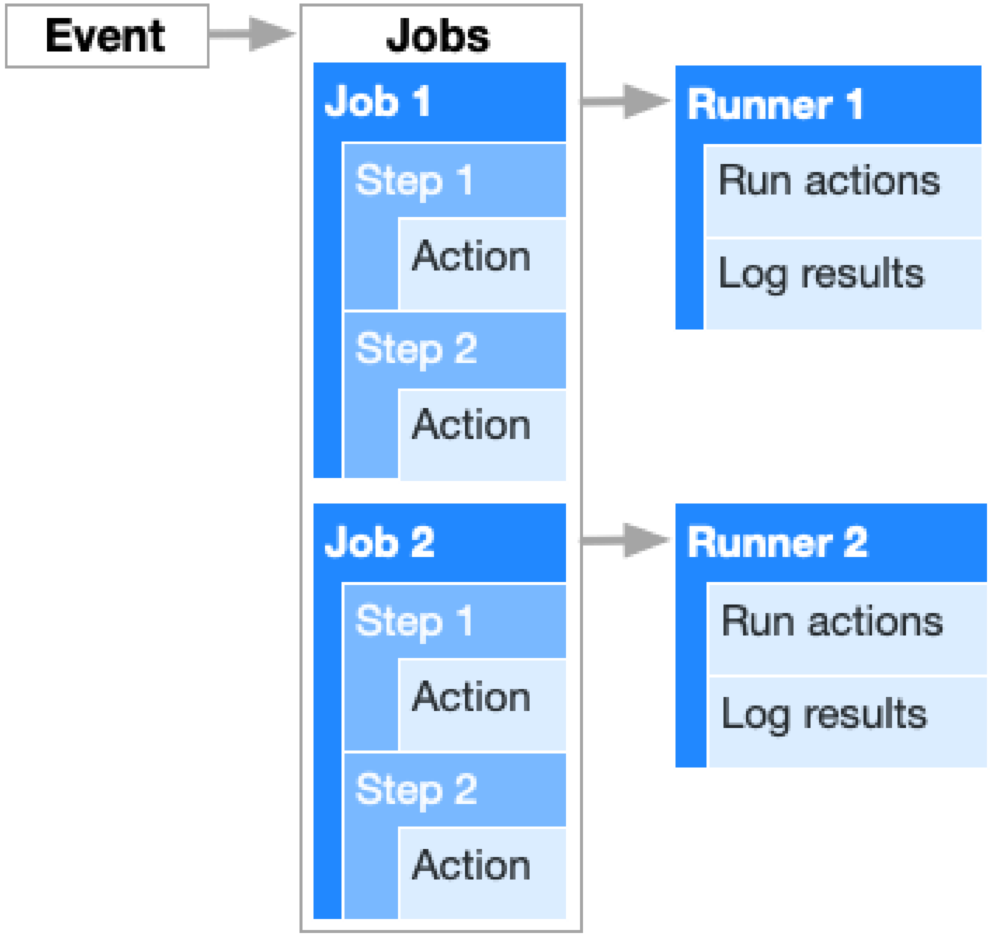
The solution: Frequent, regular integrations that are smaller in scope. This is called “continuous integration.”



Smaller scale integrations are easier to automate and there are a number of services that can be used for this purpose.

GitHub Actions

“Events” on GitHub automatically trigger predefined scripts to build, test, package, or release a project.



Event

Jobs

Job 1

Step 1

Action

Step 2

Action

Job 2

Step 1

Action

Step 2

Action

Runner 1

Run actions

Log results

Runner 2

Run actions

Log results

GitHub Actions jargon

- **Workflows**: An automated procedure that is added to the repository. A workflow is made up of one or more jobs and are triggered by an event.

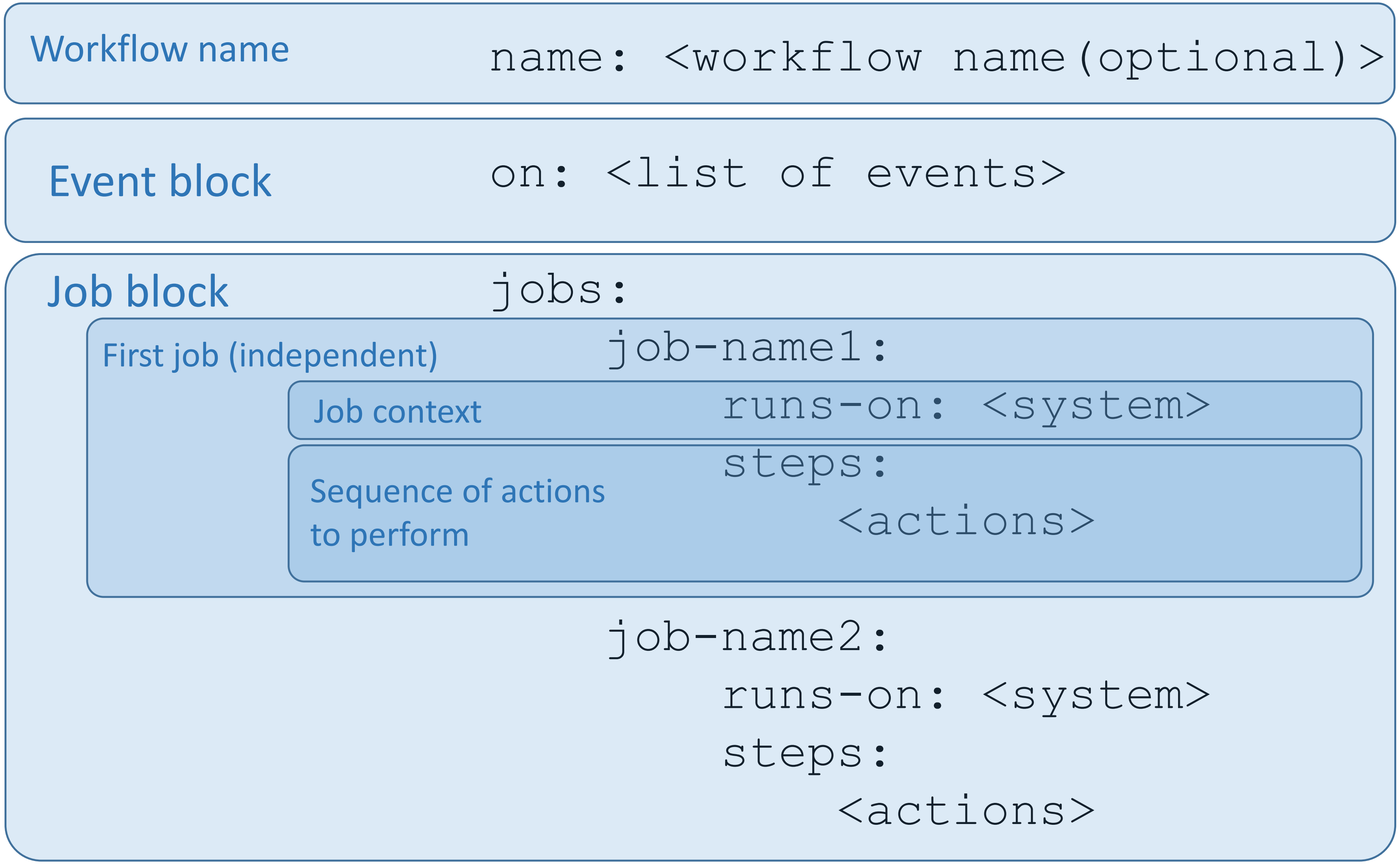
- **Events**: A specific activity that triggers a workflow. These events (mostly) originate from GitHub, such as a `push` or `pull-request`.

- **`Jobs`**: A set of `steps` that is executed within the workflow. Jobs are run independent of one another. Jobs can be run in parallel (default) or in series. Each job has its own compute environment.

- **`Steps`**: The set of tasks that run within a job. A single step can be a single *Action* or a shell command. The `steps` in a single job share a common compute environment.

- **Actions**: *Actions* are standalone commands that are combined within `steps` to create a job. Actions are the smallest building block of a workflow. You can create your own actions or use those created by the GitHub community.

Anatomy of an Actions script



Workflow name

Event block

Job block

First job (independent)

Job context

Sequence of actions to perform

```
name: <workflow name(optional)>

on: <list of events>

jobs:
    job-name1:
        runs-on: <system>
        steps:
            <actions>

    job-name2:
        runs-on: <system>
        steps:
            <actions>
```

Component of jobs

`runs-on:` Required keyword that specifies the compute environment in which to run the job. Options:

`ubuntu-latest, Ubuntu-18.04, windows-latest, windows-2016, macos-latest, macos-11, self-hosted`

`name:` optional keyword with a string value for display on GitHub

`needs:` optional keyword to connect multiple jobs

`if:` logical keyword usually used with an expression `${{ expression }}` for control flow of job

`steps:` keyword whose value is the list of actions

`strategy:` Context in which to define a matrix of different configurations in which to run the job

Component of steps

`name:` Name of the action. Optional keyword with a string value for display on GitHub

`uses:` Keyword used to run community Actions or separate user defined workflow

`run:` Keyword used to run a single action (command or script)

`shell:` Keyword to explicitly define what shell environment in which to run the action. Options:
`bash, pwsh, cmd (windows), python, sh (Linux/macOS)` (depends on value of `runs-on`)

`with:` A map of the input parameters passed to the single action

`env:` key/value pairs specifying environment variables to be used in the steps block

community Actions

`actions/checkout@v2`

This is an action that checks out your repository and downloads it to the runner, allowing you to run actions against your code (such as testing tools).

`actions/setup-python@main`

This action configures the server (runner) with python so that you can run python scripts

`actions/upload-artifact@v2`

Allows you to generate a file (artifact) from the job, to be made available to other jobs or stored on GitHub

`actions/download-artifact@v2`

Allows you to download an artifact (a file produced by another job or workflow run) to be used in this job.
If being done within the same workflow run, you must use “`needs: <upload job name>`”