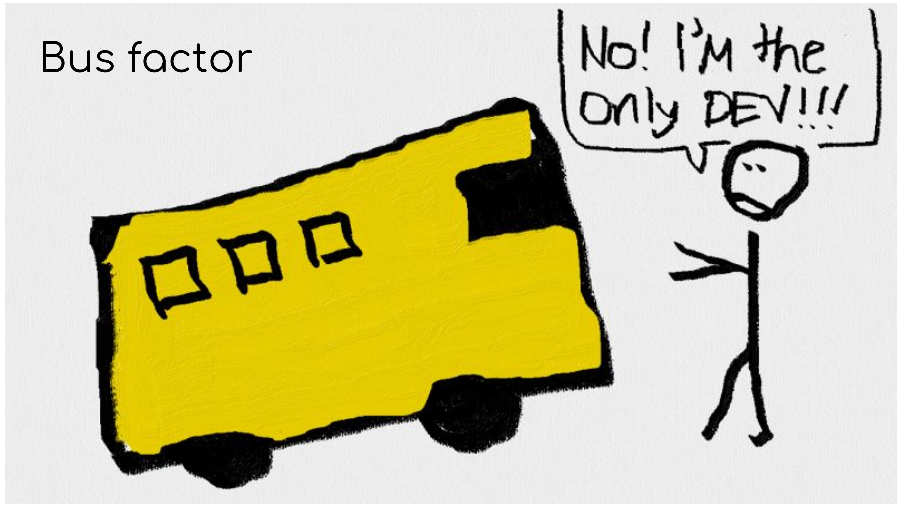
ADALTAS

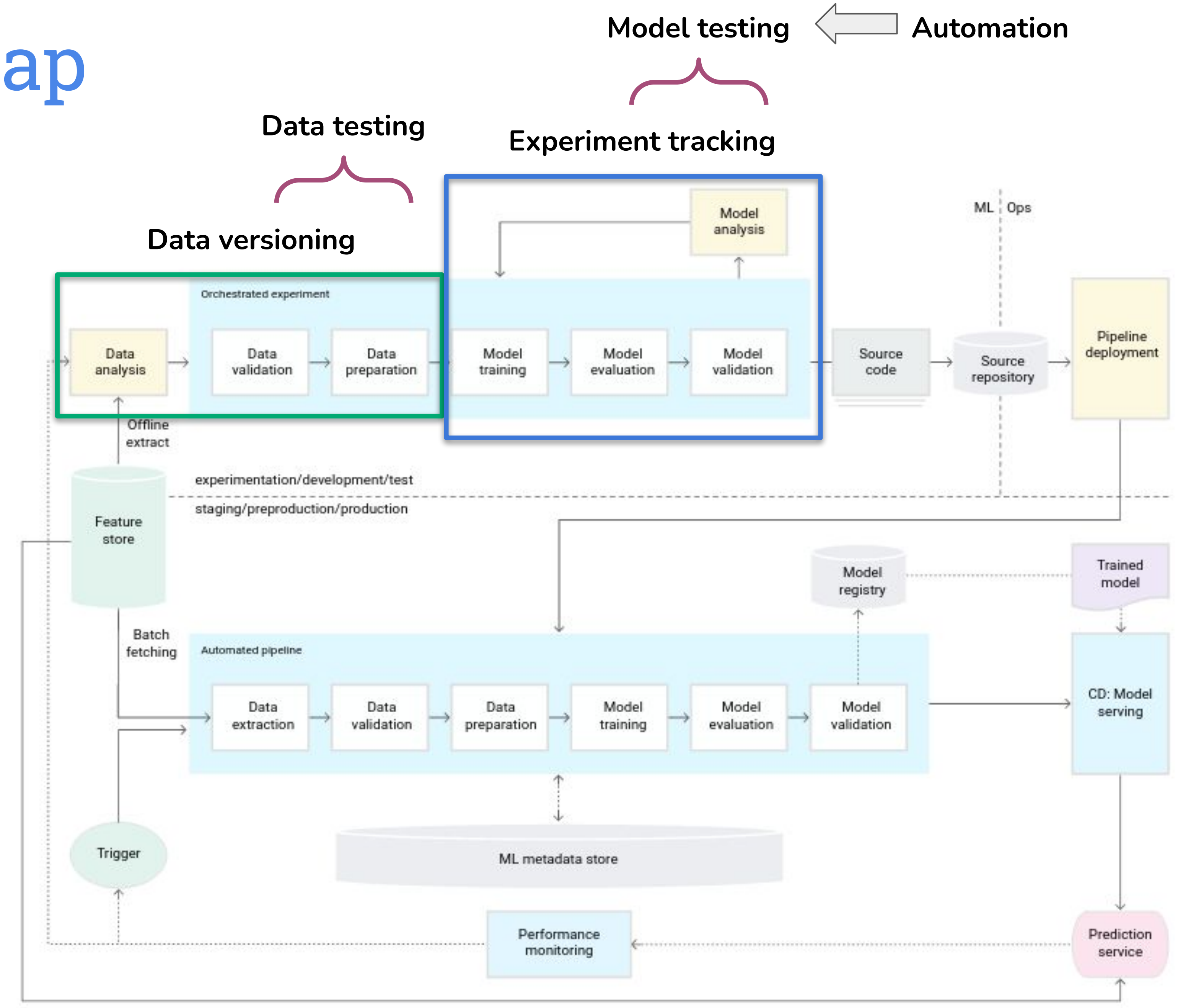
5. Automation

Continuous testing / integration / delivery (deployment)

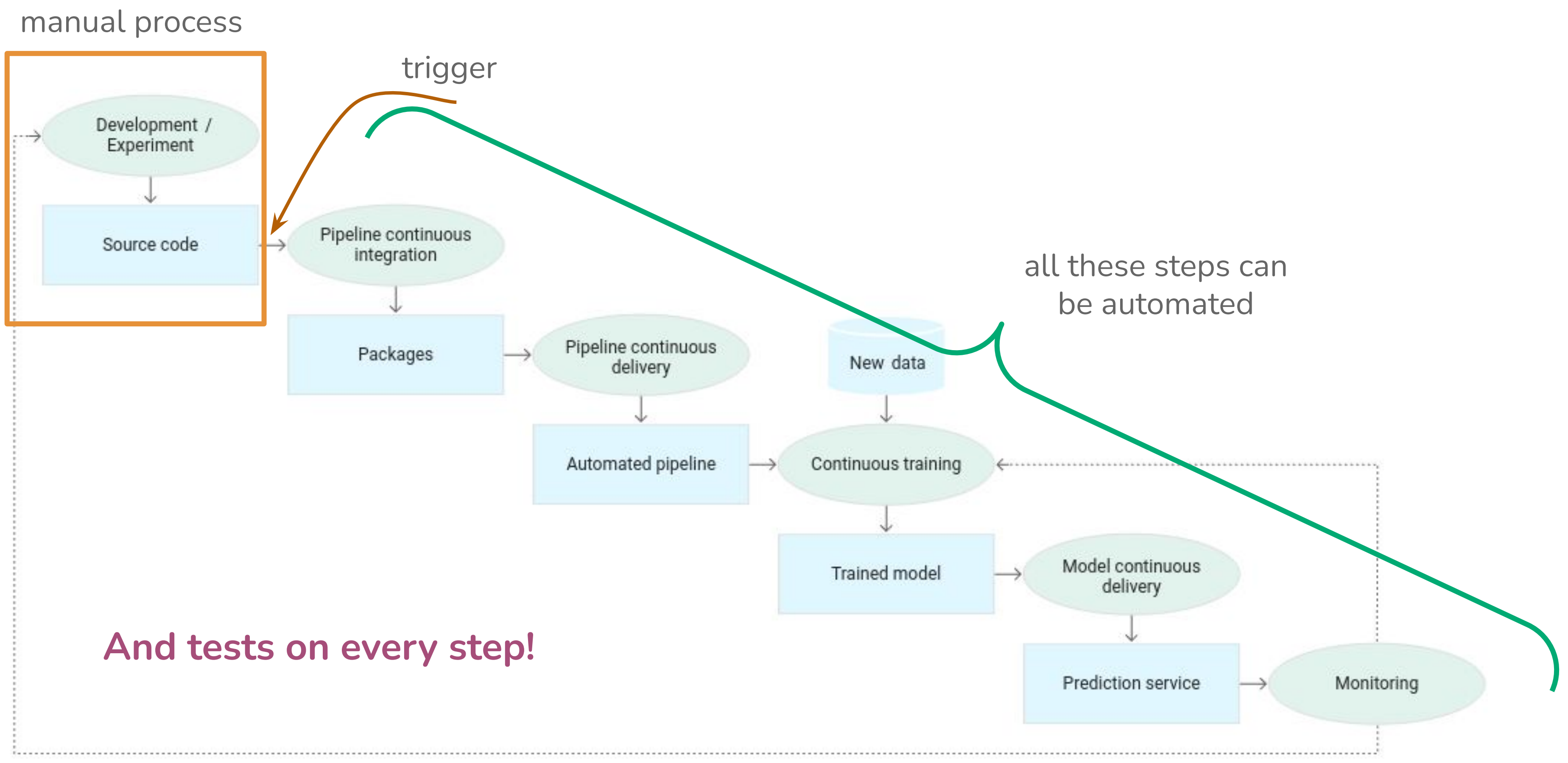
Benefits of automation



Roadmap


Model testing
Automation
Data testing
Experiment tracking
Data versioning
Orchestrated experiment
Data analysis
Data validation
Data preparation
Model training
Model evaluation
Model validation
Model analysis
Source code
Source repository
Pipeline deployment
ML
Ops
Offline extract
experimentation/development/test
staging/preproduction/production
Feature store
Model registry
Trained model
Batch fetching
Automated pipeline
Data extraction
Data validation
Data preparation
Model training
Model evaluation
Model validation
CD: Model serving
Trigger
ML metadata store
Performance monitoring
Prediction service

Fully automated pipeline



And tests on every step!

DevOps Principles (Software development)

Continuous Integration. Members of the development team integrate their code in a shared repository, several times a day. Each developer segments the work into small, manageable chunks of code and detects potential merge conflicts and bugs quicker.

Continuous Delivery. As the code is continuously integrated, it is also consistently delivered to the end-user. Smaller contributions allow faster update releases, which is a crucial factor for customer satisfaction.

Continuous Deployment. A big part of DevOps is automating processes to speed up production. Continuous deployment involves automating releases of minor updates that do not pose a substantial threat to the existing architecture.

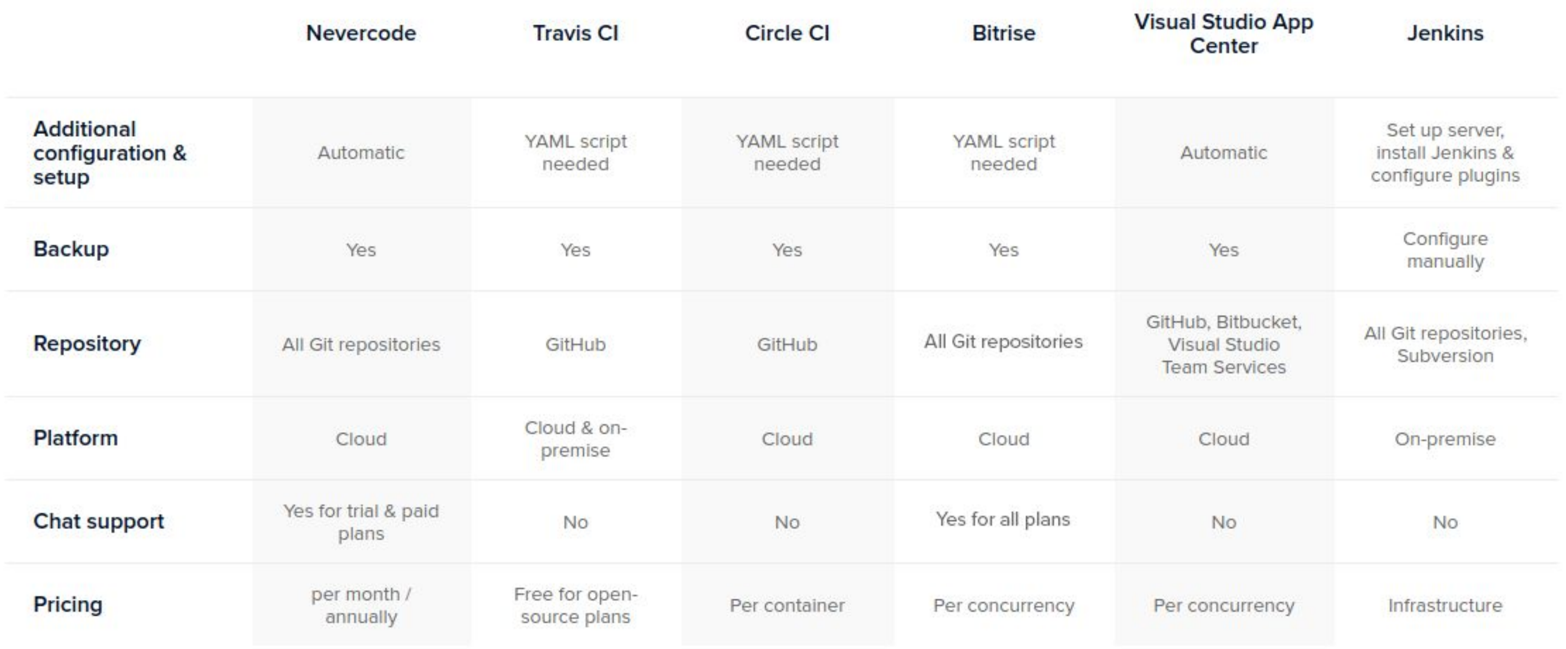
Continuous Testing. Such a strategy involves testing as much as possible in every step of development. Automated tests give valuable feedback and a risk assessment of the process at hand.

Continuous Operations. The DevOps team is always working on upgrading software with small but frequent releases. That is why DevOps requires constant monitoring of performance. Its main goal is to prevent downtime and availability issues during code release.

=> Small but frequent improvements (agility)
=> Constant testing
=> Automation of process to speed up and decrease human interactions

Tools

- Many traditional tools, known in DevOps:
  - Jenkins
  - Travis CI
  - CircleCI



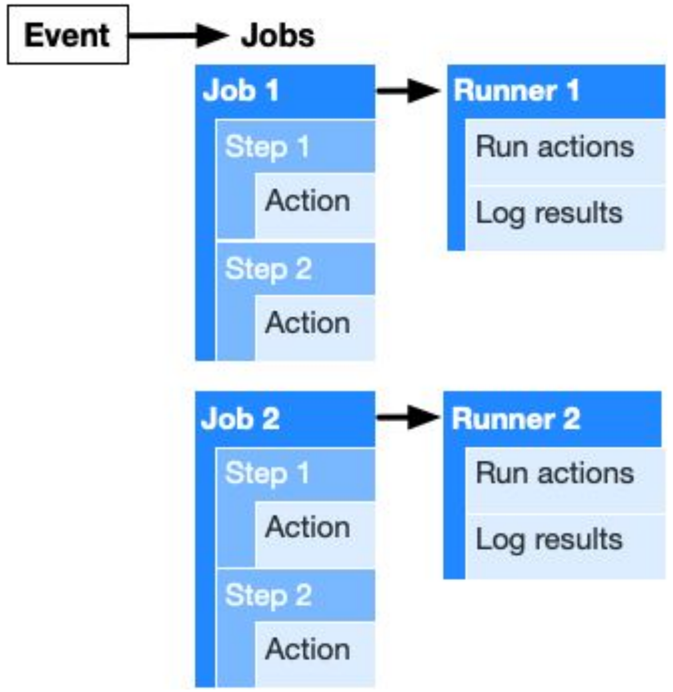
	Nevercode	Travis CI	Circle CI	Bitrise	Visual Studio App Center	Jenkins
Additional configuration & setup	Automatic	YAML script needed	YAML script needed	YAML script needed	Automatic	Set up server, install Jenkins & configure plugins
Backup	Yes	Yes	Yes	Yes	Yes	Configure manually
Repository	All Git repositories	GitHub	GitHub	All Git repositories	GitHub, Bitbucket, Visual Studio Team Services	All Git repositories, Subversion
Platform	Cloud	Cloud & on-premise	Cloud	Cloud	Cloud	On-premise
Chat support	Yes for trial & paid plans	No	No	Yes for all plans	No	No
Pricing	per month / annually	Free for open-source plans	Per container	Per concurrency	Per concurrency	Infrastructure

- Newer:
  - GitHub Actions
  - GitLab CI

Anatomy of GitHub Actions

ADALTAS

- Allow us to automate a process
- Are event-driven - a specific event will trigger a workflow (one or many steps/scripts)



Event - specific activities that trigger a workflow run. For example, a workflow is triggered when somebody pushes to the repository or when a pull request is created.

Runner - a machine with the Github Actions runner application installed. Then runner waits for available jobs it can then execute. After picking up a job they run the job's actions and report the progress and results back to Github. Runners can be hosted on Github or self-hosted on your own machines/servers.

Workflow - an automated process that is made up of one or multiple jobs and can be triggered by an event. Workflows are defined using a YAML file in the .github/workflows directory.

Job - multiple steps and runs in an instance of the virtual environment. Jobs can run independently of each other or sequential if the current job depends on the previous job to be successful.

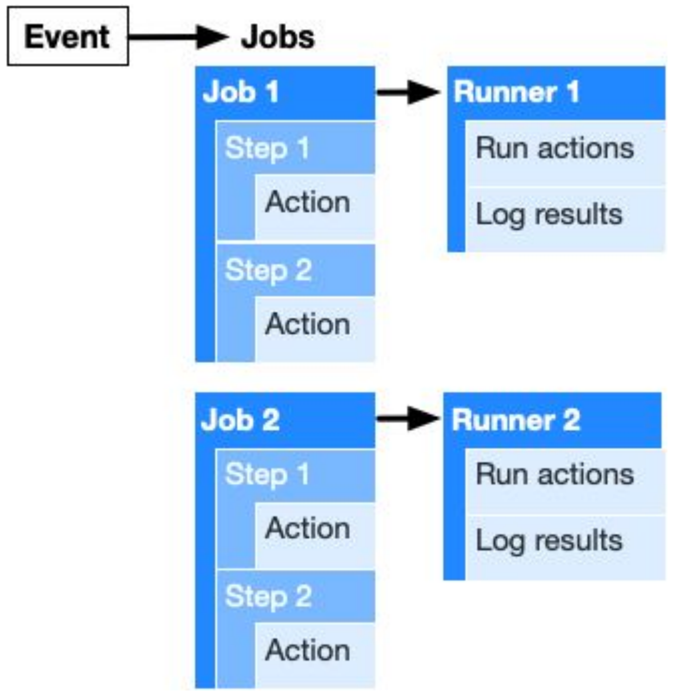
Step - a set of tasks that can be executed by a job. Steps can run commands or actions.

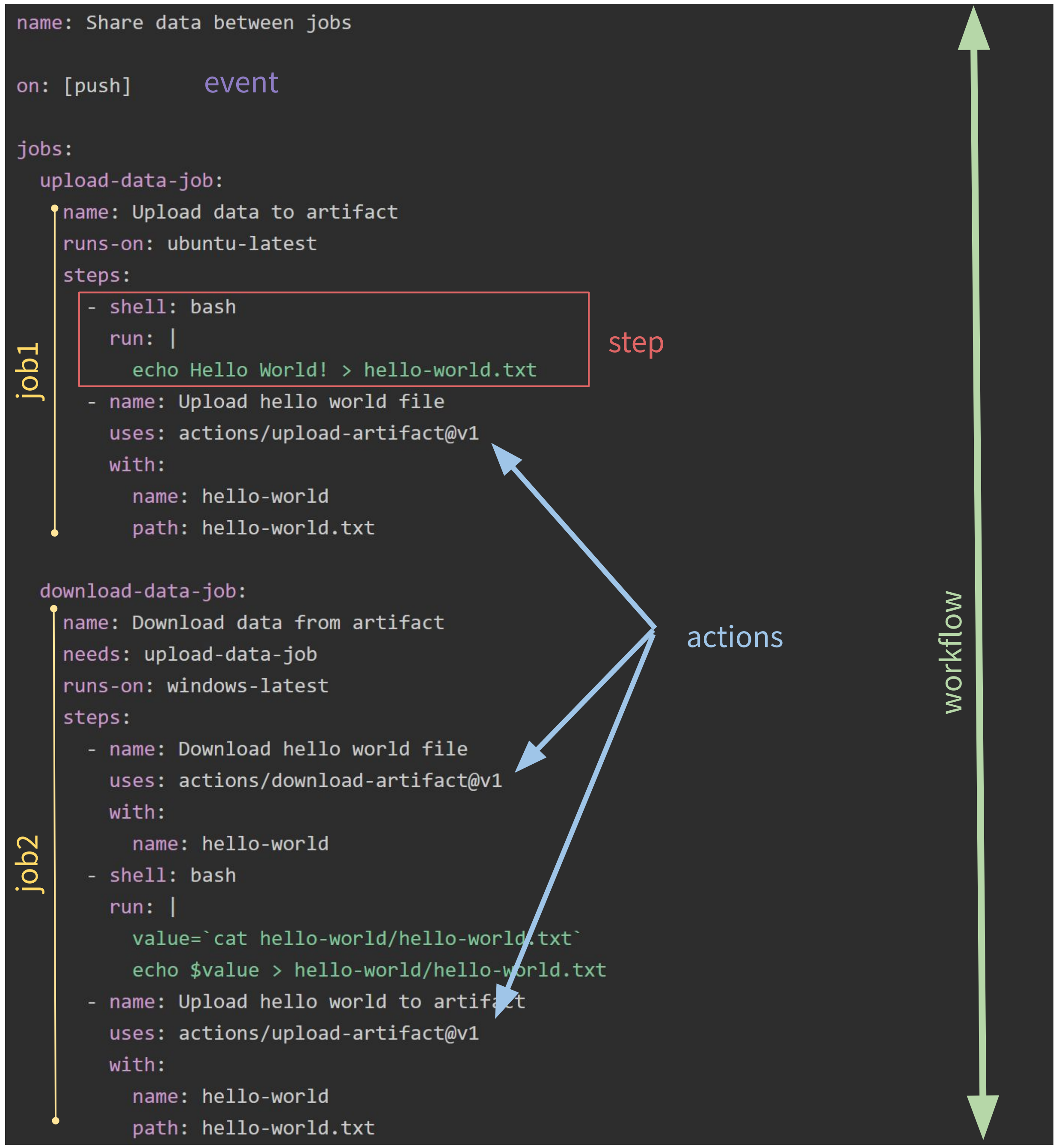
Actions - the smallest portable building block of a workflow and can be combined as steps to create a job. You can create your own Actions or use publicly shared Actions from the Marketplace.

https://docs.github.com/en/actions/learn-github-actions/introduction-to-github-actions
https://gabrieltanner.org/blog/an-introduction-to-github-actions

GitHub Actions

ADALTAS





```yaml
name: Share data between jobs

on: [push]

jobs:
  upload-data-job:
    name: Upload data to artifact
    runs-on: ubuntu-latest
    steps:
      - shell: bash
        run: |
          echo Hello World! > hello-world.txt
      - name: Upload hello world file
        uses: actions/upload-artifact@v1
        with:
          name: hello-world
          path: hello-world.txt

  download-data-job:
    name: Download data from artifact
    needs: upload-data-job
    runs-on: windows-latest
    steps:
      - name: Download hello world file
        uses: actions/download-artifact@v1
        with:
          name: hello-world
      - shell: bash
        run: |
          value=`cat hello-world/hello-world.txt`
          echo $value > hello-world/hello-world.txt
      - name: Upload hello world to artifact
        uses: actions/upload-artifact@v1
        with:
          name: hello-world
          path: hello-world.txt
```

https://docs.github.com/en/actions/reference/workflow-syntax-for-github-actions

What will we do?

ADALTAS

- Objective:
    - Automate model evaluation with GitHub Actions
- Result:
    - Every time that we will push changes to our repository, the tests will run