Using CLI Tools at Your Job

Richard Horridge

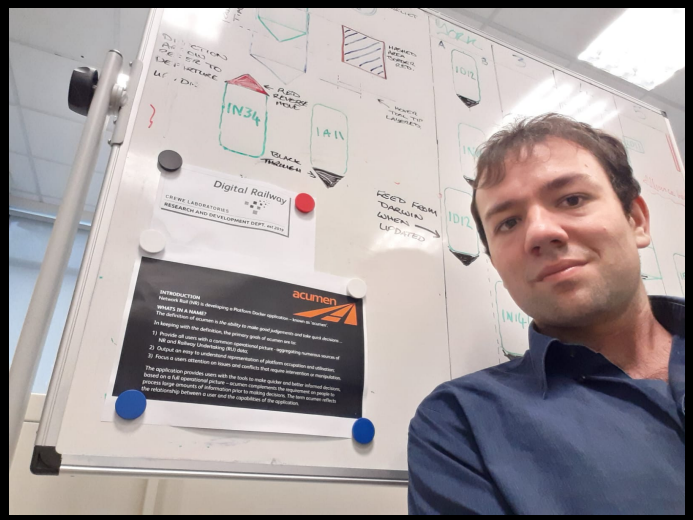
UNIVERSITY OF BIRMINGHAM

Introduction

About Me

Richard Horridge

- PhD Researcher in Next Generation Railway Track Condition Monitoring at University of Birmingham
  Centre for Railway Research and Education
- Operations Interface Manager (OIM) in System Operator at Network Rail
- PhD: Scaling up instrumentation of passenger trains with inertial sensors to gain deeper understanding of the state of railway track
- OIM: Developing and maintaining Operational Decision Support Tools for railway operations staff
- Six years of experience working across industry and academia
- AWS Certified Solutions Architect Professional and Microsoft Certified Azure Administrator



acumen ODST

- acumen Operational Decision Support Tool (ODST)
- Deployed at Birmingham New Street, York, Crewe, London Euston and London King's Cross
- Used by operations staff to make tactical decisions by providing them with a full operational picture

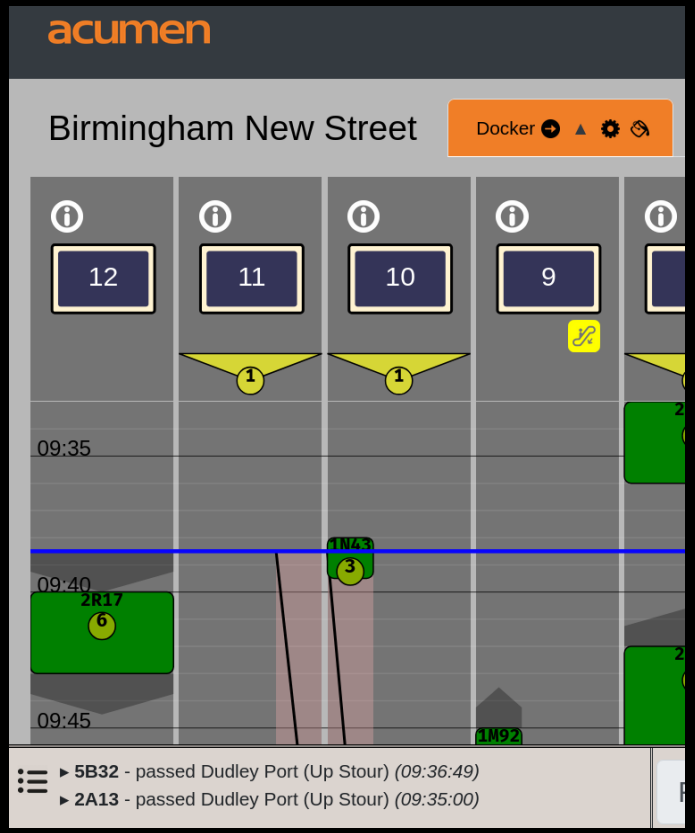


Figure: acumen Platform Docker

aSIST

- acumen Signallers Information Support Tools (aSIST)
- Deployed at over 170 signals all over the UK
- Used by signallers to notify them of incorrectly set routes

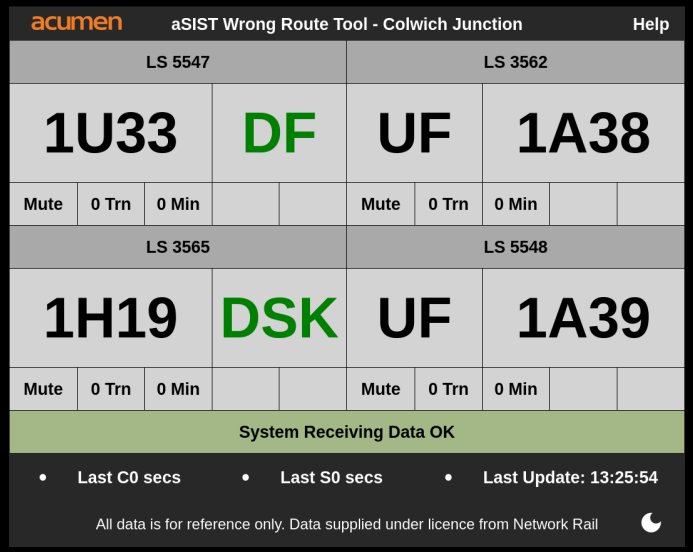


Figure: aSIST Wrong Route Tool

Condition Monitoring Research

- Collecting accelerometer and gyroscope data from train vehicle bogies
- Using it to monitor degradation in railway track condition
- Current focus on reliably positioning and aligning recording data
- Aim to scale up instrumentation of vehicles to cover entire fleet of trains

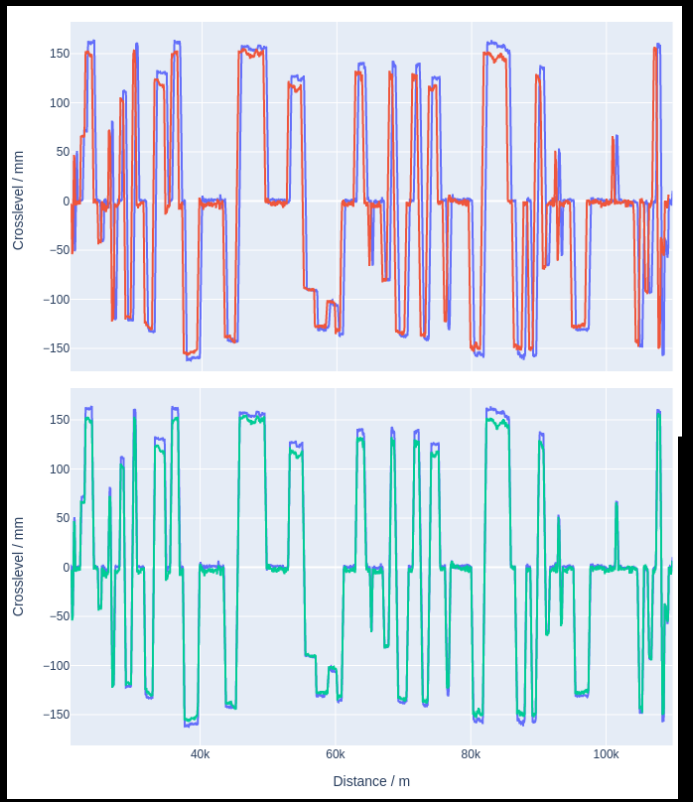


Figure: Aligning track geometry recordings

CLI Tools and Text Editors

- Choose the editor(s) that work for you
- Focus on tools not editor-specific functionality
- Standardise project configuration and tasks
- 'Narrow waist' of UNIX pipes and commands

e.g. Just (task runner), uv (Python toolchain manager), Find + Xargs (shell-centric programming)

```
# Run unit tests
test tests=unit-tests:
    poetry run pytest {{tests}} -m "{{unit-test-marks}}
 and {{slow-marks}}" -vv

# Run ALL unit tests, including all marks
test-unit-all:
    poetry run pytest {{unit-tests}} -vvv

# Run static analysis
mypy:
    poetry run mypy

# Generate coverage report for unit tests
coverage-run:
    poetry run coverage run --branch --source {{python-
name}} -m pytest -m "{{unit-test-marks}} and {{slow-mar
ks}}" {{unit-tests}}

# Report coverage in terminal
coverage-report:
    poetry run coverage report --sort=cover --skip-cove
red

# Generate HTML coverage, failing under a percentage
```

Figure: Platform-agnostic task runner (Just, Poetry)

SSH for System Administration

- Managing multiple remotes using `/etc/hosts` and `~/.ssh/config`
- Standard UNIX utilities on remote servers e.g. `vi`
- Securing your servers using firewalls, SSH config etc.
- Hosting applications in containers
- Exposing applications through reverse proxy
- Connecting to remote ports locally using SSH tunnelling
- Automating operations using provisioning tools

e.g. Fail2Ban (blocking login attempts), UFW (firewall), Docker-Compose (running applications), nginx-proxy (reverse proxy), Ansible (automation)

```
---
- name: Setup SSHD
  hosts: dev
  become: yes
  tasks:
    - name: ensure ssh root login is disabled
      lineinfile:
        dest: /etc/ssh/sshd_config
        regexp: '^#?PermitRootLogin'
        line: 'PermitRootLogin no'
    - name: disable password authentication
      lineinfile:
        dest: /etc/ssh/sshd_config
        regexp: '^#?PasswordAuthentication'
        line: 'PasswordAuthentication no'
    - name: prohibit empty passwords
      lineinfile:
        dest: /etc/ssh/sshd_config
        regexp: '^#?PermitEmptyPasswords'
        line: 'PermitEmptyPasswords no'
    - name: restart sshd
      service:
        name: sshd
        state: restarted
      when: ansible_distribution == "Ubuntu" and ans
e_distribution_version < "24"
    - name: restart sshd
```

Figure: Ansible playbook for securing SSH

Version Control in Industry

- Keeping work up-to-date using rebase
- Managing multiple development branches
- Standardising changes and review processes
- Make small, reversible changes
- Use standard commit message formats
- Run pre-commit hooks to solve common issues
- Document releases in non-technical language

e.g. pre-commit (standard hooks), Magit (powerful Git UI in Emacs), Pull Request Templates (checklist for proposed changes), Conventional Commits (spec for Git commits), Git Reflog (find lost commits), GitHub CLI (commandline interface for GitHub)

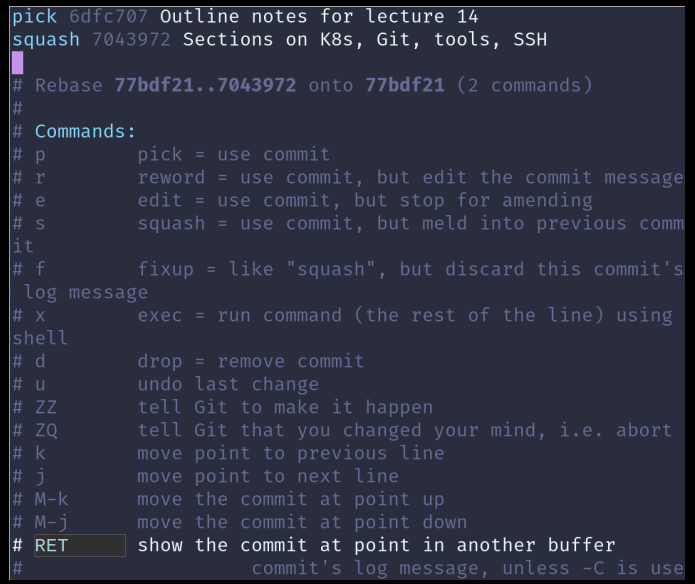


Figure: Magit rebase interface

Continuous Integration and Delivery

Continuous Integration (CI)

- Run repeatable tasks on your codebase
- Generate signed and verified build artefacts
- Ensure that code passes quality checks
- Many providers allow self-hosting to reduce cost

e.g. GitHub Actions, GitLab CI/CD, Jenkins, Travis CI, CircleCI

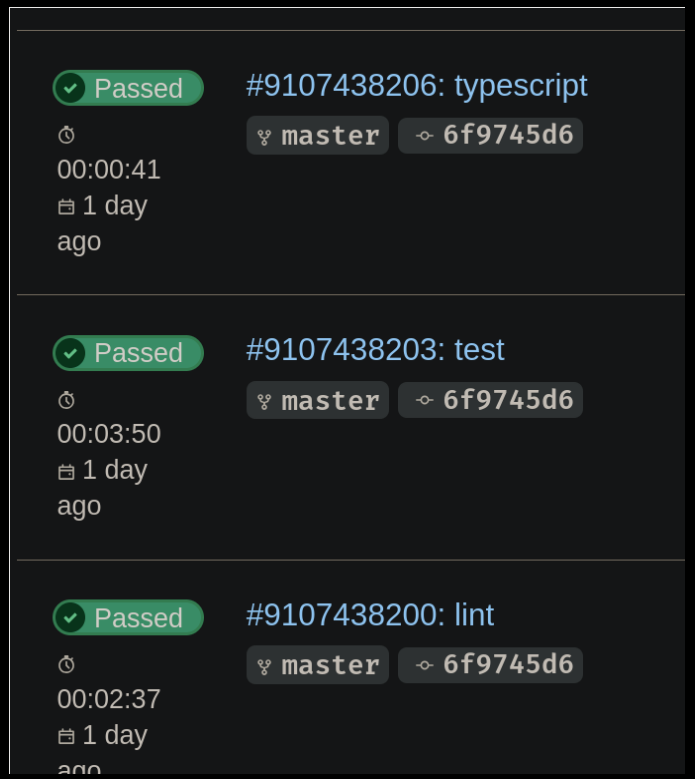


Figure: Continuous Integration pipelines in GitLab

Continuous Integration and Delivery

Continuous Delivery (CD)

- Automated deployment of artefacts to end-users
- Reduce volume of repeated manual work
- Combine with Infrastructure-as-Code for complex projects
- Focus on small incremental changes

e.g. ansible-pull (run tasks on remove servers automatically)

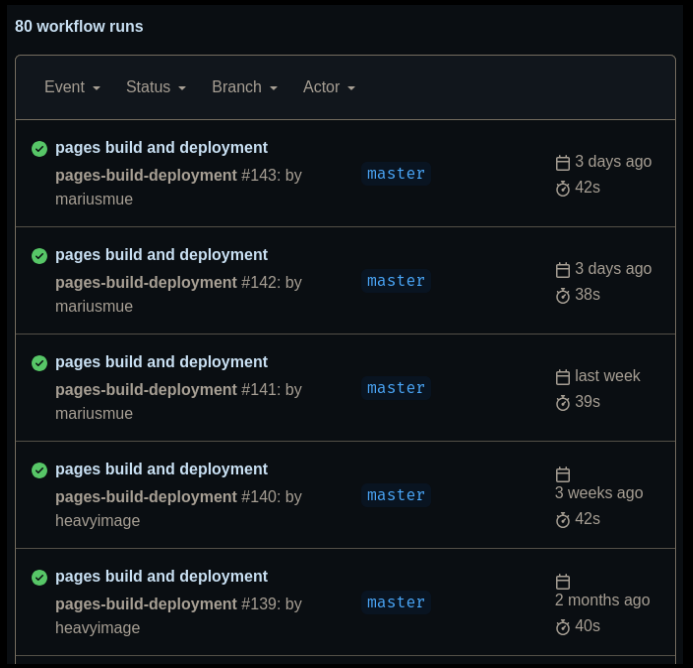


Figure: Deployment pipelines for Missing Semester website in GitHub

Documentation and Project Management

Documentation

- Ensure documentation is targeted at the reader
- Use plain text formats and version it using Git
- Publish documentation in HTML and optionally PDF formats
- Consider keeping diagrams in plain-text format alongside documentation
- Generate documentation automatically where possible in CI pipelines

e.g. Diátaxis and Divio (structuring documentation), MkDocs (generating documentation), GitHub Pages (static site hosting), PlantUML and Mermaid.JS (diagrams as code)

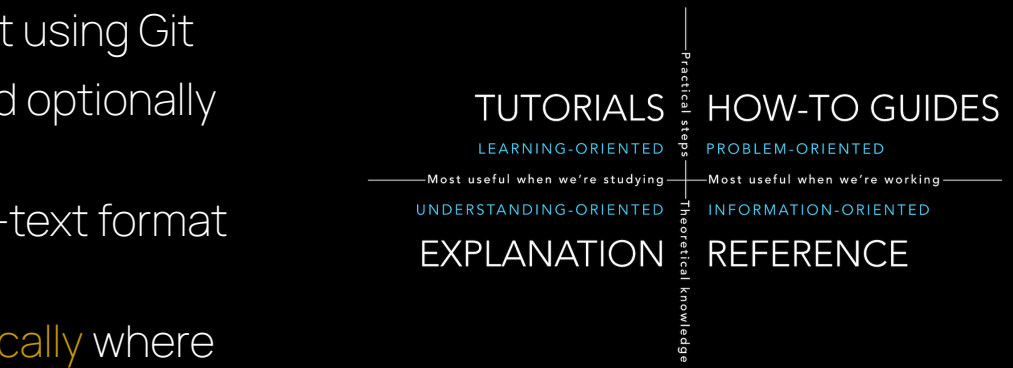


Figure: Divio documentation system

Documentation and Project Management

Project Management

- Keep track of issues and tasks
- Maintain a project plan and backlog
- Ensure your team is onboard with your processes
- Allow teams to work independently

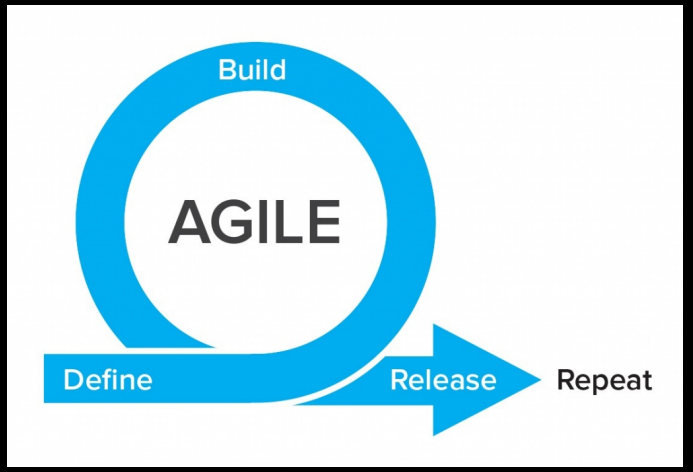


Figure: Agile lifecycle model

Case Study 1: Reporting of Open GitHub issues

Using the GitHub GraphQL API to format open issues in a repository.

```
OWNER="my-name"
REPOSITORY="my-repo"

# Format CSV header
echo '{}' | jq -r '["ID", "Issue Number", \
    "Issue URL", "Issue Title", "Labels", \
    "Project Number", "Project Title", \
    "Milestone"] | @csv'

# Query GitHub API
gh api graphql \
   -F repo="${REPOSITORY}" \
   -F owner="${OWNER}" -f \
   query="$(cat issues.gql)" \
    | jq -r '.data.repository.issues.nodes[]
        | \
        [.id, .number, .url, .title, \
        (.labels.nodes | map(.name) |
        join(",")), \
        .projectsV2.nodes[].number, \
        .projectsV2.nodes[].title, \
        .milestone.number ] | @csv'
```

```
query($owner: String!, $repo: String!) {
  repository(owner: $owner, name: $repo) {
    issues(first: 80, states: OPEN) {
      nodes{
        id
        number
        title
        labels(first: 10) {
          nodes {
            name
          }
        }
        assignees(first: 1) {
          nodes{
            name
          }
        }
        milestone {
          number
        }
        projectsV2(first: 1) {
          nodes {
            number
            title
          }
        }
      }
    }
  }
}
```

Generative AI Uses

- Current trend is increasing usage of LLMs for writing code
- You likely have to comply with workplace policy
- Depends highly on industry e.g. only allowing M365 Copilot
- Beware of data exfiltration and complexity creep!

e.g. OpenCode and Pi (agent harnesses), Claude Opus 4.6, Gemini 3.1 Pro, GPT 5.4 (proprietary large language models), OpenRouter (API for accessing LLMs)

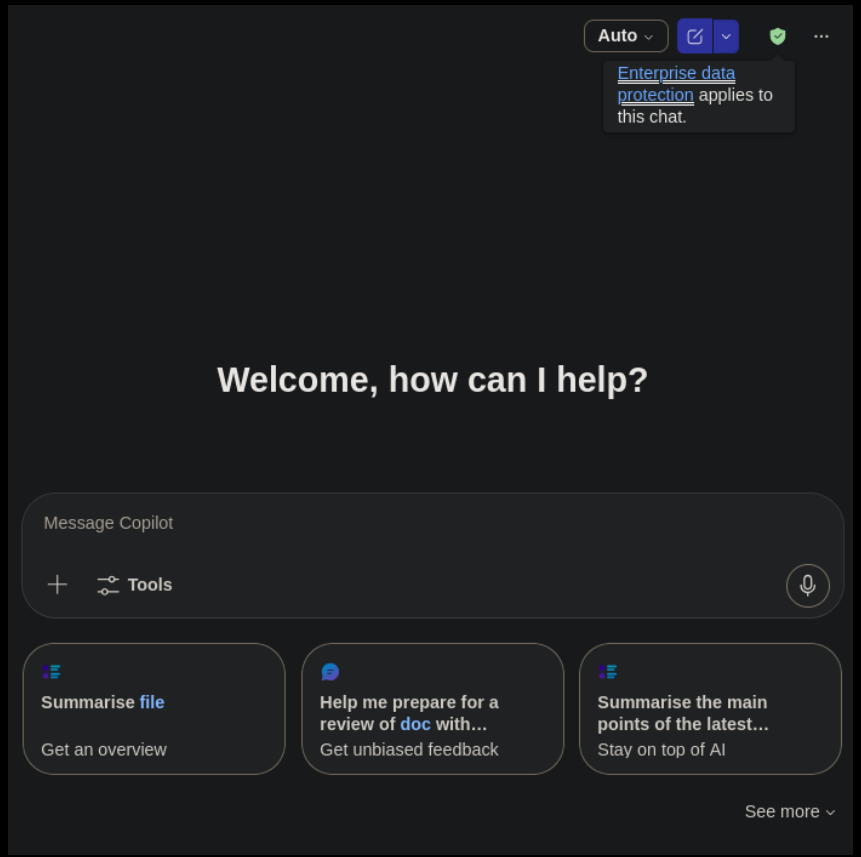
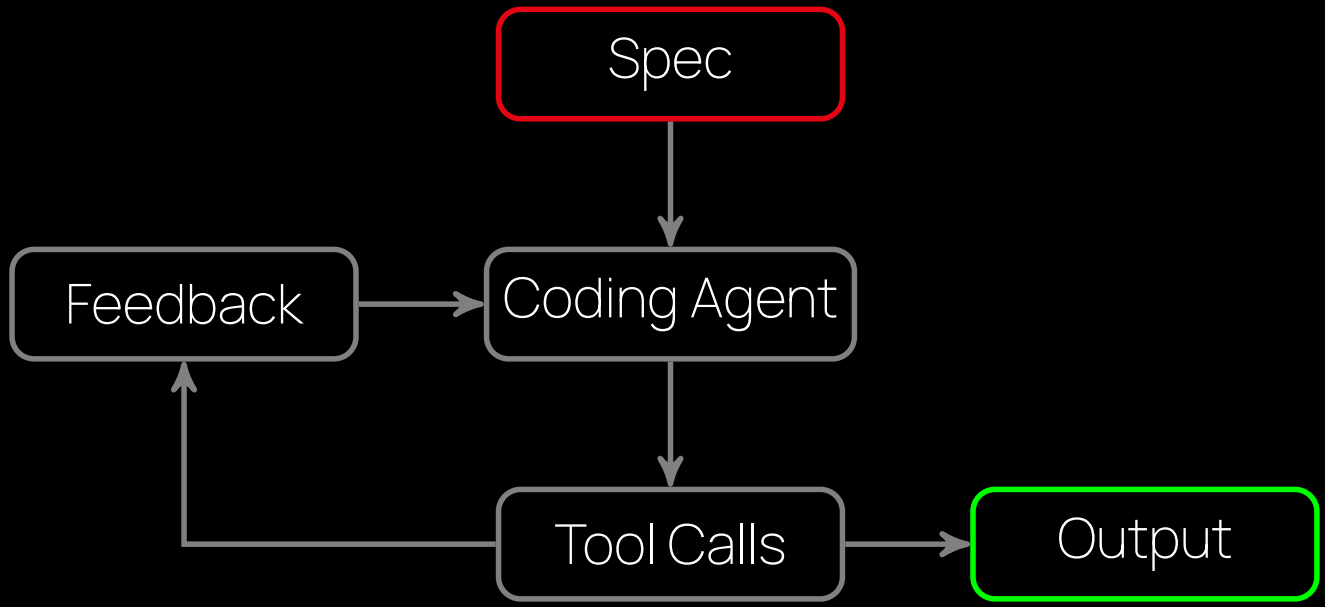


Figure: Enterprise data protection for M365 Copilot

Generative AI Uses

- Agent harnesses can be very effective due to self-reinforcing feedback
- Challenge shifts to accurately specifying work
- Be strict on taste and smell of code
- Carefully specify what is needed
- Underlines importance of accurate documentation written for humans



LaTeX for Industry / Academia

- LaTeX has uses for non-academic writing as well
- Write documents in lightweight markup language
- Exporting documents to different formats
- Stick to HTML where possible for accessibility

e.g. Org-Mode (markup format and environment), Org-Ref (integrate academic referencing), Pandoc (convert between many document formats), Beamer and Beamer Poster (generating presentations and posters using LaTeX), Quarto (technical publishing to various formats)

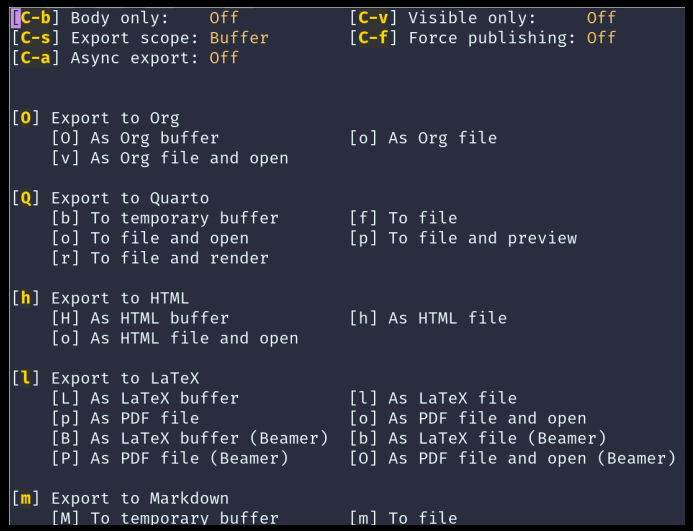


Figure: Export options from Emacs Org-Mode

Web Application Frontends

- Web applications have significant advantages for developers and users
- Many web solutions are overengineered for their purpose
- Simple and boring solutions are usually best
- When you do need the abstractions, frameworks are worth it
- Adding more dependencies generally worsens user experience

e.g. TypeScript (type checking for JavaScript), esbuild (bundling JavaScript and CSS), htmx (manage dynamic behaviour using HTML attributes), TinyJS (minimalist JavaScript framework)

Code | Blame | 35 lines (35 loc) · 2 KB

```
(() => {
  const assignDeep = (elm, props) => Object.entries(props).forEach(([key, va
    if (typeof value === 'object') return assignDeep(elm[key], value)
    try { Object.assign(elm, {[key]: value}) } catch { elm.setAttribute(key,
  })
  Array.from([
    'a', 'abbr', 'address', 'area', 'article', 'aside', 'audio', 'b', 'base'
    'bdi', 'bdo', 'blockquote', 'body', 'br', 'button', 'canvas', 'caption',
    'cite', 'code', 'col', 'colgroup', 'data', 'datalist', 'dd', 'del',
    'details', 'dfn', 'dialog', 'div', 'dl', 'dt', 'em', 'embed', 'fieldset'
    'figcaption', 'figure', 'footer', 'form', 'h1', 'h2', 'h3', 'h4', 'h5',
    'h6', 'head', 'header', 'hr', 'html', 'i', 'iframe', 'img', 'input', 'in
    'kbd', 'label', 'legend', 'li', 'link', 'main', 'map', 'mark', 'meta',
    'meter', 'nav', 'noscript', 'object', 'ol', 'optgroup', 'option', 'outpu
    'p', 'param', 'picture', 'pre', 'progress', 'q', 'rp', 'rt', 'ruby', 's'
    'samp', 'script', 'section', 'select', 'small', 'source', 'span', 'strong
    'style', 'sub', 'summary', 'sup', 'table', 'tbody', 'td', 'template',
    'textarea', 'tfoot', 'th', 'thead', 'time', 'title', 'tr', 'track', 'u',
    'ul', 'var', 'video', 'wbr'
  ]).forEach(tag => window[tag] = function(...args) {
    const props = typeof args[0] === 'object' && !(args[0] instanceof HTMLEle
    const elm = document.createElement(tag)
    props && assignDeep(elm, props)
    elm.append(...args.map(a => typeof a === 'string' ? document.createTextNo
    return elm
  })
  window['$'] = selector => document.querySelector(selector)
  window['$$'] = selector => Array.from(document.querySelectorAll(selector))
  window['createState'] = state => {
    state._updates = Object.fromEntries(Object.keys(state).map(s => [s, []]))
    state._update = s => state._updates[s].forEach(u => u())
    state.addUpdate = (s, u) => state._updates[s].push(u)
    return new Proxy(state, {set(o, p, v) {o[p] = v; o._update(p); return tr
  }
})()
```

Figure: Maybe this is all the framework code you need? (TinyJS)

Database Design

Databases

- Think about your data and how it will be used
- Schema vs schema-less
- Database extensions allowing other data use-cases
- Handling scaling of your application
- Use the right tool for the job

e.g. SQLite3 (widely deployed embeddable RDBMS), PostgreSQL (networked RDBMS), SQLAlchemy (ORM for RDBMS in Python), DuckDB (analytical DBMS), Redis and its implementations (key-value store DB), Neo4J (graph database), InfluxDB (time-series database), Prometheus (metrics database), Grafana (visualise time-series data), PostGIS (geospatial database extension), TimescaleDB (time-series database extension)

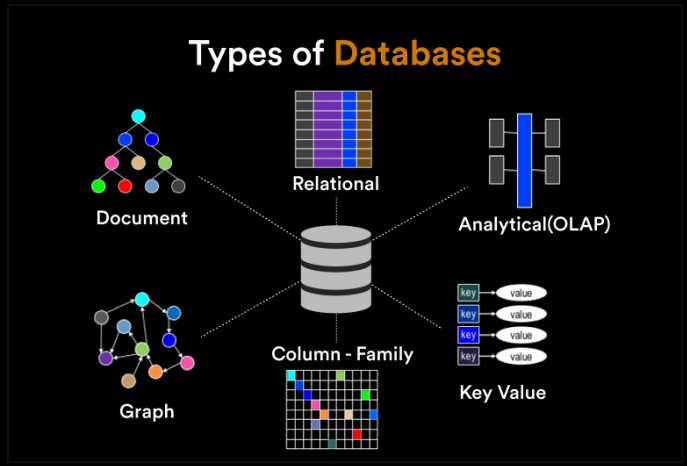


Figure: Different Types of Databases

Case Study 2: Database Types and use-cases

Key-value:

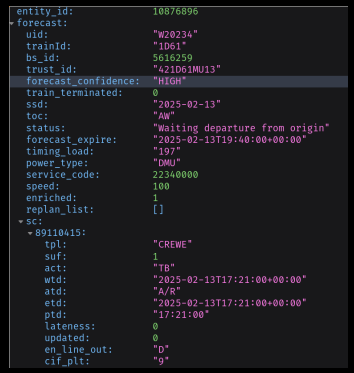


Figure: **Key-Value** database (acumen Forecasting)

Blob storage:

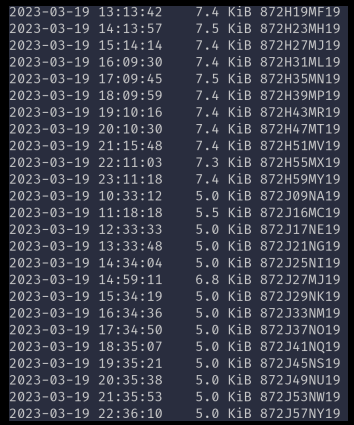


Figure: **Blob storage** database (acumen Entity Archive)

Relational:

uid	days_run	identity	origin_tiploc	origin_wtd	dest_tiploc	dest_wta
54212	0111000	8Y72	TONBES	0000-01-01T20:10:00Z	TONBES	0000-01-01T03:15:00Z
54236	0100000	8Y71	TONBES	0000-01-01T20:40:00Z	TONBES	0000-01-01T03:00:00Z
54249	0100000	3Y04	TONBDWG	0000-01-01T20:40:00Z	TONBDWG	0000-01-01T23:43:00Z
54255	0100000	8Y75	TONBES	0000-01-01T22:44:00Z	TONBES	0000-01-01T06:22:00Z
54724	0100000	3Y90	PURLSDG	0000-01-01T21:52:00Z	TONBDWG	0000-01-01T01:09:00Z
54726	0100000	8Y92	HORSHUS	0000-01-01T18:47:00Z	HORSHUS	0000-01-01T03:54:00Z
54728	0100000	8Y93	PRSP	0000-01-01T00:19:00Z	HORSHUS	0000-01-01T03:24:00Z
H05164	0100000	8Y83	EFNGSHD	0000-01-01T19:08:00Z	EFNGSHD	0000-01-01T01:00:00Z
H05171	0100000	8Y85	TOTTYD	0000-01-01T20:05:00Z	TOTTYD	0000-01-01T01:54:00Z
54774	0100000	3Y89	TOTTYD	0000-01-01T22:16:00Z	TOTTYD	0000-01-01T01:42:00Z
54780	0100000	3Y88	TOTTYD	0000-01-01T10:25:00Z	TOTTYD	0000-01-01T16:49:00Z
54821	0100000	3T26	EFLDTMD	0000-01-01T15:44:00Z	CKHLCSD	0000-01-01T16:07:00Z
54824	0100000	5Q94	FARNHMD	0000-01-01T22:06:00Z	ELGHBRM	0000-01-01T23:24:00Z

Figure: **Relational** database (acumen Train Service Database)

Graph:

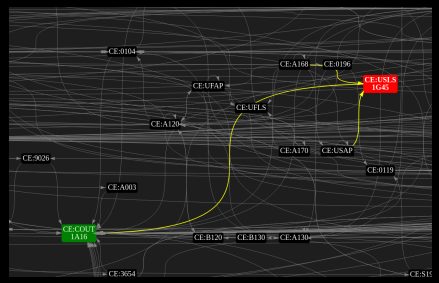


Figure: **Graph** database (acumen Network Model)

Time series:

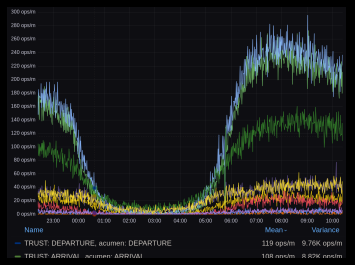


Figure: **Time-series** database (Prometheus)

Cloud Provider Managed Services

- Many different services on offer from numerous providers
- All offer basic VPS services, storage and networking
- Easy to be locked-in to a vendor
- Beware of hidden costs e.g. data ingress, DoS, metrics and logging
- Choice may not be up to you!

e.g. AWS, Azure, Google Cloud, Oracle Cloud Infrastructure, Hetzner, DigitalOcean, Cloudflare, Backblaze B2

Figure: Different cloud providers - there are many more (© 2025 Amazon, Microsoft, Google, Oracle, Hetzner, DigitalOcean, Cloudflare)

Cloud Automation Tools

Tools and Output Formats

- Use CLI tools provided by cloud providers
- Standard interchange format (JSON)
- Parsing and presenting data using tools

e.g JQ (JSON parser), AWS CLI (CLI tool for Amazon Web Services), etc.

Figure: CLI example output (querying AWS EC2 instance types)

Cloud Automation Tools

Infrastructure as Code (IaC)

- Version your infrastructure using Git
- Be careful of vendor lock-in with proprietary IaC
- Prefer configuration languages to YAML e.g. HCL
- Integrate with CI
- Beware of cost!

e.g. Terraform (infrastructure as code), OpenTofu (fork of Terraform following license change), Checkov (secure IaC configuration), Infracost (check cost impact of resources)/

```
data "aws_ami" "ubuntu" {
  most_recent = true

  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-
  }

  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }

  owners = ["099720109477"] # Canonical
}

resource "aws_instance" "web" {
  ami           = data.aws_ami.ubuntu.id
  instance_type = "t3.micro"

  tags = {
    Name = "HelloWorld"
  }
}
```

Copy

Figure: Terraform example for creating an AWS EC2 instance

Cloud Automation Tools

Kubernetes (K3s)

- Horizontal scaling of containerised applications
- Complex but solves common problems in software deployment
- Difficult to run stateful systems
- Many different implementations of the API
- Options to run locally and in the cloud
- Package management and custom resource definitions
- Is it necessary?

e.g. K3s (lightweight Linux K3s distro), Minikube (cross-platform K3s distro), Kubectl (K3s CLI), K9s (terminal UI for K3s), Helm (K3s package manager), cert-manager (automatically provision TLS certificates), arkade (quickly install K3s tools and packages)

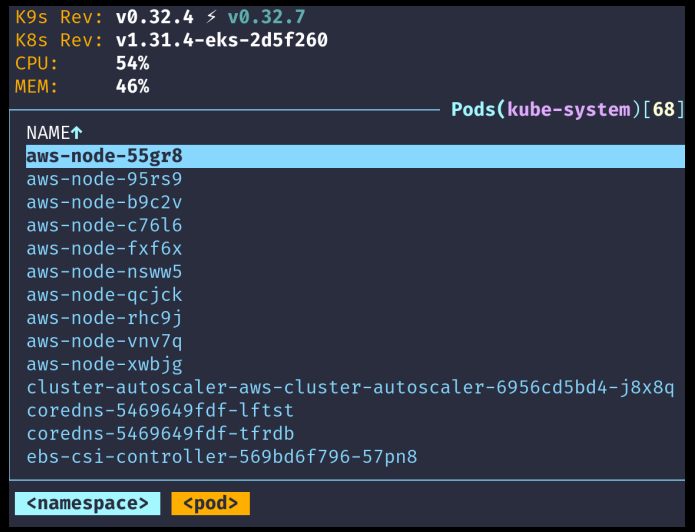


Figure: Pods running in a development Kubernetes cluster (K9s UI)

Case Study 3: Migration from AWS to Azure

- Planned and partially executed migration
- Focus on reusability of Terraform code
- Modules that are independent of cloud vendors
- Made easier by designing for open source platforms
- Limit dependency on vendor-specific APIs

e.g. Redis, PostgreSQL, Python, Terraform, Kubernetes

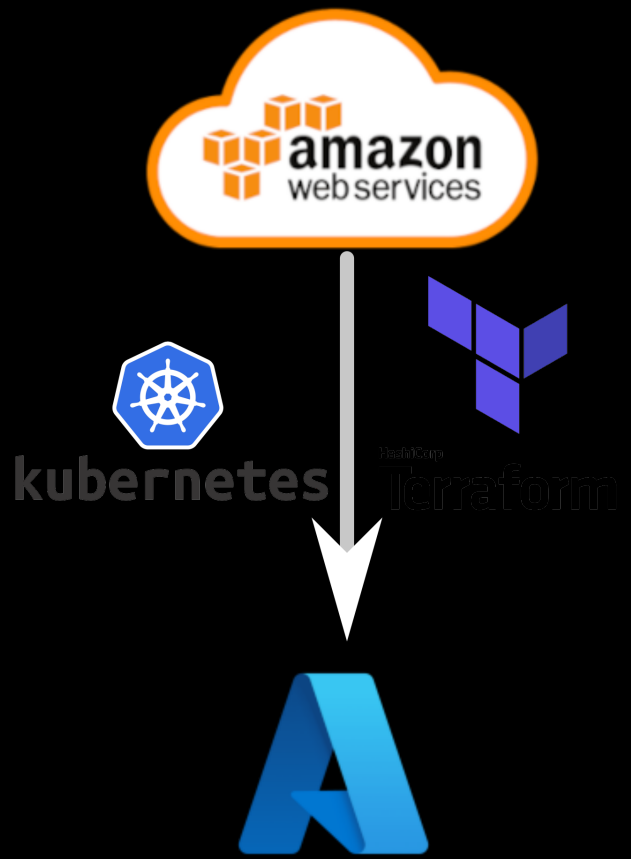


Figure: Migration from AWS to Azure (© 2025 Amazon, Microsoft, HashiCorp, Linux Foundation)

Conclusions

Common Themes

- Narrow Waists
- Open-Source standards and code
- Avoiding vendor lock-in
- Use the right tools for the job!

Conclusions

Future Work

- Improve the ecosystem
- Advocate for open-source tools and workflows
- Find projects that interest you
- It doesn't take much to be considered an expert!

Image References

- Divio Documentation System - `https://docs.divio.com/documentation-system/`
- Agile Lifecycle Model - `https://techgenies.com/agile-lifecycle-model-advantages-and-disadvantages/`
- TinyJS - `https://github.com/victorqribeiro/TinyJS/blob/master/tiny.js`
- Types of Databases - `https://www.founderjar.com/types-of-databases/`